1. **What is difference between JDK, JRE and JVM?**

JVM: JVM is an acronym for Java Virtual Machine, it is an abstract machine which provides the runtime environment in which java bytecode can be executed. It is a specification.

JRE: JRE stands for Java Runtime Environment. It is the implementation of JVM.

JDK: JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.

1. **How many types of memory areas are allocated by JVM?**

Class (Method) Area: Class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

Heap: It is the runtime data area in which objects are allocated.

Stack: Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

Program Counter Register: It contains all the native methods used in the application.

Native Method Stack: It contains:

1) Interpreter: Read bytecode stream then execute the instructions.

2) Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term ?compiler? Refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

1. **Local variable vs instance variable vs class variable.**

**Local variable** A variable that is declared inside the method is called local variable.

These variables do not have default values.

**Instance variable (Non-static Fields)** A variable that is declared inside the class but outside the method is called instance variable. They are non-static fields.

They do have default values.

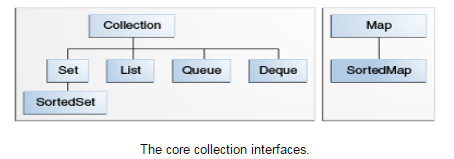
**Class variable (Static Fields)** A variable that is declared as static is called static variable. It cannot be local.

**Note:-** All member variable (instance variable) have to load into **heap** so they have to initialized with default values when an instance of class is created. In case of local variables, they don't get loaded into heap they are stored in **stack** until they are being used.

**Collection Interview Questions**

[**http://javahungry.blogspot.com/2015/05/50-java-collections-interview-questions-and-answers.html**](http://javahungry.blogspot.com/2015/05/50-java-collections-interview-questions-and-answers.html)

1. What is the Collection framework?



**Collection** — the root of the collection hierarchy. A collection represents a group of objects known as its elements.

**Set —** a collection that cannot contain duplicate elements. The Java platform contains three general-purpose Set implementations:

* **HashSet -** which stores its elements in a hash table, is the best-performing implementation; however it makes no guarantees concerning the order of iteration.
  + uses hashtable to store the elements.It extends AbstractSet class and implements Set interface.
  + contains unique elements only.
* **TreeSet -** which stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashSet.
  + contains unique elements only like HashSet. The TreeSet class implements NavigableSet interface that extends the SortedSet interface.
  + maintains ascending order.
* **LinkedHashSet -** which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set (insertion-order).
  + contains unique elements only like HashSet. It extends HashSet class and implements Set interface.
  + maintains insertion order.

**List —** an ordered collection (sometimes called a sequence). Lists can contain duplicate elements. The Java platform contains two general-purpose List implementations. **ArrayList**, which is usually the better-performing implementation, and **LinkedList** which offers better performance under certain circumstances.

* **ArrayList**
  + Java ArrayList class uses a dynamic array for storing the elements. It extends AbstractList class and implements List interface.
  + Java ArrayList class can contain duplicate elements.
  + Java ArrayList class maintains insertion order.
  + Java ArrayList class is non synchronized.
  + Java ArrayList allows random access because array works at the index basis.
  + In Java ArrayList class, manipulation is slow because a lot of shifting needs to be occurred if any element is removed from the array list.
* **LinkedList**
  + Java LinkedList class uses doubly linked list to store the elements. It extends the AbstractList class and implements List.
  + Java LinkedList class can contain duplicate elements.
  + Java LinkedList class maintains insertion order.
  + Java LinkedList class is non synchronized.
  + In Java LinkedList class, manipulation is fast because no shifting needs to be occurred.

**Queue —** a collection used to hold multiple elements prior to processing. Besides basic Collection. operations, a Queue provides additional insertion, extraction, and inspection operations. Queues typically, but do not necessarily, order elements in a FIFO (first-in, first-out) manner. In a FIFO queue, all new elements are inserted at the tail of the queue and the head of the queue is the element that would be removed.

**Deque —** a collection used to hold multiple elements prior to processing. Besides basic Collection operations, a Deque provides additional insertion, extraction, and inspection operations. Deques can be used both as FIFO (first-in, first-out) and LIFO (last-in, first-out). In a deque all new elements can be inserted, retrieved and removed at both ends.

