1 . Why is Java so popular?

2 . What is platform independence?

3 . What is bytecode?

4 . Compare JDK vs JVM vs JRE&nbsp;

5 . What are the important differences between C++ and Java?

6 . What is the role for a classloader in Java?

7 . What are Wrapper classes?

8 . Why do we need Wrapper classes in Java?

9 . What are the different ways of creating Wrapper class instances?

10 . What are differences in the two ways of creating Wrapper classes?

11 . What is auto boxing?

12 . What are the advantages of auto boxing?

13 . What is casting?

14 . What is implicit casting?

15 . What is explicit casting?

16 . Are all String’s immutable?

17 . Where are String values stored in memory?

18 . Why should you be careful about String concatenation(+) operator in loops?

19 . How do you solve above problem?

20 . What are differences between String and StringBuffer?

21 . What are differences between StringBuilder and StringBuffer?

22 . Can you give examples of different utility methods in String class?

23 . What is a class?

24 . What is an object?

25 . What is state of an object?

26 . What is behavior of an object?

27 . What is the super class of every class in Java?

28 . Explain about toString method ?

29 . What is the use of equals method in Java?

30 . What are the important things to consider when implementing equals method?

31 . What is the Hashcode method used for in Java?

32 . Explain inheritance with examples .&nbsp;

33 . What is method overloading?

34 . What is method overriding?

35 . Can super class reference variable can hold an object of sub class?

36 . Is multiple inheritance allowed in Java?

37 . What is an interface?

38 . How do you define an interface?

39 . How do you implement an interface?

40 . Can you explain a few tricky things about interfaces?

41 . Can you extend an interface?

42 . Can a class extend multiple interfaces?

43 . What is an abstract class?

44 . When do you use an abstract class?

45 . How do you define an abstract method?

46 . Compare abstract class vs interface?

47 . What is a constructor?

48 . What is a default constructor?

49 . Will this code compile?

50 . How do you call a super class constructor from a constructor?

51 . Will this code compile?

52 . What is the use of this()?

53 . Can a constructor be called directly from a method?

54 . Is a super class constructor called even when there is no explicit call from a sub class constructor?

55 . What is polymorphism?

56 . What is the use of instanceof operator in Java?

57 . What is coupling?

58 . What is cohesion?

59 . What is encapsulation?

60 . What is an inner class?

61 . What is a static inner class?

62 . Can you create an inner class inside a method?

63 . What is an anonymous class?

64 . What is default class modifier?

65 . What is private access modifier?

66 . What is default or package access modifier?

67 . What is protected access modifier?

68 . What is public access modifier?

69 . What access types of variables can be accessed from a class in same package?

70 . What access types of variables can be accessed from a class in different package?

71 . What access types of variables can be accessed from a sub class in same package?

72 . What access types of variables can be accessed from a sub class in different package?

73 . What is the use of a final modifier on a class?

74 . What is the use of a final modifier on a method?

75 . What is a final variable?

76 . What is a final argument?

77 . What happens when a variable is marked as volatile?

78 . What is a static variable?

79 . Why should you always use blocks around if statement?

80 . Guess the output

81 . Guess the output

82 . Guess the output of this switch block .&nbsp;

83 . Guess the output of this switch block?

84 . Should default be the last case in a switch statement?

85 . Can a switch statement be used around a String

86 . Guess the output of this for loop

87 . What is an enhanced for loop?

88 . What is the output of the for loop below?

89 . What is the output of the program below?

90 . What is the output of the program below?

91 . Why is exception handling important?

92 . What design pattern is used to implement exception handling features in most languages?

93 . What is the need for finally block?

94 . In what scenarios is code in finally not executed?

95 . Will finally be executed in the program below?

96 . Is try without a catch is allowed?

97 . Is try without catch and finally allowed?

98 . Can you explain the hierarchy of exception handling classes?

99 . What is the difference between error and exception?

100 . What is the difference between checked exceptions and unchecked exceptions?

101 . How do you throw an exception from a method?

102 . What happens when you throw a checked exception from a method?

103 . What are the options you have to eliminate compilation errors when handling checked exceptions?

104 . How do you create a custom exception?

105 . How do you handle multiple exception types with same exception handling block?

106 . Can you explain about try with resources?

107 . How does try with resources work?

108 . Can you explain a few exception handling best practices?

109 . What are the default values in an array?

110 . How do you loop around an array using enhanced for loop?

111 . How do you print the content of an array?

112 . How do you compare two arrays?

113 . What is an enum?

114 . Can you use a switch statement around an enum?

115 . What are variable arguments or varargs?

116 . What are asserts used for?

117 . When should asserts be used?

118 . What is garbage collection?

119 . Can you explain garbage collection with an example?

120 . When is garbage collection run?

121 . What are best practices on garbage collection?

122 . What are initialization blocks?

123 . What is a static initializer?

124 . What is an instance initializer block?

125 . What is tokenizing?

126 . Can you give an example of tokenizing?

127 . What is serialization?

128 . How do you serialize an object using serializable interface?

129 . How do you de-serialize in Java?

130 . What do you do if only parts of the object have to be serialized?

131 . How do you serialize a hierarchy of objects?

132 . Are the constructors in an object invoked when it is de-serialized?

133 . Are the values of static variables stored when an object is serialized?

134 . Why do we need collections in Java?

135 . What are the important interfaces in the collection hierarchy?

136 . What are the important methods that are declared in the collection interface?

137 . Can you explain briefly about the List interface?

138 . Explain about ArrayList with an example?

139 . Can an ArrayList have duplicate elements?

140 . How do you iterate around an ArrayList using iterator?

141 . How do you sort an ArrayList?

142 . How do you sort elements in an ArrayList using comparable interface?

143 . How do you sort elements in an ArrayList using comparator interface?

144 . What is vector class? How is it different from an ArrayList?

145 . What is linkedList? What interfaces does it implement? How is it different from an ArrayList?

146 . Can you briefly explain about the Set interface?

147 . What are the important interfaces related to the Set interface?

148 . What is the difference between Set and sortedSet interfaces?

149 . Can you give examples of classes that implement the Set interface?

150 . What is a HashSet?

151 . What is a linkedHashSet? How is different from a HashSet?

152 . What is a TreeSet? How is different from a HashSet?

153 . Can you give examples of implementations of navigableSet?

154 . Explain briefly about Queue interface?

155 . What are the important interfaces related to the Queue interface?

156 . Explain about the Deque interface?

157 . Explain the BlockingQueue interface?

158 . What is a priorityQueue?

159 . Can you give example implementations of the BlockingQueue interface?

160 . Can you briefly explain about the Map interface?

161 . What is difference between Map and sortedMap?

162 . What is a HashMap?

163 . What are the different methods in a Hash Map?

164 . What is a TreeMap? How is different from a HashMap?

165 . Can you give an example of implementation of navigableMap interface?

166 . What are the static methods present in the collections class?

167 . What is the difference between synchronized and concurrent collections in Java?

168 . Explain about the new concurrent collections in Java?

169 . Explain about copyonwrite concurrent collections approach?

170 . What is compareandswap approach?

171 . What is a lock? How is it different from using synchronized approach?

172 . What is initial capacity of a Java collection?

173 . What is load factor?

174 . When does a Java collection throw UnsupportedOperationException?

175 . What is difference between fail-safe and fail-fast iterators?

176 . What are atomic operations in Java?

177 . What is BlockingQueue in Java?

178 . What are Generics?

179 . Why do we need Generics? Can you give an example of how Generics make a program more flexible?

180 . How do you declare a generic class?

181 . What are the restrictions in using generic type that is declared in a class declaration?

182 . How can we restrict Generics to a subclass of particular class?

183 . How can we restrict Generics to a super class of particular class?

184 . Can you give an example of a generic method?

185 . What is the need for threads in Java?

186 . How do you create a thread?

187 . How do you create a thread by extending thread class?

188 . How do you create a thread by implementing runnable interface?

189 . How do you run a thread in Java?

190 . What are the different states of a thread?

191 . What is priority of a thread? How do you change the priority of a thread?

192 . What is executorservice?

193 . Can you give an example for executorservice?

194 . Explain different ways of creating executor services .&nbsp;

195 . How do you check whether an executionservice task executed successfully?

196 . What is callable? How do you execute a callable from executionservice?

197 . What is synchronization of threads?

198 . Can you give an example of a synchronized block?

199 . Can a static method be synchronized?

200 . What is the use of join method in threads?

201 . Describe a few other important methods in threads?

202 . What is a deadlock?

203 . What are the important methods in Java for inter-thread communication?

204 . What is the use of wait method?

205 . What is the use of notify method?

206 . What is the use of notifyall method?

207 . Can you write a synchronized program with wait and notify methods?

208 . What is functional programming?

209 . Can you give an example of functional programming?

210 . What is a stream?

211 . Explain about streams with an example?

what are intermediate operations in streams?

212 . What are terminal operations in streams?

213 . What are method references?

214 . What are lambda expressions?

215 . Can you give an example of lambda expression?

216 . Can you explain the relationship between lambda expression and functional interfaces?

217 . What is a predicate?

218 . What is the functional interface – function?

219 . What is a consumer?

220 . Can you give examples of functional interfaces with multiple arguments?