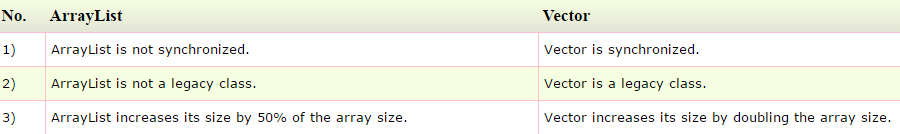
**Map —** an object that maps keys to values. A Map cannot contain duplicate keys; each key can map to at most one value.

* **HashMap**
  + A HashMap contains values based on the key. It implements the Map interface and extends AbstractMap class.
  + It contains only unique elements.
  + It may have one null key and multiple null values.
* **LinkedHashMap**
  + A LinkedHashMap contains values based on the key. It implements the Map interface and extends HashMap class.
  + It contains only unique elements.
  + It may have one null key and multiple null values.
  + It is same as HashMap instead maintains insertion order.
* **TreeMap**
  + A TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
  + It contains only unique elements.
  + It cannot have null key but can have multiple null values.
  + It is same as HashMap instead maintains ascending order.

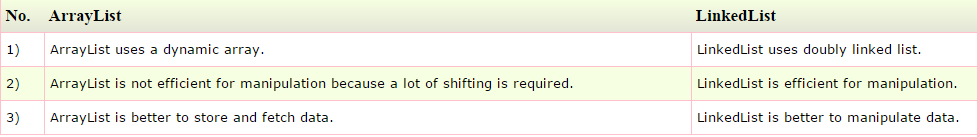
**SortedSet —** a Set that maintains its elements in ascending order.

**SortedMap —** a Map that maintains its mappings in ascending key order.

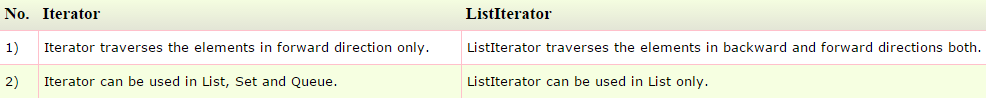
1. What is the difference between ArrayList and Vector?



1. What is the difference between ArrayList and LinkedList?



1. What is the difference between Iterator and ListIterator?



1. What is the difference between Iterator and Enumeration?

The main difference between Iterator and Enumeration is that Iterator has remove() method while Enumeration doesn't.  
Hence, using Iterator we can manipulate objects by adding and removing the objects from the collections. Enumeration behaves like a read only interface as it can only traverse the objects and fetch it.

1. What is the difference between List and Set?

List can contain duplicate elements whereas Set contains only unique elements.

1. What is the difference between HashSet and TreeSet?

HashSet maintains **no order** whereas TreeSet maintains **ascending order**.

1. What is the difference between Set and Map?

Set contains values only whereas Map contains key and values both.

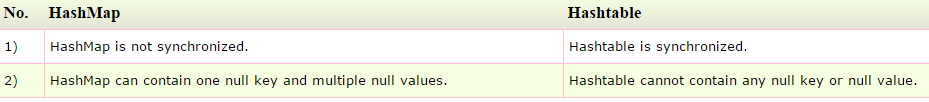
1. What is the difference between HashSet and HashMap?

HashSet contains only values whereas HashMap contains entry(key,value). HashSet can be iterated but HashMap need to convert into Set to be iterated.

1. What is the difference between HashMap and TreeMap?

HashMap maintains **no order** but TreeMap maintains **ascending order**.

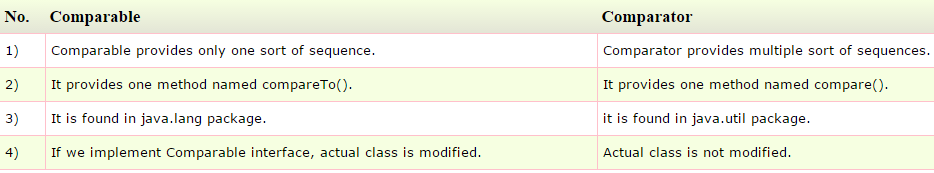
1. What is the difference between HashMap and Hashtable?



1. What is the difference between Collection and Collections?

Collection is an interface whereas Collections is a class. Collection interface provides normal functionality of data structure to List, Set and Queue. But, Collections class is to sort and synchronize collection elements.

1. What is the difference between Comparable and Comparator?



1. What is the advantage of Properties file?

If you change the value in properties file, you don't need to recompile the java class. So, it makes the application easy to manage.

1. What does the hashCode() method?

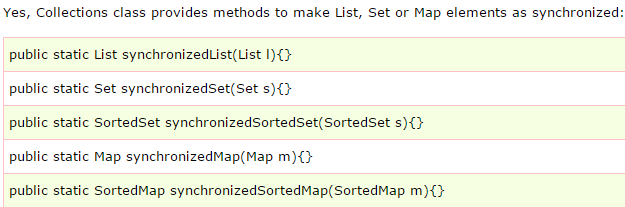
The hashCode() method returns a hash code value (an integer number). The hashCode() method returns the same integer number, if two keys (by calling equals() method) are same.

1. Why we override equals() method?

The equals method is used to check whether two objects are same or not. It needs to be overridden if we want to check the objects based on property.

For example, Employee is a class that has 3 data members: id, name and salary. But, we want to check the equality of employee object on the basis of salary. Then, we need to override the equals() method.

1. How to synchronize List, Set and Map elements?



1. What is the advantage of generic collection?

If we use generic class, we don't need typecasting. It is typesafe and checked at compile time.

1. What is hash-collision in Hashtable and how it is handled in Java?

Two different keys with the same hash value is known as hash-collision. Two different entries will be kept in a single hash bucket to avoid the collision.

1. What is the Dictionary class?

The Dictionary class provides the capability to store key-value pairs.

1. What is the default size of load factor in hashing based collection?

The default size of load factor is **0.75**. The default capacity is computed as initial capacity \* load factor. For example, 16 \* 0.75 = 12. So, 12 is the default capacity of Map.

1. How do you remove elements during Iteration?

Iterator also has a method **remove()** when remove is called, the current element in the iteration is deleted.

1. What are the advantages of ArrayList over arrays?

Some of the advantages ArrayList has over arrays are:

It can grow dynamically

It provides more powerful insertion and search mechanisms than arrays.

Array is static in size while ArrayList is dynamic in size.  
Array can contain primitive data types while ArrayList can not contain primitive data types.

1. How to obtain Array from an ArrayList?

ArrayList<String> myArrayList = new ArrayList<String>();

String[] myArray = myArrayList.toArray(new String[0]);

1. Why insertion and deletion in ArrayList is slow compared to LinkedList ?

* ArrayList internally uses and array to store the elements, when that array gets filled by inserting elements a new array of roughly 1.5 times the size of the original array is created and all the data of old array is copied to new array.
* During deletion, all elements present in the array after the deleted elements have to be moved one step back to fill the space created by deletion. In linked list data is stored in nodes that have reference to the previous node and the next node so adding element is simple as creating the node an updating the next pointer on the last node and the previous pointer on the new node. Deletion in linked list is fast because it involves only updating the next pointer in the node before the deleted node and updating the previous pointer in the node after the deleted node.

1. Why are Iterators returned by ArrayList called Fail Fast ?

Because, if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

1. How do you decide when to use ArrayList and When to use LinkedList?

If you need to support random access, without inserting or removing elements from any place other than the end, then ArrayList offers the optimal collection. If, however, you need to frequently add and remove elements from the middle of the list and only access the list elements sequentially, then LinkedList offers the better implementation.

1. How do you decide when to use HashMap and when to use TreeMap?

For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then TreeMap is your better alternative. Depending upon the size of your collection, it may be faster to add elements to a HashMap, then convert the map to a TreeMap for sorted key traversal.

1. **How do you sort an ArrayList (or any list) of user-defined objects?**

Create an implementation of the *java.lang.Comparable* interface that knows how to order your objects and pass it to *java.util.Collections.sort*(List, Comparator).

1. What is the Comparable interface?

The Comparable interface is used to sort collections.