1. Compile time runtime exceptions and list of exceptions

java.lang

1. [Throwable](https://docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html)
2. [Exception](https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html)
   1. [CloneNotSupportedException](https://docs.oracle.com/javase/8/docs/api/java/lang/CloneNotSupportedException.html)
   2. [InterruptedException](https://docs.oracle.com/javase/8/docs/api/java/lang/InterruptedException.html)
   3. [ReflectiveOperationException](https://docs.oracle.com/javase/8/docs/api/java/lang/ReflectiveOperationException.html)
      1. [ClassNotFoundException](https://docs.oracle.com/javase/8/docs/api/java/lang/ClassNotFoundException.html)
      2. [IllegalAccessException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalAccessException.html)
      3. [InstantiationException](https://docs.oracle.com/javase/8/docs/api/java/lang/InstantiationException.html)
      4. [NoSuchFieldException](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchFieldException.html)
      5. [NoSuchMethodException](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchMethodException.html)
   4. [RuntimeException](https://docs.oracle.com/javase/8/docs/api/java/lang/RuntimeException.html)
      1. [ArithmeticException](https://docs.oracle.com/javase/8/docs/api/java/lang/ArithmeticException.html)
      2. [ArrayStoreException](https://docs.oracle.com/javase/8/docs/api/java/lang/ArrayStoreException.html)
      3. [ClassCastException](https://docs.oracle.com/javase/8/docs/api/java/lang/ClassCastException.html)
      4. [EnumConstantNotPresentException](https://docs.oracle.com/javase/8/docs/api/java/lang/EnumConstantNotPresentException.html)
      5. [IllegalArgumentException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalArgumentException.html)
         1. [IllegalThreadStateException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalThreadStateException.html)
         2. [NumberFormatException](https://docs.oracle.com/javase/8/docs/api/java/lang/NumberFormatException.html)
      6. [IllegalMonitorStateException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalMonitorStateException.html)
      7. [IllegalStateException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalStateException.html)
      8. [IndexOutOfBoundsException](https://docs.oracle.com/javase/8/docs/api/java/lang/IndexOutOfBoundsException.html)
         1. [ArrayIndexOutOfBoundsException](https://docs.oracle.com/javase/8/docs/api/java/lang/ArrayIndexOutOfBoundsException.html)
         2. [StringIndexOutOfBoundsException](https://docs.oracle.com/javase/8/docs/api/java/lang/StringIndexOutOfBoundsException.html)
      9. [NegativeArraySizeException](https://docs.oracle.com/javase/8/docs/api/java/lang/NegativeArraySizeException.html)
      10. [NullPointerException](https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html)
      11. [SecurityException](https://docs.oracle.com/javase/8/docs/api/java/lang/SecurityException.html)
      12. [TypeNotPresentException](https://docs.oracle.com/javase/8/docs/api/java/lang/TypeNotPresentException.html)
      13. [UnsupportedOperationException](https://docs.oracle.com/javase/8/docs/api/java/lang/UnsupportedOperationException.html)
3. [Error](https://docs.oracle.com/javase/8/docs/api/java/lang/Error.html)
   1. [AssertionError](https://docs.oracle.com/javase/8/docs/api/java/lang/AssertionError.html)
   2. [LinkageError](https://docs.oracle.com/javase/8/docs/api/java/lang/LinkageError.html)
      1. [BootstrapMethodError](https://docs.oracle.com/javase/8/docs/api/java/lang/BootstrapMethodError.html)
      2. [ClassCircularityError](https://docs.oracle.com/javase/8/docs/api/java/lang/ClassCircularityError.html)
      3. [ClassFormatError](https://docs.oracle.com/javase/8/docs/api/java/lang/ClassFormatError.html)
         1. [UnsupportedClassVersionError](https://docs.oracle.com/javase/8/docs/api/java/lang/UnsupportedClassVersionError.html)
      4. [ExceptionInInitializerError](https://docs.oracle.com/javase/8/docs/api/java/lang/ExceptionInInitializerError.html)
      5. [IncompatibleClassChangeError](https://docs.oracle.com/javase/8/docs/api/java/lang/IncompatibleClassChangeError.html)
         1. [AbstractMethodError](https://docs.oracle.com/javase/8/docs/api/java/lang/AbstractMethodError.html)
         2. [IllegalAccessError](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalAccessError.html)
         3. [InstantiationError](https://docs.oracle.com/javase/8/docs/api/java/lang/InstantiationError.html)
         4. [NoSuchFieldError](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchFieldError.html)
         5. [NoSuchMethodError](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchMethodError.html)
      6. [NoClassDefFoundError](https://docs.oracle.com/javase/8/docs/api/java/lang/NoClassDefFoundError.html)
      7. [UnsatisfiedLinkError](https://docs.oracle.com/javase/8/docs/api/java/lang/UnsatisfiedLinkError.html)
      8. [VerifyError](https://docs.oracle.com/javase/8/docs/api/java/lang/VerifyError.html)
   3. [ThreadDeath](https://docs.oracle.com/javase/8/docs/api/java/lang/ThreadDeath.html)
   4. [VirtualMachineError](https://docs.oracle.com/javase/8/docs/api/java/lang/VirtualMachineError.html)
      1. [InternalError](https://docs.oracle.com/javase/8/docs/api/java/lang/InternalError.html)
      2. [OutOfMemoryError](https://docs.oracle.com/javase/8/docs/api/java/lang/OutOfMemoryError.html)
      3. [StackOverflowError](https://docs.oracle.com/javase/8/docs/api/java/lang/StackOverflowError.html)
      4. [UnknownError](https://docs.oracle.com/javase/8/docs/api/java/lang/UnknownError.html)
4. Types of class loader

Java class loaders can be broadly classified into below categories:

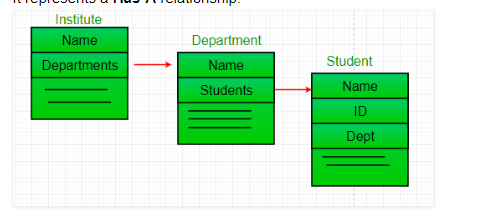
* Bootstrap Class Loader  
  Bootstrap class loader loads java’s core classes like java.lang, java.util etc. These are classes that are part of java runtime environment. Bootstrap class loader is native implementation and so they may differ across different JVMs.
* Extensions Class Loader  
  JAVA\_HOME/jre/lib/ext contains jar packages that are extensions of standard core java classes. Extensions class loader loads classes from this ext folder. Using the system environment propery java.ext.dirs you can add ‘ext’ folders and jar files to be loaded using extensions class loader.
* System Class Loader  
  Java classes that are available in the java classpath are loaded using System class loader.

1. **Difference between Composition, Aggregation and Association in OOP?** ([answer](http://javarevisited.blogspot.sg/2014/02/ifference-between-association-vs-composition-vs-aggregation.html))  
   If two objects are related to each other, they are said to be associated with each other. Composition and Aggregation are two forms of association in object-oriented programming. The composition is stronger association than Aggregation. In Composition, one object is OWNER of another object while in Aggregation one object is just USER of another object. If an object A is composed of object B then B doesn't exist if A ceased to exists, but if object A is just an aggregation of object B then B can exists even if A ceased to exist.

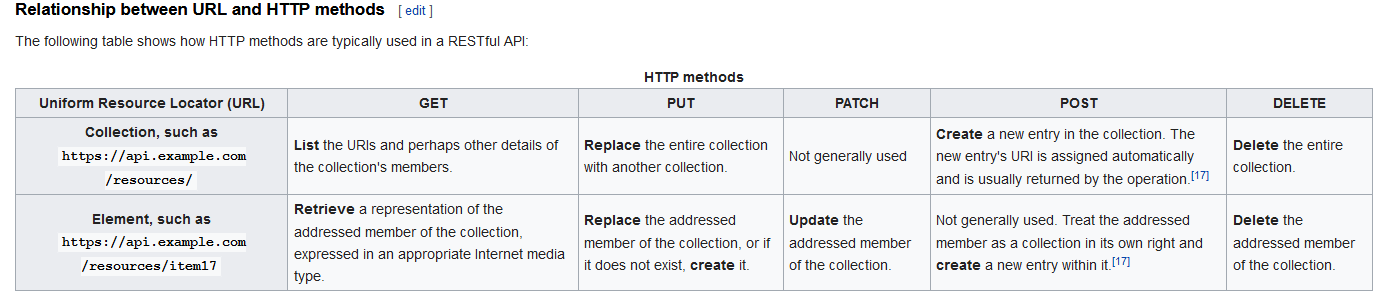
<https://www.geeksforgeeks.org/association-composition-aggregation-java/>

Is a Relation : A Library has books (Without books there is no library close relation)

Has a Relation



1. Rest –Representation state transfer

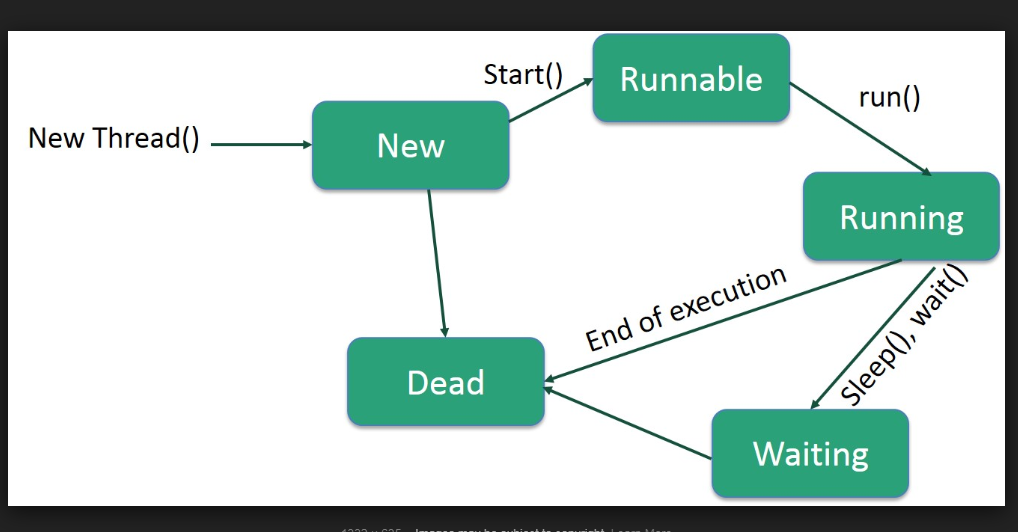


1. Suitable Collection for shopping cart(Hash Map)
2. Runnable vs Thread

When we want to extend other classes we need to go runnable

By Extending thread we have complete control over thread

1. Hash map with same hash code with multiple objects
2. Thread Life Cycle



1. Yield VS Join Vs wait Vs Notify

Yield – will give control to high priority thread

Join -  **join**() method which allows one **thread** to wait until another **thread** completes its execution.