**int i = object1.compareTo(object2);**

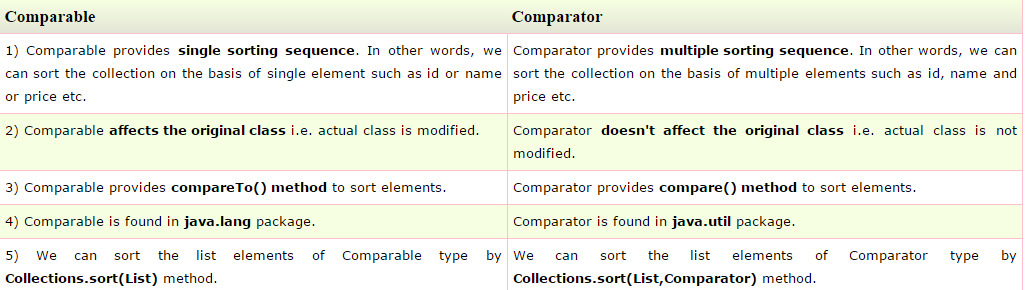
If object1 < object2: The value of i returned will be negative.

If object1 > object2: The value of i returned will be positive.

If object1 = object2: The value of i returned will be zero.

1. **Difference between Comparable and Comparator?**

Comparable and Comparator both are interfaces and can be used to sort collection elements.



*6) int objectOne.compareTo(objectTwo) int compare(ObjOne, ObjTwo)*

1. **What is the root interface in collection hierarchy?**

Root interface in collection hierarchy is **Collection interface.** Few may argue that Collection interface extends **Iterable interface**. So iterable should be the root interface. But iterable interface present in **java.lang** package not in **java.util** package. So collection interface is the root interface from java.util package.

1. **What is the difference between Collection and Collections?**

Collection is an interface while Collections is a java class, both are present in java.util package.

1. **Which collection classes are synchronized or thread-safe?**

Stack, Properties, Vector and Hashtable can be used in multi threaded environment because they are synchronized classes (or thread-safe).

1. What is an iterator?

Iterator is an interface. It is found in java.util package. It provides methods to iterate over any Collection.

1. Which design pattern followed by Iterator?

It follows iterator design pattern. Iterator design pattern provides us to navigate through the collection of objects by using a common interface without letting us know about the underlying implementation.

Enumeration is an example of Iterator design pattern.

1. Which methods you need to override to use any object as key in HashMap?

To use any object as key in HashMap, it needs to implement equals() and hashCode() method.

1. What is the difference between Queue and Stack?

**Queue** is a data structure which is based on FIFO (first in first out ) property . An example of Queue in real world is buying movie tickets in the multiplex or cinema theaters.  
  
**Stack** is a data structure which is based on LIFO (last in first out) property. An example of Stack in real world is insertion or removal of CD from the CD case.

1. How to reverse the List in Collections?

There is a built in reverse method in Collections class. reverse(List list) accepts list as parameter.

**Collections.reverse(listobject);**

1. How to convert the array of strings into the list?

Arrays class of java.util package contains the method asList() which accepts the array as parameter.

So,

String[] wordArray = {"Love Yourself" , "Alive is Awesome" , "Be in present"};

List wordList = Arrays.asList(wordArray);

1. What is the difference between HashMap and ConcurrentHashMap?

Main differences between HashMap and ConcurrentHashMap are :  
a. HashMap is not synchronized while ConcurrentHashMap is synchronized.  
b. HashMap can have one null key and any number of null values while ConcurrentHashMap does not allow null keys and null values.

<http://javahungry.blogspot.com/2014/02/hashmap-vs-concurrenthashmap-java-collections-interview-question.html>

1. How HashMap works in Java?

HashMap works on the principle of Hashing.

1. How HashSet works internally in java?

HashSet internally uses HashMap to maintain the uniqueness of elements.

1. Why Map interface does not extend the Collection interface in Java Collections Framework?

Map interface is not compatible with the Collection interface. Since Map requires key as well as value , for example , if we want to add key-value pair then we will use put(Object key , Object value) . So there are two parameters required to add element to the HashMap object  . In Collection interface add(Object o) has only one parameter.

1. What is CopyOnWriteArrayList? How it is different from ArrayList in Java?

CopyOnWriteArrayList is a thread safe variant of ArrayList. It guaranteed not to throw ConcurrentModificationException.