Wait – it won’t release lock until it notify

Setpriority – sets time priority

1. How you handled out of space memory issue
2. Static variable are thread safe

No it belongs to class memory

1. Volatile Keyword

When multiple thread access same variable it will fetch latest variable

1. What is Intrepputed or Illegal state Exception

When some thread uses as intrupt method by some other thread we get this exception

1. Why wait and notify and notify all are at Object Level

Those will perform lock operations on Objects not on thread that’s y those are in object level

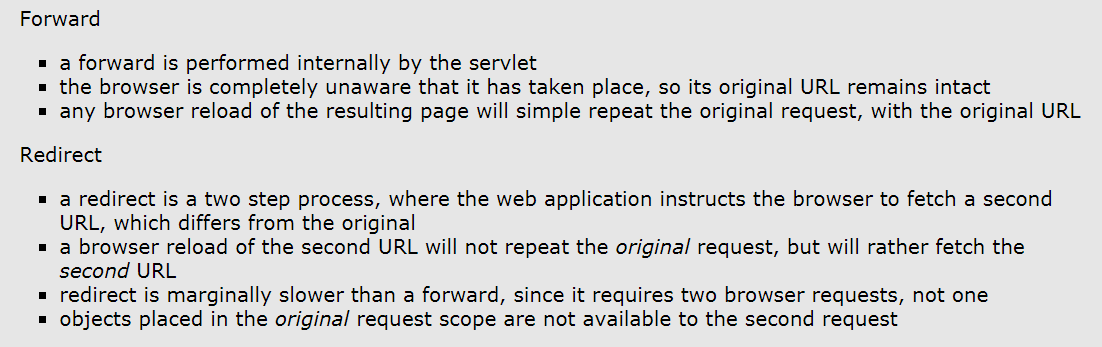
1. Thread group vs Thread pool

Thread group combining two or more threads for specific operation

ThreadPool will help the max number of connection

(Using Executor Service)

1. Action Forward vs Action Redirect



### **Methods of Object class**

|  |
| --- |
| The Object class provides many methods. They are as follows: |

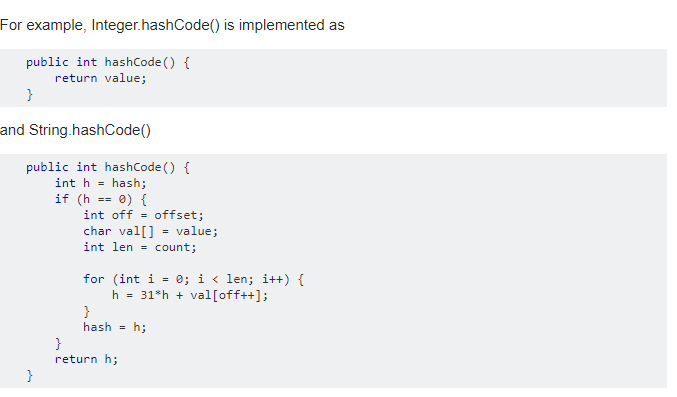
|  |  |
| --- | --- |
| **Method** | **Description** |
| public final Class getClass() | returns the Class class object of this object. The Class class can further be used to get the metadata of this class. |
| public int hashCode() | returns the hashcode number for this object. |
| public boolean equals(Object obj) | compares the given object to this object. |
| protected Object clone() throws CloneNotSupportedException | creates and returns the exact copy (clone) of this object. |
| public String toString() | returns the string representation of this object. |
| public final void notify() | wakes up single thread, waiting on this object's monitor. |
| public final void notifyAll() | wakes up all the threads, waiting on this object's monitor. |
| public final void wait(long timeout)throws InterruptedException | causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait(long timeout,int nanos)throws InterruptedException | causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait()throws InterruptedException | causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method). |
| protected void finalize()throws Throwable | is invoked by the garbage collector before object is being garbage collected. |

1. How the hash value calculated

// bitwise xor

// 0101 ^ 0111=0010

Object.hashcode(key) ^ Object.hashcode(value);



1. Complexity of collections
2. Diff preparedstatement,callable statement and statement in jdbc
3. Array.length vs Arraylist.size()

use length attribute to get number of elements in a array, also known as length, and for same thing in Collection classes e.g. [ArrayList](http://java67.blogspot.sg/2012/11/java-arraylist-example-contains-add-set.html), [Vector](http://java67.blogspot.sg/2012/09/arraylist-vs-vector-in-java-interview.html), use size() method

1. Multiple start() method in threads

It will throw illegal state modification exception

1. Difference between an array and ArrayList in Java

Array length is fixed and array list support primitive datatype.

1. Semaphore

It has accuire and release methods , it will allow only fixed count of connections only in threads .

It uses for database connections

1. Count down latch vs cyclic barrier

A CountDownLatch can be used only once in a program(until it’s count reaches 0).

A CyclicBarrier can be used again and again once all the threads in a barriers is released.

1. Methods in collection util (reverse,syncronise,umodifiedlist)
2. Diff between collection and collections

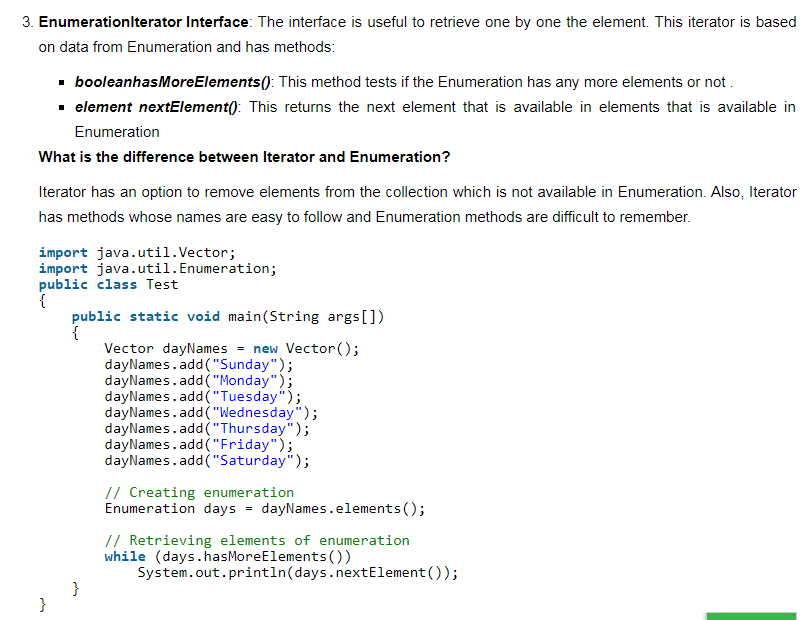
Collection is interface and collections is a util class

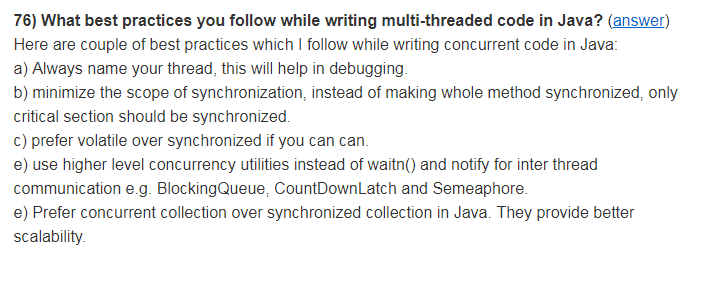
1. **Can we overload static methods?**  
   The answer is ‘**Yes**’. We can have two ore more static methods with same name, but differences in input parameters. For example, consider the following Java program. Refer [this](https://www.geeksforgeeks.org/can-we-overload-or-override-static-methods-in-java/) for details.
2. **Can we overload methods that differ only by static keyword?**  
   We **cannot** overload two methods in Java if they differ only by static keyword (number of parameters and types of parameters is same). See following Java program for example. Refer [this](https://www.geeksforgeeks.org/can-we-overload-or-override-static-methods-in-java/) for details.
3. **Can we overload main() in Java?**  
   Like other static methods, we **can** [overload main() in Java](https://www.geeksforgeeks.org/gfact-48-overloading-main-in-java/). Refer overloading main() in Java for more details.
4. Object level locking and class level locking
5. What is externilzation

Customizing serlization process

public void writeExternal(ObjectOutput out) throws IOException {

public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {

1. How do you find memory usage from Java program? How much percent of the heap is used?  
   You can use memory related methods from java.lang.Runtime class to get the free memory, total memory and maximum heap memory in Java.  By using these methods, you can find out how many percents of the heap is used and how much heap space is remaining. Runtime.freeMemory() return amount of free memory in bytes, Runtime.totalMemory() returns total memory in bytes and Runtime.maxMemory() returns maximum memory in bytes.  
   
2. Spring bean life cycle



1. Transaction management
2. Things need to be done for basic operation with database

Loading class driver ,Register service, making connection ,executing query, close connection

1. Design patterns used in Spring (Singleton,FatoryBean,Abstract Bean)
2. How to find missing number in array (use the formula n(n+1)/2)
3. Concurrency package
4. How can you measure the performance of the hasmap

We can measure based on two factors

* 1. Load factor 0.75 (Tell you )
  2. Initial capacity

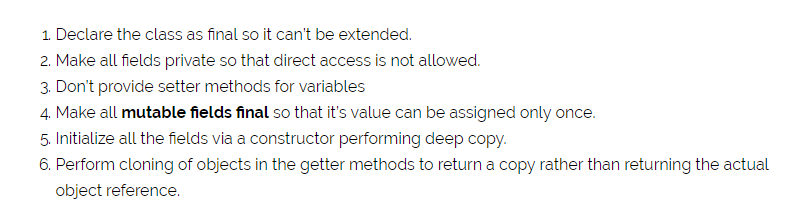
1. Can we add more null keys to hashmap

Yes, but it will override the old value

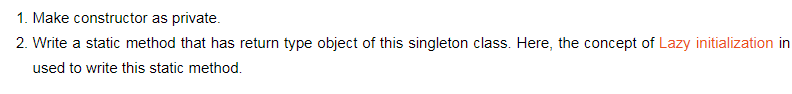
1. What will happen if two values with key

Latest value overrides old value

1. How to make object as immutable



1. How to make object as singleton



1. How to make immutable objects as read only
2. How thread creations works in JVM
3. Joins in SQL
4. When to use Thread and Runnable class
5. Can we give private modifier in subclass to public modifier in super class

No

1. How singleton works in multi thread env
2. String reversal in linked list



1. Nth highest salary in sql for all dB

select \* from Products p1 where n-1 = (select count(Distinct(price)) from Products p2 where p2.price>p1.price)

1. Swagger documentation customization
2. Internal implementation of Concurrency hashmap
3. When to use concurrency hasmap vs normal hashmap
4. Functional interface usage ?
5. SOLID design principles

# Design Principles [S.O.L.I.D.] in Java

June 7, 2013 by Lokesh Gupta

Classes are the building blocks of your java application. If these blocks are not strong, your building (i.e. application) is going to face the tough time in future. This essentially means that not so well-written can lead to very difficult situations when the application scope goes up or application faces certain design issues either in production or maintenance.

On the other hand, set of well designed and written classes can speed up the coding process by leaps and bounds, while reducing the number of bugs in comparison.

In this post, I will list down 5 most recommended design principles, you should keep in mind, while writing your classes. These design principles are called SOLID, in short. They also form the [**best practices**](https://howtodoinjava.com/category/best-practices/) to be followed for designing your application classes.

**Table Of Contents**

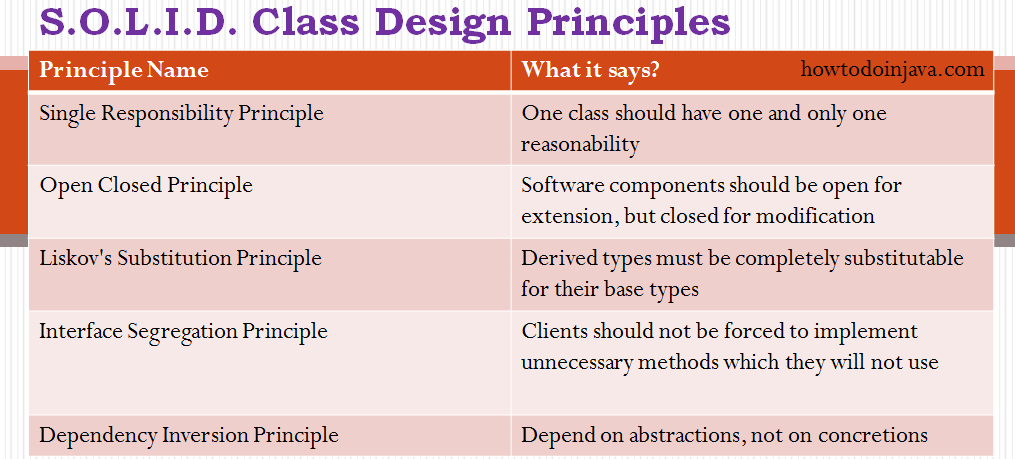
[Single Responsibility Principle](https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/#SRP)

[Open Closed Principle](https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/#OCP)

[Liskov's Substitution Principle](https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/#LSP)

[Interface Segregation Principle](https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/#ISP)

[Dependency Inversion Principle](https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/#DI)

5 java class design principles

Lets drill down all of them one by one.

## ****Single Responsibility Principle****

The name of the principle says it all:

**"One class should have one and only one responsibility"**

In other words, you should write, change and maintain a class for only one purpose. If it is model class then it should strictly represent only one actor/ entity. This will give you the flexibility to make changes in future without worrying the impacts of changes for another entity.

Similarly, If you are writing service/manager class then it should contain only that part of method calls and nothing else. Not even utility global functions related to module. Better separate them in another globally accessible class file. This will help in maintaining the class for that particular purpose, and you can decide the visibility of class to specific module only.

## ****Open Closed Principle****

This is second important rule which you should keep in mind while designing your application. It says:

**"Software components should be open for extension, but closed for modification"**

What does it mean?? It means that your classes should be designed such a way that whenever fellow developers wants to change the flow of control in specific conditions in application, all they need to extend your class and override some functions and that’s it.

If other developers are not able to design desired behavior due to constraints put by your class, then you should reconsider changing your class. I do not mean here that anybody can change the whole logic of your class, but he/she should be able to override the options provided by software in unharmful way permitted by software.

For example, if you take a look into any good framework like struts or spring, you will see that you can not change their core logic and request processing, BUT you modify the desired application flow just by extending some classes and plugin them in configuration files.

## ****Liskov’s Substitution Principle****

This principle is a variation of previously discussed open closed principle. It says:

**"Derived types must be completely substitutable for their base types"**

It means that the classes fellow developer created by extending your class should be able to fit in application without failure. I.e. if a fellow developer poorly extended some part of your class and injected into framework/ application then it should not break the application or should not throw fatal [**exceptions**](https://howtodoinjava.com/best-practices/java-exception-handling-best-practices/).

This can be insured by using strictly following first rule. If your base class is doing one thing strictly, the fellow developer will override only one feature incorrectly in worst case. This can cause some errors in one area, but whole application will not do down.

## ****Interface Segregation Principle****

This principle is my favorite one. It is applicable to interfaces as single responsibility principle holds to classes. It says:

**"Clients should not be forced to implement unnecessary methods which they will not use"**

Take an example. Developer Alex created an interface Reportable and added two methods generateExcel() and generatedPdf(). Now client ‘A’ wants to use this interface but he intend to use reports only in PDF format and not in excel. Will he achieve the functionality easily.

NO. He will have to implement two methods, out of which one is extra burden put on him by designer of software. Either he will implement another method or leave it blank. So are not desired cases, right??

So what is the solution? Solution is to create two interfaces by breaking the existing one. They should be like PdfReportable and ExcelReportable. This will give the flexibility to user to use only required functionality only.

## ****Dependency Inversion Principle****

Most of us are already familiar with the words used in principle’s name. It says:

**"Depend on abstractions, not on concretions"**

In other words. you should design your software in such a way that various modules can be separated from each other using an abstract layer to bind them together. The classical use of this principle of [**BeanFactory**](https://howtodoinjava.com/spring/spring-core/different-spring-3-ioc-containers-with-example/) in [**spring framework**](https://howtodoinjava.com/java-spring-framework-tutorials/). In spring framework, all modules are provided as separate components which can work together by simply injected dependencies in other module. They are so well closed in their boundaries that you can use them in other software modules apart from spring with same ease.

This has been achieved by dependency inversion and open closed principles. All modules expose only abstraction which is useful in extending the functionality or plugin in another module.

These were **five class design principles** which makes the best practices to be followed to design your application classes. Let me know of your thoughts.

1. Data encapsulations vs abstraction
2. Searalize the static methods

No you can serialize static methods

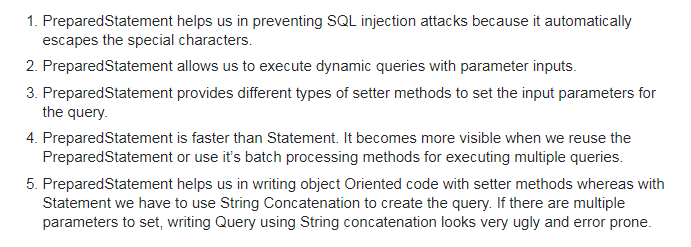
1. Jvm architecture
2. Fail fast vs fail safe
3. Demon thread
4. How you know object is available for garbage collection

Object reference changed or allocation object as null

1. Hasmap internal working
2. Return type of hash code

int

1. Comparable vs comparator
2. Big annotation of Hash map
3. Diff of java version and features
4. Thread pooling and disadvatages (memoryleak, deadlock)
5. Why the performance of cuncurrancy hashmap is fast
6. What is the difference between CyclicBarrier and CountDownLatch in Java
7. Methods of Object (clone,getclass,wait,equals,notify,notifyall,hascode,tostring,finalize)
8. Prepared statement vs statement (SQL injection)



1. What is differnance between sumbit() and execute() in threadpool?

Both methods are ways to submit a task to thread pools but there is a slight difference between them. execute(Runnable command) is defined in Executor interface and executes given task in future, but more importantly, it does not return anything. Its return type is void. On other hand submit() is an overloaded method, it can take either Runnable or Callable task and can return Future object which can hold the pending result of computation. This method is defined on ExecutorService interface, which extends Executor interface, and every other thread pool class e.g. ThreadPoolExecutor or ScheduledThreadPoolExecutor gets these methods.

1. Why is Java called the ‘Platform Independent Programming Language’?

Platform independence means that execution of your program does not dependent on type of operating system(it could be any : Linux, windows, Mac ..etc). So compile code only once and run it on any System (In C/C++, we need to compile the code for every machine on which we run it). Java is both compiler(javac) and interpreter(jvm) based lauguage. Your java source code is first compiled into byte code using javac compiler. This byte code can be easily converted to equivalent machine code using JVM. JVM(Java Virtual Machine) is available in all operating systems we install. Hence, byte code generated by javac is universal and can be converted to machine code on any operating system, this is the reason why java is platform independent.

1. What is livelock and deadlock?

This question is extension of previous interview question. A livelock is similar to a deadlock, except that the states of the threads or processes involved in the livelock constantly change with regard to one another, without any one progressing further. Livelock is a special case of resource starvation. A real-world example of livelock occurs when two people meet in a narrow corridor, and each tries to be polite by moving aside to let the other pass, but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time. In short, the main difference between livelock and deadlock is that in former state of process change but no progress is made.

**45)Explain Final keyword in java?**

Final keyword in java is used to restrict usage of variable, class and method.  
   
Variable: Value of Final variable is constant, you can not change it.  
Method: you can’t override a Final method.  
Class: you can’t inherit from Final class.

Refer [this](http://quiz.geeksforgeeks.org/final-keyword-java/) for details

**46)When is the super keyword used?**  
   
super keyword is used to refer:

* immediate parent class constructor,
* immediate parent class variable,
* immediate parent class method.

Refer [this](http://quiz.geeksforgeeks.org/super-keyword/) for details.

**47)What is the difference between StringBuffer and String?**

String is an Immutable class, i.e. you can not modify its content once created. While StringBuffer is a mutable class, means you can change its content later. Whenever we alter content of String object, it creates a new string and refer to that,it does not modify the existing one. This is the reason that the performance with StringBuffer is better than with String.  
Refer [this](https://www.geeksforgeeks.org/g-fact-27-string-vs-stringbuilder-vs-stringbuffer/) for details.

**48)Why multiple inheritance is not supported in java?**

Java supports multiple inheritance but not through classes, it supports only through its interfaces. The reason for not supporting multiple inheritance is to avoid the conflict and complexity arises due to it and keep Java a Simple Object Oriented Language. If we recall [this in C++](https://www.geeksforgeeks.org/multiple-inheritance-in-c/), there is a special case of multiple inheritance (diamond problem) where you have a multiple inheritance with two classes which have methods in conflicts. So, Java developers decided to avoid such conflicts and didn’t allow multiple inheritance through classes at all.

**49)Can a top level class be private or protected?**

Top level classes in java can’t be private or protected, but inner classes in java can. The reason for not making a top level class as private is very obvious, because nobody can see a private class and thus they can not use it. Declaring a class as protected also doesn’t make any sense. The only difference between default visibility and protected visibility is that we can use it in any package by inheriting it. Since in java there is no such concept of package inheritance, defining a class as protected is no different from default.

**50)What is the difference between ‘throw’ and ‘throws’ in Java Exception Handling?**

Following are the differences between two:

* throw keyword is used to throw Exception from any method or static block whereas throws is used to indicate that which Exception can possibly be thrown by this method
* If any method throws checked Exception, then caller can either handle this exception(using try catch block )or can re throw it by declaring another ‘throws’ clause in method declaration.
* throw clause can be used in any part of code where you feel a specific exception needs to be thrown to the calling method

E.g.  
**throw**  
throw new Exception(“You have some exception”)  
throw new IOException(“Connection failed!!”)  
**throws**  
throws IOException, NullPointerException, ArithmeticException

**51)What is finalize() method?**  
   
Unlike c++ , we don’t need to destroy objects explicitly in Java. ‘[Garbage Collector](https://www.geeksforgeeks.org/garbage-collection-java/)‘ does that automatically for us. Garbage Collector checks if no references to an object exist, that object is assumed to be no longer required, and the memory occupied by the object can be freed. Sometimes an object can hold non-java resources such as file handle or database connection, then you want to make sure these resources are also released before object is destroyed. To perform such operation Java provide protected void finalize() in object class. You can override this method in your class and do the required tasks. Right before an object is freed, the java run time calls the finalize() method on that object. Refer [this](https://www.geeksforgeeks.org/garbage-collection-java/) for more details.