1. What are concurrentCollectionClasses?

In jdk1.5 , Java Api developers had introduced new package called java.util.concurrent that have thread-safe collection classes as they allow collections to be modified while iterating . The iterator is fail-safe that is it will not throw ConcurrentModificationException.  
Some examples of concurrentCollectionClasses are :  
a. CopyOnWriteArrayList  
b. ConcurrentHashMap

1. How will you make Collections readOnly?

We can make the Collection readOnly by using the following lines code:

General : Collections.unmodifiableCollection(Collection c)

Collections.unmodifiableMap(Map m)

Collections.unmodifiableList(List l)

Collections.unmodifiableSet(Set s)

1. What is UnsupportedOperationException?

This exception is thrown to indicate that the requested operation is not supported.  
Example of UnsupportedOperationException:  
In other words, if you call add() or remove() method on the readOnly collection . We know readOnly collection can not be modified . Hence , UnsupportedOperationException will be thrown.

1. How do you use a custom object as key in Collection  classes like HashMap?

If one is using the custom object as key then one needs to override equals() and hashCode() method.

1. Explain the importance of hashCode() and equals() method ? Explain the contract also?

HashMap object uses Key object hashCode() method and equals() method to find out the index to put the key-value pair. If we want to get value from the HashMap same both methods are used . Somehow, if both methods are not implemented correctly , it will result in two keys producing the same hashCode() and equals() output.

Contract of hashCode() and equals() method  
  
a.  If  object1.equals(object2) , then  object1.hashCode() == object2.hashCode() should always be true.  
  
b. If object1.hashCode() == object2.hashCode() is true does not guarantee object1.equals(object2)

1. What is hash-collision in Hashtable? How it was handled in Java?

In Hashtable , if two different keys have the same hash value then it lead to hash -collision. A bucket of type linkedlist used to hold the different keys of same hash value.

**Thread Interview Questions**

1. What is hash-collision in Hashtable? How it was handled in Java?

* The thread is an independent path of execution.
* Threads consumes CPU in best possible manner, hence enables multiprocessing. Multi threading reduces idle time of CPU which improves performance of application.
* Thread are light weight process.
* We can create multiple threads in java, even if we don’t create any Thread, one Thread at least do exist i.e. main thread.
* Multiple threads run parallely in java.
* Threads have their own stack.

1. What is advantage of threads?

Suppose one thread needs 10 minutes to get certain task, 10 threads used at a time could complete that task in 1 minute, because threads can run in parallel.

1. What is difference between Process and Thread in java?

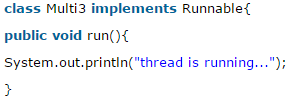
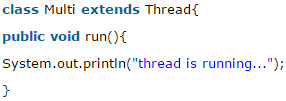
* Both process and Thread are independent path of execution but one process can have multiple Threads.
* Every process has its own memory space, executable code and a unique process identifier (PID) while every thread has its own stack in Java but it uses process main memory and share it with other threads.
* Threads are also refereed as task or light weight process (LWP) in operating system
* Threads from same process can communicate with each other by using Programming language construct like wait() and notify() in Java.

1. How to implement Threads in java?

Threads can be created in two ways

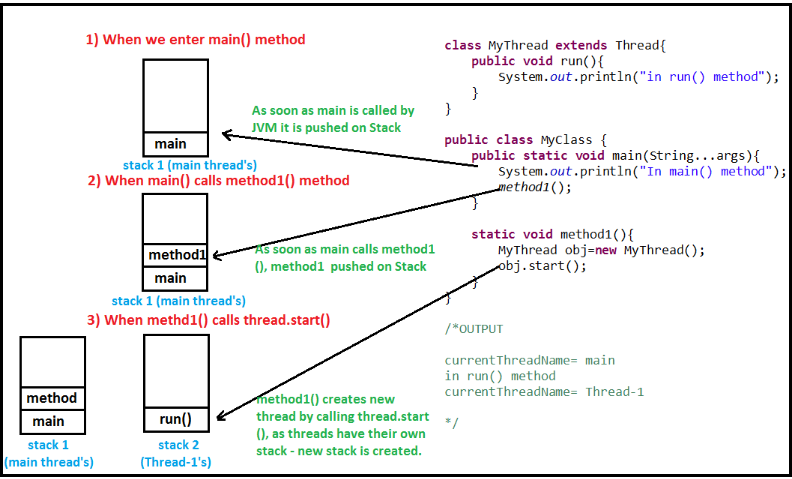
* By implementing java.lang.Runnable interface
* Extending java.lang.Thread class

and then extending run method.