**52)Difference in Set and List interface?**

Set and List both are child interface of Collection interface. There are following two main differences between them

* List can hold duplicate values but Set doesn’t allow this.
* In List interface data is present in the order you inserted but in the case of Set insertion order is not preserved.

**53)What will happen if you put System.exit(0) on try or catch block? Will finally block execute?**  
   
By Calling System.exit(0) in try or catch block, we can skip the finally block. System.exit(int) method can throw a SecurityException. If Sysytem.exit(0) exits the JVM without throwing that exception then finally block will not execute. But, if System.exit(0) does throw security exception then finally block will be executed.  
**Can we** [**Overload or Override static methods in java**](https://www.geeksforgeeks.org/can-we-overload-or-override-static-methods-in-java/) **?**

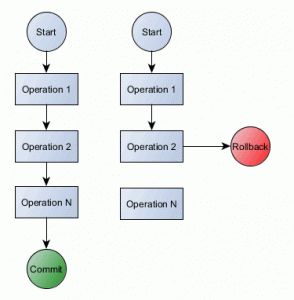
* **Overriding :** Overriding is related to run-time polymorphism. A subclass (or derived class) provides a specific implementation of a method in superclass (or base class) at runtime.
* **Overloading:** Overloading is related to compile time (or static) polymorphism. This feature allows different methods to have same name, but different signatures, especially number of input parameters and type of input paramaters.
* **Can we overload static methods?**The answer is **‘Yes’**. We can have two ore more static methods with same name, but differences in input parameters
* **Can we Override static methods in java?**We can declare static methods with same signature in subclass, but it is not considered overriding as there won’t be any run-time polymorphism. Hence the answer is **‘No’**. Static methods cannot be overridden because method overriding only occurs in the context of dynamic (i.e. runtime) lookup of methods. Static methods (by their name) are looked up statically (i.e. at compile-time).

Read [more](https://www.geeksforgeeks.org/can-we-overload-or-override-static-methods-in-java/)

1. ACID Prinicples

Transactions are omnipresent in today’s enterprise systems, providing data integrity even in highly concurrent environments. So let’s get started by first defining the term and the context where you might usually employ it.

A transaction is a collection of read/write operations succeeding only if all contained operations succeed.

[](https://vladmihalcea.files.wordpress.com/2014/01/transaction-workflow1.gif)

Inherently a transaction is characterized by four properties (commonly referred as ACID):

1. Atomicity
2. Consistency
3. Isolation
4. Durability

In a relational database, every SQL statement must execute in the scope of a transaction. Without defining the transaction boundaries explicitly, the database is going to use an implicit transaction which is wraps around every individual statement. The implicit transaction begins before the statement is executed and end (commit or rollback) after the statement is executed.  
The implicit transaction mode is commonly known as [autocommit](http://en.wikipedia.org/wiki/Autocommit).

For an enterprise application, the auto-commit mode is something you’d generally want to avoid since it has serious performance penalties, and it doesn’t allow you to include multiple [DML](http://en.wikipedia.org/wiki/Data_manipulation_language)operations in a single atomic Unit of Work.

It’s very important to understand those, hence we will discuss each and every one of them as follows.

# Atomicity

Atomicity takes individual operations and turns them into an all-or-nothing unit of work, succeeding if and only if all contained operations succeed.

A transaction might encapsulate a state change (unless it is a read-only one). A transaction must always leave the system in a consistent state, no matter how many concurrent transactions are interleaved at any given time.

# Consistency

Consistency means that constraints are enforced for every committed transaction. That implies that all Keys, Data types, Checks and Trigger are successful and no constraint violation is triggered.

# Isolation

Transactions require concurrency control mechanisms, and they guarantee correctness even when being interleaved. Isolation brings us the benefit of hiding uncommitted state changes from the outside world, as failing transactions shouldn’t ever corrupt the state of the system. Isolation is achieved through [concurrency control](http://en.wikipedia.org/wiki/Concurrency_control) using pessimistic or optimistic locking mechanisms.

# Durability

A successful transaction must permanently change the state of a system, and before ending it, the state changes are recorded in a persisted [transaction log](https://vladmihalcea.com/how-does-a-relational-database-work/). If our system is suddenly affected by a system crash or a power outage, then all unfinished committed transactions may be replayed.

For messaging systems like [JMS](http://en.wikipedia.org/wiki/Java_Message_Service), transactions are not mandatory. That’s why we have non-transacted [acknowledgement modes](http://docs.oracle.com/javaee/6/api/javax/jms/Session.html).

File system operations are usually non-managed, but if your business requirements demand transaction file operations, you might make use a tool such as [XADisk](https://xadisk.java.net/).

While messaging and file systems use transactions optionally, for database management systems transactions are compulsory.

# Challanges

ACID is old school. [Jim Gray](http://research.microsoft.com/en-us/um/people/gray/papers/theTransactionConcept.pdf) described atomicity, consistency and durability long before I was even born. But that particular paper doesn’t mention anything about isolation. This is understandable if we think of the production systems of the late 70’s, which according to Jim Gray:

“At present, the largest airlines and banks have about 10,000 terminals and about 100 active transactions at any instant”.

So all efforts were spent on delivering correctness rather than concurrency. Things have changed drastically ever since, and nowadays even modest set-ups are able to run 1000 TPS.

From a database perspective, the atomicity is a fixed property, but everything else may be traded off for performance/scalability reasons.

If the database system is composed of multiple nodes, then distributed system consistency (C in [CAP Theorem](https://en.wikipedia.org/wiki/CAP_theorem) not C in ACID) mandates that all changes be propagated to all nodes ([multi-master replication](http://en.wikipedia.org/wiki/Multi-master_replication)). If slaves nodes are updated asynchronously then we break the consistency rule, the system becoming “[eventually consistent](http://en.wikipedia.org/wiki/Eventual_consistency)“.

Peter Bailis has a [very good article](http://www.bailis.org/blog/linearizability-versus-serializability/) explaining the difference between Consistency in CAP Theorem and Consistency in ACID.

A transaction is a data state transition, so the system must operate as if all transactions occur in a serial form even if those are concurrently executed.  
If there would be only one connection running at all times, then serializability wouldn’t impose any concurrency control cost. In reality, all transactional systems must accommodate concurrent requests, hence serialization has its toll on scalability. The [Amdahl’s law](http://en.wikipedia.org/wiki/Amdahl%27s_law) describes the relation between serial execution and concurrency:

“The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program.”

As you’ll see later, most database management systems choose (by default) to relax correctness guarantees to achieve better concurrency.

Playing with durability makes sense for [highly performing clustered databases](http://wiki.postgresql.org/images/3/3b/2011-11-11_PostgreSQL_SyncRepPerformance.pdf) if the enterprise system business requirements don’t mandate durable transactions. But, most often durability is better off untouched.

# Isolation Levels

Although some database management systems offer [MVCC](https://vladmihalcea.com/how-does-mvcc-multi-version-concurrency-control-work/), usually concurrency control is achieved through locking. But as we all know, locking increases the serializable portion of the executed code, affecting [parallelization](http://en.wikipedia.org/wiki/Amdahl%27s_law#Parallelization).

The SQL standard defines four Isolation levels:

* READ\_UNCOMMITTED
* READ\_COMMITTED
* REPEATABLE\_READ
* SERIALIZABLE

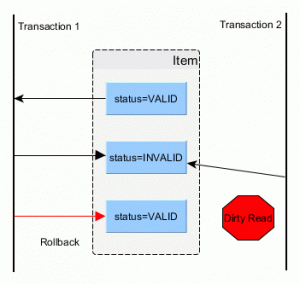
All but the SERIALIZABLE level are subject to data anomalies (phenomena) that might occur according to the following pattern:

|  |  |  |  |
| --- | --- | --- | --- |
| **Isolation Level** | **Dirty read** | **Non-repeatable read** | **Phantom read** |
| READ\_UNCOMMITTED | allowed | allowed | allowed |
| READ\_COMMITTED | prevented | allowed | allowed |
| REPEATABLE\_READ | prevented | prevented | allowed |
| SERIALIZABLE | prevented | prevented | prevented |

# Phenomena

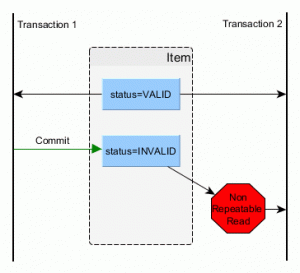
But what are all those phenomena we have just listed? Let’s discuss each and every one of them.

## Dirty read

[](https://vladmihalcea.files.wordpress.com/2014/01/acid-dirty-read.gif)

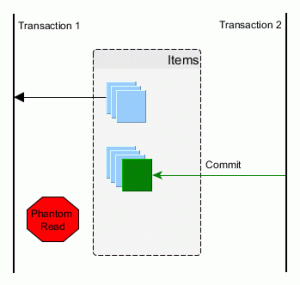
A dirty read happens when a transaction is allowed to read uncommitted changes of some other running transaction. This happens because there is no locking preventing it. In the picture above, you can see that the second transaction uses an inconsistent value as of the first transaction had rollbacked.

## Non-repeatable read

[](https://vladmihalcea.files.wordpress.com/2014/01/acid-non-repeatable-read.gif)

A non-repeatable read manifests when consecutive reads yield different results due to a concurring transaction that has just updated the record we’re reading. This is undesirable since we end up using stale data. This is prevented by holding a shared lock (read lock) on the read record for the whole duration of the current transaction.

## Phantom read

[](https://vladmihalcea.files.wordpress.com/2014/01/acid-phantom-read.gif)

A phantom read happens when a second transaction inserts a row that matches a previous select criteria of the first transaction. We, therefore, end up using stale data, which might affect our business operation. This is prevented using range locks or [predicate locking](https://vladmihalcea.com/how-does-database-pessimistic-locking-interact-with-insert-update-and-delete-sql-statements/).

## Even more ph

**54) Why the main method is static in java?**  
The method is static because otherwise there would be ambiguity: which constructor should be called? Especially if your class looks like this:

public class JavaClass

{

protected JavaClass(int x)

{ }

public void main(String[] args)

{

}

}

Should the JVM call new JavaClass(int)? What should it pass for x? If not, should the JVM instantiate JavaClass without running any constructor method? because that will special-case your entire class – sometimes you have an instance that hasn’t been initialized, and you have to check for it in every method that could be called. There are just too many edge cases and ambiguities for it to make sense for the JVM to have to instantiate a class before the entry point is called. That’s why main is static.

**What happens if you remove static modifier from the main method?**  
Program compiles successfully . But at runtime throws an error “NoSuchMethodError”.

**What is the** [**scope of variables**](https://www.geeksforgeeks.org/variable-scope-in-java/) **in Java in following cases?**

* **Member Variables** (Class Level Scope) : The member variables must be declared inside class (outside any function). They can be directly accessed anywhere in class
* **Local Variables** (Method Level Scope) : Variables declared inside a method have method level scope and can’t be accessed outside the method.
* **Loop Variables** (Block Scope) : A variable declared inside pair of brackets “{” and “}” in a method has scope withing the brackets only.

Read [more](https://www.geeksforgeeks.org/variable-scope-in-java/)

**What is** [**“this” keyword in java**](http://quiz.geeksforgeeks.org/this-reference-in-java/)**?**  
Within an instance method or a constructor, this is a reference to the current object — the object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using this.  
Usage of this keyword

* Used to refer current class instance variable.
* To invoke current class constructor.
* It can be passed as an argument in the method call.
* It can be passed as argument in the constructor call.
* Used to return the current class instance.
* Used to invoke current class method (implicitly)

**What is an** [**abstract class**](https://www.geeksforgeeks.org/abstract-classes-in-java/)**? How abstract classes are similar or different in Java from C++?**  
Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.

* Like C++, in Java, an instance of an abstract class cannot be created, we can have references of abstract class type though.
* Like C++, an abstract class can contain constructors in Java. And a constructor of abstract class is called when an instance of a inherited class is created
* In Java, we can have an abstract class without any abstract method. This allows us to create classes that cannot be instantiated, but can only be inherited.
* Abstract classes can also have final methods (methods that cannot be overridden). For example, the following program compiles and runs fine.

Read [more](https://www.geeksforgeeks.org/abstract-classes-in-java/)

**Which class is the superclass for every class ?**  
Object class

[**Can we overload main() method?**](https://www.geeksforgeeks.org/gfact-48-overloading-main-in-java/)  
The main method in Java is no extra-terrestrial method. Apart from the fact that main() is just like any other method & can be overloaded in a similar manner, JVM always looks for the method signature to launch the program.

* The normal main method acts as an entry point for the JVM to start the execution of program.
* We can overload the main method in Java. But the program doesn’t execute the overloaded main method when we run your program, we need to call the overloaded main method from the actual main method only.

Read [more](https://www.geeksforgeeks.org/gfact-48-overloading-main-in-java/)

**What is** [**object cloning**](http://quiz.geeksforgeeks.org/cloning-in-java/)**?**  
Object cloning means to create an exact copy of the original object. If a class needs to support cloning, it must implement java.lang.Cloneable interface and override clone() method from Object class. Syntax of the clone() method is :

protected Object clone() throws CloneNotSupportedException

If the object’s class doesn’t implement Cloneable interface then it throws an exception ‘CloneNotSupportedException’ .

Read [more](http://quiz.geeksforgeeks.org/cloning-in-java/)

**How is** [**inheritance in C++  different from Java?**](https://www.geeksforgeeks.org/comparison-of-inheritance-in-c-and-java/)

1. In Java, all classes inherit from the Object class directly or indirectly. Therefore, there is always a single inheritance tree of classes in Java, and Object class is root of the tree.
2. In Java, members of the grandparent class are not directly accessible. See [this G-Fact](https://www.geeksforgeeks.org/archives/15258) for more details.
3. The meaning of protected member access specifier is somewhat different in Java. In Java, protected members of a class “A” are accessible in other class “B” of same package, even if B doesn’t inherit from A (they both have to be in the same package).
4. Java uses *extends* keyword for inheritance. Unlike C++, Java doesn’t provide an inheritance specifier like public, protected or private. Therefore, we cannot change the protection level of members of base class in Java, if some data member is public or protected in base class then it remains public or protected in derived class. Like C++, private members of base class are not accessible in derived class.  
   Unlike C++, in Java, we don’t have to remember those rules of inheritance which are combination of base class access specifier and inheritance specifier.
5. In Java, methods are virtual by default. In C++, we explicitly use virtual keyword. See [this G-Fact](https://www.geeksforgeeks.org/archives/8876) for more details.
6. Java uses a separate keyword *interface* for interfaces, and *abstract* keyword for abstract classes and abstract functions.
7. Unlike C++, Java doesn’t support multiple inheritance. A class cannot inherit from more than one class. A class can implement multiple interfaces though.
8. In C++, default constructor of parent class is automatically called, but if we want to call parametrized constructor of a parent class, we must use [Initializer list](https://www.geeksforgeeks.org/archives/13797). Like C++, default constructor of the parent class is automatically called in Java, but if we want to call parameterized constructor then we must use super to call the parent constructor.

See examples [here](https://www.geeksforgeeks.org/comparison-of-inheritance-in-c-and-java/)

**Why method overloading is not possible by changing the return type in java?**  
In C++ and Java, functions can not be overloaded if they differ only in the return type . The return type of functions is not a part of the mangled name which is generated by the compiler for uniquely identifying each function. The No of arguments, Type of arguments & Sequence of arguments are the parameters which are used to generate the unique mangled name for each function. It is on the basis of these unique mangled names that compiler can understand which function to call even if the names are same(overloading).

**Can we override private methods in Java?**  
No, a private method cannot be overridden since it is not visible from any other class. Read [more](https://www.geeksforgeeks.org/can-override-private-methods-java/)

**What is** [**blank final variable**](https://www.geeksforgeeks.org/blank-final-in-java/)**?**  
A final variable in Java can be assigned a value only once, we can assign a value either in declaration or later.

final int i = 10;

i = 30; // Error because i is final.

A **blank final** variable in Java is a [final](https://www.geeksforgeeks.org/g-fact-48/) variable that is not initialized during declaration. Below is a simple example of blank final.

// A simple blank final example

final int i;

i = 30;

Read [more](https://www.geeksforgeeks.org/blank-final-in-java/)

**What is** [**“super” keyword in java**](http://quiz.geeksforgeeks.org/super-keyword/)**?**  
The super keyword in java is a reference variable that is used to refer parent class objects. The keyword “super” came into the picture with the concept of Inheritance. Whenever you create the instance of subclass, an instance of parent class is created implicitly i.e. referred by super reference variable.  
Various scenarios of using java super Keyword:

* super is used to refer immediate parent instance variable
* super is used to call parent class method
* super() is used to call immediate parent constructor

Read [more](http://quiz.geeksforgeeks.org/super-keyword/)

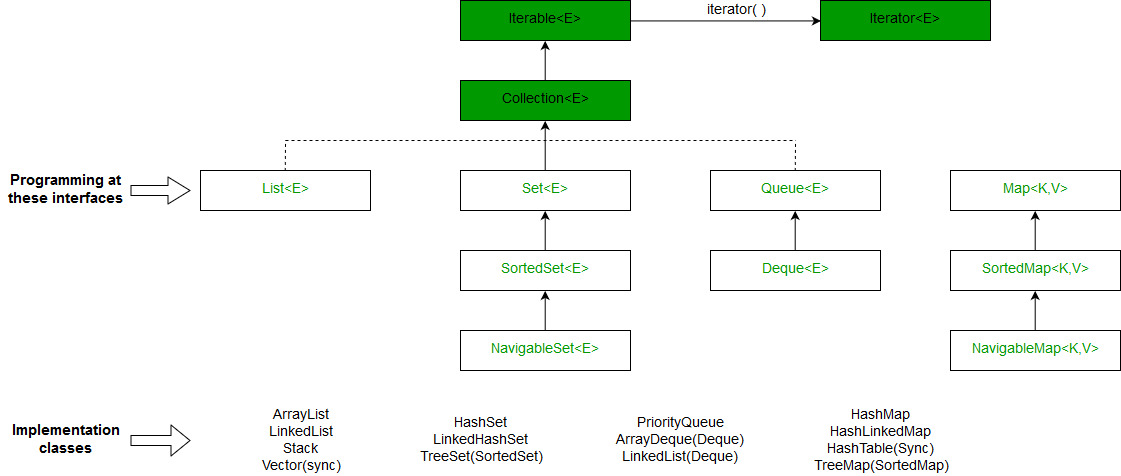
**What is** [**static variable in Java**](https://www.geeksforgeeks.org/static-class-in-java/)**?**  
The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

The static can be:

* variable (also known as class variable)
* method (also known as class method)
* block
* nested class

**Differences between** [**HashMap and HashTable in Java**](http://quiz.geeksforgeeks.org/differences-between-hashmap-and-hashtable-in-java/)**.**  
1. HashMap is non synchronized. It is not-thread safe and can’t be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.  
2. HashMap allows one null key and multiple null values whereas Hashtable doesn’t allow any null key or value.  
3. HashMap is generally preferred over HashTable if thread synchronization is not needed  
[Read more](http://quiz.geeksforgeeks.org/differences-between-hashmap-and-hashtable-in-java/)

**How are Java** [**objects stored in memory**](https://www.geeksforgeeks.org/g-fact-46/)**?**  
In Java, all objects are dynamically allocated on **Heap**. This is different from C++ where objects can be allocated memory either on Stack or on Heap. In C++, when we allocate abject using new(), the object is allocated on Heap, otherwise on Stack if not global or static.  
In Java, when we only declare a variable of a class type, only a reference is created (memory is not allocated for the object). To allocate memory to an object, we must use new(). So the object is always allocated memory on heap.  Read [more](https://www.geeksforgeeks.org/g-fact-46/)



#### 1. What do you mean by Platform independence of java?

You can write and compile program in one Operating system and run in other operating system.  
**For example:**  
You can compile program in Windows and can run it in Unix.

#### 2. What is difference between JVM, JRE and JDK ?

**JVM :** JVM stands for Java Virtual Machine. It is virtual machine which actually runs the byte code.

**JRE :** JRE stands for Java Runtime Environment. It provides runtime environment for java code. It has JVM , libraries such as rt.jar and other files.

**JDK :** JDK stands for Java development kit. It is superset of JRE, it has JRE + compilation and debugging tools(javac and java).

#### 3. What are memory areas allocated in JVM?

Memory areas allocated in JVM are:

* Heap area
* Method area
* JVM language stacks
* Program counter (PC) register
* Native method stacks

#### 4. What are some core concepts of OOPS in Java ?

Core concepts of OOPs are :

* Encapsulation
* Polymorphism
* Abstraction
* Inheritance

#### 5. What is Abstraction?

Abstraction is achieved using [interface](https://www.java2blog.com/2017/04/interface-in-java-with-example.html) and [abstract class](https://www.java2blog.com/2017/04/abstract-class-java.html) in Java.

You can read about [abstraction](https://www.java2blog.com/2017/04/abstraction-java-example.html) for more details..

#### 6. What is encapsulation?

You can refer about [encapsulation](https://www.java2blog.com/2017/05/encapsulation-java-example.html) for more details..

#### 7. What is Polymorphism in java?

You can refer about [Polymorphism](https://www.java2blog.com/2017/05/polymorphism-java-example.html) for more details..

#### 8. What is inheritance in java?

Inheritance allows to inherit properties and methods of parent class, so you can reuse all methods and properties.