1. Does Thread implements their own Stack, if yes how? (Important)

Yes, Threads have their own stack.



1. We should implement Runnable interface or extend Thread class. What are differences between implementing Runnable and extending Thread?

* Java programming language doesn't support multiple inheritances of class, but it allows you to implement multiple interfaces. Which means, it's better to implement **Runnable** then extends **Thread** if you also want to extend another class.
* Thread safety: When we implement Runnable interface, same object is shared amongst multiple threads, but when we extend Thread class each and every thread gets associated with new object.
* Inheritance (Implementing Runnable is lightweight operation) : When we extend Thread unnecessary all Thread class features are inherited, but when we implement Runnable interface no extra feature are inherited, as Runnable only consists only of one abstract method i.e. run() method. So, implementing Runnable is lightweight operation.

1. When threads are not lightweight process in java?

Threads are lightweight process only if threads of same process are executing concurrently. But if threads of different processes are executing concurrently then threads are heavy weight process.

1. How can you ensure all threads that started from main must end in order in which they started and also main should end in last? (Important)?

We can use join() method to ensure all threads that started from main must end in order in which they started and also main should end in last.



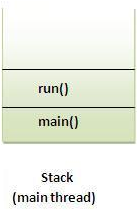
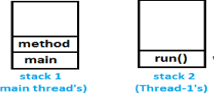
If we note output, all threads ended in order in which they were called and main thread has ended last.

First, main thread was called, it started Thread1 and then we called join() method on Thread1, once Thread1 ended main thread started Thread2 and we called join() method on Thread2, once Thread2 ended main thread also ended.

1. What is the difference between start() and run() method of Thread class?

When you call start() method, main thread internally calls run() method to start newly created Thread. Each thread starts in a separate call stack.

Invoking the run() method from main thread, the run() method goes onto the current call stack rather than at the beginning of a new call stack. There will be no context-switching between threads because thread1 and thread2 will be treated as normal object not thread object.

Run() Start()

1. What is significance of using Volatile keyword? (Important)?

The Java volatile keyword is used to mark a Java variable as "being stored in main memory". More precisely that means, that every read of a volatile variable will be read from the computer's main memory, and not from the CPU cache, and that every write to a volatile variable will be written to main memory, and not just to the CPU cache.

If a field is declared [volatile](http://www.javamadesoeasy.com/2015/03/volatile-keyword-in-java-difference.html), in that case the Java memory model ensures that all threads see a consistent value for the variable.

<http://tutorials.jenkov.com/java-concurrency/volatile.html>

1. Differences between synchronized and volatile keyword in Java? (Important)

Before we move on let’s take a look at two important features of locks and synchronization.

Mutual Exclusion: It means that only one thread or process can execute a block of code (critical section) at a time.

Visibility: It means that changes made by one thread to shared data are visible to other threads.

**Java’s synchronized keyword guarantees both mutual exclusion and visibility.** If we make the blocks of threads that modifies the value of shared variable synchronized only one thread can enter the block and changes made by it will be reflected in the main memory. All other thread trying to enter the block at the same time will be blocked and put to sleep.

In some cases we may only desire the visibility and not atomicity. Use of synchronized in such situation is an overkill and may cause scalability problems. Here volatile comes to the rescue. Volatile variables have the visibility features of synchronized but not the atomicity features. The values of volatile variable will never be cached and all writes and reads will be done to and from the main memory.

Couple of points:-

* Volatile does not acquire any lock on variable or object, but Synchronization acquires lock on method or block in which it is used.
* Volatile variables are not cached, but variables used inside synchronized method or block are cached.
* When volatile is used will never create deadlock in program, as volatile never obtains any kind of lock. But in case if synchronization is not done properly, we might end up creating deadlock in program.
* Synchronization may cost us performance issues, as one thread might be waiting for another thread to release lock on object. But volatile is never expensive in terms of performance.