#### 9. What is constructor in java?

Constructor can be considered a special code which is used to initiaze objects.  
It has two main points

* Class and Constuctor name should match
* Constructor should not have any return type else it will be same as method.

You can read more about [Constructor in Java](https://www.java2blog.com/2017/05/constructor-java.html).

#### 10. Can we declare constructor as final?

No, Constructor can not be declared as final. If you do so, you will get compile time error.

#### 11. What is immutable object in java?

Immutable object is object whose state can not be changed once created. You can take String object as example for immutable object.

#### 12. Why String is declared final or immutable in java?

There are various reasons to make String immutable.

* String pool
* Thread Safe
* Security
* Class Loading
* Cache hash value

You can refer why [String is immutable in java](https://www.java2blog.com/2016/05/why-string-is-immutable-in-java.html) for more details.

#### 13. What are access modifier available in java?

It Specifies accessibility of variables, methods , constructor of class.

There are four access modifier in java

**Private** : Accessible only to the class.

**Default :** Accessible in the package.

**Protected :** Accessible in the packages and its subclasses.

**Public :** Accessible everywhere

#### 14. What is difference between Abstract class and interface?

You can refer [difference between Abstract class and interface](https://www.java2blog.com/2014/06/difference-between-abstract-class-and.html) for more details.

#### 15. Can one interface implement another interface in java?

No, One interface can not implement another interface. It can extend it using extends keyword.

#### 16. What is marker interface?

Marker interfaces are interfaces which have no method but it is used to indicate JVM to behave specially when any class implement these interfaces.

**For example :**If you implement cloneable interface and then call .clone method of object, it will clone your object. If you do not implement cloneable interface, it will throw cloneNotSupported exception.

#### 17. What is method overloading and method overriding in java?

**Method overloading :** Method overloading is concept that allows a class to have same method name but diferent method arguments. Method overloading is also known as compile time polymorphism.  
 **Method overriding :** If child class contain same method as parent class with same method signature. This is called method overriding. Method overriding is also known as dynamic polymorphism.

#### 18. Can you override static methods in Java?

No, you can not override static methods in Java. You can create same method in child class but it won’t be dynamic polymorphism. It will be method hiding. Static methods belong at class level not at object level hence you can not override static method.

#### 19. Can you override private methods in Java?

No, you can not override private methods in Java. Private methods are not visible to subclass, hence you can not override private method but you can hide it.

#### 20. Difference between path and classpath in java?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Path** | **classpath** |
| Locate | It allows operating system to locate executable such as javac, java | It allows classloader to locate all .class file used by program |
| Overriding | You can not override path variable with java setting | You can override classpath by using -cp with java,javac or class-path in manifest file. |
| Inclusion | You need to include bin folder of jdk (**For example jdk1.7.1/bin**) | You need to include all the classes which is required by program |
| Used by | Operating system | java classloaders |

You can refer [difference between Path and ClassPath in java](https://www.java2blog.com/2016/05/difference-between-path-and-classpath-in-java.html) for more details.

#### 21. What is difference between StringBuffer and StringBuilder in java?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **StringBuffer** | **StringBuilder** |
| Thread-safe | StringBuffer is thread safe. Two threads can not call methods of StringBuffer simultaneously. | StringBuilder is not thread safe, so two threads can call methods of StringBuilder simultaneously. |
| Performance | It is less performance efficient as it is thread-safe | It is more performance efficient as it is not thread-safe. |

#### 22. What are methods you should override when you put an object as key in HashMap?

You need to implement hashcode() and equals() method if you put key as object in HashMap. You can go through [hashcode and equals method in java](https://www.java2blog.com/2014/02/hashcode-and-equals-method-in-java.html) for more details.

#### 23. Can you explain internal working of HashMap in java?

* There is an Entry[] array called table which has size 16.
* This table stores Entry class’s object. HashMap class has a inner class called Entry.This Entry have key value as instance variable.

Lets see structure of entry class Entry Structure.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | static class Entry implements Map.Entry  {          final K key;          V value;          Entry next;          final int hash;          ...//More code goes here  } |

Whenever we try to put any key value pair in hashmap, Entry class object is instantiated for key value and that object will be stored in above mentioned Entry[](table). Now you must be wondering, where will above created Entry object get stored(exact position in table). The answer is, hash code is calculated for a key by calling Hashcode() method. This hashcode is used to calculate index for above Entry[] table.  
You can read [How HashMap works internally in java](https://www.java2blog.com/2014/02/how-hashmap-works-in-java.html) for more details.

#### 24. Why java uses another hash function internally to calculate hash value apart from hashcode method which you have implemented?

It is due to avoid large number of collisions due to bad hashcode method written by developers.

You can refer [hash method of HashMap](https://www.java2blog.com/2014/02/hash-and-indexfor-method-in-hashmap.html) for more details.

#### 25. What if you don’t override hashcode method while putting custom objects as key in HashMap?

As we did not implement hashcode method, each object will have different hashcode(memory address) by default, so even if we have implemented equals method correctly, it won’t work as expected.

#### 26. Can you explain internal working of HashSet in java?

HashSet internally uses HashMap to store elements in HashSet. It uses PRESENT as dummy object as value in that HashMap. HashSet uses HashMap to check duplicates in the HashSet.

You can refer [How HashSet works internally in java](https://www.java2blog.com/2014/07/how-hashset-works-in-java.html) for more details

#### 27. What are differences between HashMap and HashSet in java?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **HashMap** | **HashSet** |
| Interface | This is core difference among them.HashMap implements Map interface | HashSet implement Set interface |
| Method for storing data | It stores data in a form of key->value pair.So it uses put(key,value) method for storing data | It uses add(value) method for storing data |
| Duplicates | HashMap allows duplicate value but not duplicate keys | HashSet does not allow duplicate values. |
| Performance | It is faster than hashset as values are stored with unique keys | It is slower than HashMap |
| HashCode Calculation | In hash map hashcode value is calculated using key object | In this,hashcode is calculated on the basis of value object. Hashcode can be same for two value object so we have to implement equals() method.If equals() method return false then two objects are different. |

#### 28. Can you explain internal working of ConcurrentHashMap in java?

ConcurrentHashMap uses concept of Segments to store elements. Each Segment logically contains an HashMap. ConcurrentHashMap does not lock whole object , it just lock part of it i.e. Segment.  
Structure of Segment:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | /\*\*       \* Segments are specialized versions of hash tables.  This       \* subclasses from ReentrantLock opportunistically, just to       \* simplify some locking and avoid separate construction.       \*/  static final class Segment extends ReentrantLock implements Serializable {  /\*\*        \* The per-segment table.  \*/          transient volatile HashEntry[] table;  // other methods and variables  } |

It stores a key value pair in a class called HashEntry which is similar to Entry class in HashMap. static final class HashEntry {         final K key;         final int hash;         volatile V value;         final HashEntry next; }

You can refer [internal working of ConcurrentHashMap in jav](https://www.java2blog.com/2014/12/concurrenthashmap-in-java.html)a for more details

#### 29. Do we have lock while getting value from ConcurrentHashMap?

There is no lock while getting values from ConcurrentHashMap.Segments are only for write operation.In case of read operation, it allows full concurrency and provides most recently updated value using volatile variables.

#### 30. How do you sort Collection of custom objects in java?

We need to implement the comparable interface to custom object class(Let’s say Country) and then implement compareTo(Object o) method which will be used for sorting. It will provides default way of sorting custom objects.  
If we want to sort custom object (Lets say country) on different attributes such as name, population etc.We can implement Comparator interface and can be used for sorting.  
For more details, you can go through following links:

[Comparable in java](https://www.java2blog.com/2014/06/comparable-in-java.html)  
[Comparator in java](https://www.java2blog.com/2014/06/comparator-in-java.html)

#### 31. What are differences between ArrayList and LinkedList in java?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **ArrayList** | **LinkedList** |
| Internal data structure | It uses dynamic array to store elements internally | It uses doubly Linked List to store elements internally |
| Manipulation | If  We need to insert or delete element in ArrayList, it may take O(n), as it internally uses array and we may have to shift elements in case of insertion or deletion | If  We need to insert or delete element in LinkedList, it will take O(1), as it internally uses doubly LinkedList |
| Search | Search is faster in ArrayList as uses array internally which is index based. So here time complexity is O(1) | Search is slower in LinkedList as uses doubly Linked List internally So here time complexity is O(n) |
| Interfaces | ArrayList implements List interface only, So it can be used as List only | LinkedList implements List,Deque interfaces, so it can be used as List,Stack or Queue |

You can refer [difference between ArrayList and LinkedList in java](https://www.java2blog.com/2015/06/difference-between-arraylist-and.html) for more details.

#### 32. What is Enum in java?

Java Enum is special data type which represents list of constants values. It is a special type of java class. It can contain constant, methods and constructors etc.  
You can refer [Enum in java](https://www.java2blog.com/2015/09/java-enum.html) for more details.

#### 33. How do you create custom exception in java?

You just need to extend Exception class to create custom exception. If you want to create Unchecked exception, then you need extend Runtime Exception.

You can refer to [create custom exception in Java](https://www.java2blog.com/2016/07/how-to-create-custom-exception-in-java.html).

#### 34.What is difference between Checked Exception and Unchecked Exception?

**Checked Exception:**Checked exceptions are those exceptions which are checked at compile. If you do not handle them , you will get compilation error.

**For example:** IOException

**Unchecked Exception :**Unchecked exceptions are those exceptions which are not checked at compile time. Java won’t complain if you do not handle the exception.

**For example:** NullPointerException, ArrayIndexOutOfBoundsException

You can refer [difference between checked exception and unchecked exception](https://www.java2blog.com/2016/07/difference-between-checked-and-unchecked-exception-in-java.html) for more details.

#### 35. Can we have try without catch block in java ?

Yes, we can have try without catch block by using finally block. You can use try with finally. As you know finally block always executes even if you have exception or return statement in try block except in case of System.exit().  
You can refer [Try with finally block](https://www.java2blog.com/2016/06/can-we-have-try-without-catch-block-in-java.html) for more details.

#### 36. What are ways to create a thread in java ?

There are two ways to create a thread in java

* By extending thread class
* By implementing Runnable interface.

#### 37. What are differences between Sleep and wait in java?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **wait** | **sleep** |
| Synchonized | wait should be called from synchronized context i.e. from block or method, If you do not call it using synchronized context, it will throw IllegalMonitorStateException | It need not be called from synchronized block or methods |
| Calls on | wait method operates on Object and defined in Object class | Sleep method operates on current thread and is in java.lang.Thread |
| Release of lock | wait release lock of object on which it is called and also other locks if it holds any | Sleep method does not release lock at all |
| Wake up condition | until call notify() or notifyAll() from Object class | Until time expires or calls interrupt() |
| static | wait is non static method | sleep is static method |

You can refer [difference between sleep and wait in java](https://www.java2blog.com/2014/07/difference-between-sleep-and-wait-in.html) for more details.

#### 38. Define states of thread in java?

There are 5 states of thread in java

**New** : When you create a thread object and it is not alive yet.

**Runnable:**When you call start method of thread, it goes into Runnable state. Whether it will execute immediately or execute after some times , depends on thread scheduler.

**Running :** When thread is being executed, it goes to running state.

**Blocked :** When thread waits for some resources or some other thread to complete (due to thread’s join), it goes to blocked state.

**Dead:** When thread’s run method returns, thread goes to dead state.

#### 39. Can we call run method directly to start a thread?

No, you can not directly call run method to start a thread. You need to call start method to create a new thread. If you call run method directly , it won’t create a new thread and it will be in same stack as main.  
You can refer [can we call run method directly to start a thread](https://www.java2blog.com/2016/05/can-we-call-run-method-directly-to-start-thread.html) for more details

#### 40. Can we start a thread twice in java?

No, Once you have started a thread, it can not be started again. If you try to start thread again , it will throw IllegalThreadStateException.  
You can refer [can we start thread twice](https://www.java2blog.com/2016/05/can-we-start-thread-twice-in-java.html) for more details

#### 41. What is CountDownLatch in java?

As per java docs, CountDownLatch is synchronisation aid that allow one or more threads to wait until set of operations being performed in other threads completes. So in other words, CountDownLatch waits for other threads to complete set of operations.  
CountDownLatch is initialized with count. Any thread generally main threads calls latch.awaits() method, so it will wait for either count becomes zero or it’s interrupted by another thread and all other thread need to call latch.countDown() once they complete some operation.

So count is reduced by 1 whenever latch.countDown() method get called, so if count is n that means count can be used as n threads have to complete some action or some action have to be completed n times.  
You can refer [CountDownLatch in java with example](https://www.java2blog.com/2015/08/countdownlatch-in-java.html) for more details.

#### 42. What is difference between CountDownLatch and CyclicBarrier?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **CountDownLatch** | **CyclicBarrier** |
| Reuse | It can not be reused once count reaches 0 | It can be reinitialized once parties reaches to 0, so it can reused |
| Method | It calls countDown() method to reduce the counter | It calls await() method to reduce the counter. |
| Common Event | It can not trigger common event when count reaches 0 | It can trigger common event (Runnable) once reaches to a barrier point. **Constructor** :CyclicBarrier(int parties, Runnable barrierAction) |
| Constructor | CountDownLatch(int count) | CyclicBarrier(int parties) |

#### 43. Why wait, notify and nofiyAll method belong to object class ?

In java, we put locks on shared objects not on thread, so these methods are present in Object class. As every object have mutex(lock), it make sense to put these methods in object class.

#### 44. Can you call wait, notify and notifyAll from non synchronized context?

No, you can not call wait, notify and notifyAll from non synchronized context. If you do so, it will throw IllegalMonitorStateException.

#### 45. What is the difference between creating String as new() and literal?

If you create a String using new operator, it is not interned. It will  create new object in heap memory even if String object already exists with same content.

|  |  |
| --- | --- |
| 1  2  3  4  5 | String str1=new String("hello");          String str2=new String("hello");          System.out.println(str1==str2); |

It will return false as str1 and str2 will point to different object

If you create a String using assignment operator, it goes to the String constant pool and it is interned. If you create another String with same content, both will reference to same object in String constant pool.

|  |  |
| --- | --- |
| 1  2  3  4  5 | String str1="helloworld";  String str2="helloworld";  System.out.println(str1==str2); |

It will return true as str1 and str2 will point to the same object in String constant pool.

#### 46. What is Covariant return type in java?

Covariant return type means if subclass overrides any method, return type of this overriding method can be subclass of return type of base class method.  
**For example:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | package org.arpit.java2blog;    public class BaseClass {    public A m1() {    System.out.println("In BaseClass method");    return new A();  }    public static void main(String args[])  {    BaseClass b=new SubClass();    b.m1();  }  }    class SubClass extends BaseClass {  public B m1() {    System.out.println("In SubClass method");    return new B();    }  }    class A {  }    class B extends A {    } |

Above example is perfect example of covariant return type.

#### 47. What is garbage Collection?

Garbage Collection is a process of looking at heap memory and deleting unused object present in heap memory. Garbage Collection frees unused memory. Garbage Collection is done by JVM.

#### 48. What is System.gc()?

This method is used to invoke garbage collection for clean up unreachable object but it is not guaranteed that when you invoke System.gc(), garbage collection will definitely trigger.

#### 49. What is use of finalize() method in object class?

Finalize method get called when object is being collected by Garbage Collector. This method can be used to write clean code before object is collected by Garbage Collector.

#### 50.What is difference between final, finally and finalize in Java?

**final :** Final is a keyword which is used with class to avoid being extended, with instance variable so they can not reassigned, with methods so that they can not be overridden.  
**finally :** Finally is a keyword used with try, catch and finally blocks. Finally block executes even if there is an exception. It is generally used to do some clean up work.  
**finalize :**  Finalize is a method is used to invoke garbage collection for clean up unreachable object but it is not guaranteed that when you invoke System.gc(), garbage collection will definitely trigger.

## 1. Why String is declared final or immutable in java?

There are various reasons to make String immutable.

* String pool
* Thread Safe
* Security
* Class Loading
* Cache hash value

You can refer why [String is immutable in java](https://www.java2blog.com/2016/05/why-string-is-immutable-in-java.html) for more details.

## 2. How to reverse a String in java? Can you write a program without using any java inbuilt methods?

There are many ways to do it, some of them are:

* Using for loop
* Using recursion
* Using StringBuffer

Please refer to the solution at [reverse a String in java](https://www.java2blog.com/2015/08/java-program-to-reverse-string.html)

## 3. How to check if two Strings are anagram in java?

Anagrams means if two String have same characters but in different order. For example: Angel and Angel are anagrams  
There are many ways to check if Strings are anagrams. Some of them are:

1. Using String methods
2. Using array.sort

Please refer to solution at [check if two Strings are anagram in java.](https://www.java2blog.com/2016/05/check-if-two-strings-are-anagrams-in-java-example-program.html)

## 4. How to check if String has all unique characters in java?

There are multiple ways to find if String has all unique characters or not.

* By using HashSet
* Using indexOf and lastIndexOf methods of String
* By Using ascii value of characters.

Please refer to complete solution at [check if String has all unique characters](https://www.java2blog.com/2016/06/check-if-string-has-all-unique-characters-in-java.html).

## 5. How to check if one String is rotation of another String in java?

Lets say you need to check whether str1 and str2 is rotation of one another or not.

1. Create a new String with str3= str1 + str1
2. Check if str3 [contains](https://www.java2blog.com/2016/05/java-string-contains-example.html) str2 or not.
3. if str3 [contains](https://www.java2blog.com/2016/05/java-string-contains-example.html) str2 then str2 is rotation of str1 else it is not

You can find complete solution at [check if one String is rotation of another in java](https://www.java2blog.com/2016/05/check-if-one-string-is-rotation-of-another-java.html).

## 6. Write a java program to find duplicate characters in String in java?

1. Create a [HashMap](https://www.java2blog.com/2016/04/hashmap-in-java-with-examples.html) and character of String will be inserted as key and its count as value.
2. If [Hashamap](https://www.java2blog.com/2016/04/hashmap-in-java-with-examples.html) already contains char,increase its count by 1, else put char in HashMap.
3. If value of Char is more than 1, that means it is duplicate character in that String.

Please refer to solution at [program to find duplicate characters in a String](https://www.java2blog.com/2016/05/find-duplicate-characters-in-string-java.html).

## 7. Find first non repeated character in String in java?

There are may ways to find it.  
Some of them are:

* Using LinkedHashMap
* Using indexOf and lastIndexOf methods.

Please find complete solution at [find first non repeated character in  a String](https://www.java2blog.com/2015/08/find-first-non-repeated-character-in.html).

## 8. Find all substrings of String in java?

Java program to find all substrings of a String.  
For example: If input is “abb”  then output should be “a”, “b”,”b”, “ab”, “bb”, “abb”

We will use String class’s subString method to find all subString.  
Please refer to complete solution at [find all subStrings of String.](https://www.java2blog.com/2015/08/find-all-substrings-of-string-in-java.html)

## 9. Find length of String without using any inbuilt method in java?

You can use try catch block for catching StringIndexOutOfBoundException and when this exception aries, you can simply return i(Index at which you will get the exception)  
Please refer to complete solution at [find length of String without inbuilt methods](https://www.java2blog.com/2015/08/find-length-of-string-without-using.html).

## 10. Write a java program to print all permutations of String in java?

Take out first character of String and insert into different places of permutations of remaining String recursively. Please find complete solution at [how to find all permutations of String in java](https://www.java2blog.com/2016/07/find-all-permutations-of-string-in-java.html).

## 11. What is the difference between creating String as new() and literal?

If you create a String using new operator, it is not interned. It will  create new object in heap memory even if String object already exists with same content.

|  |  |
| --- | --- |
| 1  2  3  4  5 | String str1=new String("Java2Blog");          String str2=new String("Java2Blog");          System.out.println(str1==str2); |

It will return false as str1 and str2 will point to different object

If you create a String using assignment operator, it goes to the String constant pool and it is interned. If you create another String with same content, both will reference to same object in String constant pool.

|  |  |
| --- | --- |
| 1  2  3  4  5 | String str1="Java2Blog";  String str2="Java2Blog";  System.out.println(str1==str2); |

It will return true as str1 and str2 will point to same object in String constant pool.

## 12. How many objects will be created in below code?

|  |  |
| --- | --- |
| 1  2  3  4 | String str1= "java2blog";  String str2= "java2blog"; |

Only one object will be created and will be stored in String constant pool.

## 13. How do you convert String to char array in java?

You can use string’s [toCharArray()](https://www.java2blog.com/2016/05/how-to-convert-string-to-char-array-in.html) method to convert String to char array.

## 14. What is difference between StringBuffer and StringBuilder in java?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **StringBuffer** | **StringBuilder** |
| Thread-safe | StringBuffer is thread safe. Two threads can not call methods of StringBuffer simultaneously. | StringBuilder is not thread safe, so two threads can call methods of StringBuilder simultaneously. |
| Performance | It is less performance efficient as it is thread-safe | It is more performance efficient as it is not thread-safe. |

## 15. How many objects will be created in below code?

|  |  |
| --- | --- |
| 1  2  3  4 | String str1= new String("java2blog");  String str2= new String("java2blog"); |

Three objects will be created here, two in heap memory and one in String constant pool.