1. Can we have volatile methods in java?

No, volatile is only a keyword, can be used only with variables.

1. Can we have synchronized variable in java?

No, synchronized can be used only with methods, i.e. in method declaration.

1. Can you again start Thread?

No, we cannot start Thread again, doing so will throw runtimeException java.lang.IllegalThreadStateException. The reason is once run() method is executed by Thread, it goes into dead state.

1. What is race condition in multithreading and how can we solve it? (Important)

When more than one thread try to access same resource without synchronization causes race condition.

So we can solve race condition by using either synchronized block or synchronized method.

Example (Train ticket booking)-

Let's say there is only 1 ticket available in train, and two passengers are trying to book that ticket at same time without synchronization.

It might happen that both might end up booking up ticket, though only ticket was available, which is of course going to create problem.

But if synchronization was there only of them would have been able to book ticket.

1. What if two threads try to read same resource without synchronization?

When two threads try to read on same resource without synchronization, it’s never going to create any problem.

1. What if two threads try to write to same resource without synchronization?

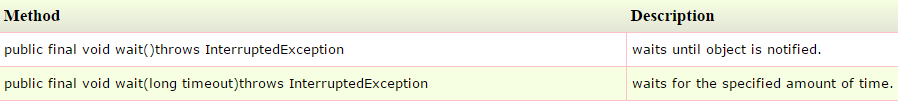
When two threads try to write to same resource without synchronization, it’s going to create synchronization problems.

1. How threads communicate between each other?

Threads can communicate with each other by using wait(), notify() and notifyAll() methods.

1. Explain wait(), notify() and notifyAll()

**Wait() –** Causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.



The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

**notify() –** Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. Syntax:

public final void notify()

**notifyAll() -** Wakes up all threads that are waiting on this object's monitor. Syntax:

public final void notifyAll()

1. Why wait(), notify() and notifyAll() are in Object class and not in Thread class? (Important)

Every Object has a monitor, acquiring that monitors allow thread to hold lock on object. But Thread class does not have any monitors.

As multiple threads exists on same object. Only one thread can hold object monitor at a time. As a result thread can notify other threads of same object that lock is available now. But, thread having these methods does not make any sense because multiple threads exists on object it’s no other way around (i.e. multiple objects exists on thread).

Wait(), notify() and notifyAll() method being in Object class allows all the threads created on that object to communicate with other.

1. Is it important to acquire object lock before calling wait(), notify() and notifyAll()?

Yes, it’s mandatory to acquire object lock before calling these methods on object. As discussed above wait(), notify() and notifyAll() methods are always called from Synchronized block only, and as soon as thread enters synchronized block it acquires object lock (by holding object monitor). If we call these methods without acquiring object lock i.e. from outside synchronize block then java.lang. IllegalMonitorStateException is thrown at runtime.

Wait() method needs to enclosed in try-catch block, because it throws compile time exception i.e. InterruptedException.

1. What is deadlock in multithreading? Write a program to form DeadLock in multi threading and also how to solve DeadLock situation. What measures you should take to avoid deadlock? (Important)

Deadlock is a situation where two threads are waiting for each other to release lock holded by them on resources.

**Example:-**

Thread-1 acquires lock on String.class and then calls sleep() method which gives Thread-2 the chance to execute immediately after Thread-1 has acquired lock on String.class and Thread-2 acquires lock on Object.class then calls sleep() method and now it waits for Thread-1 to release lock on String.class.

Conclusion: Now, Thread-1 is waiting for Thread-2 to release lock on Object.class and Thread-2 is

waiting for Thread-1 to release lock on String.class and deadlock is formed.

//Code called by Thread-1

public void run() {

synchronized (String.class) {

Thread.sleep(100);

synchronized (Object.class) {

}

}

}

//Code called by Thread-2

public void run() {

synchronized (Object.class) {

Thread.sleep(100);

synchronized (String.class) {

}

}

}

**How to solve this deadlock situation.**

Thread-1 acquires lock on String.class and then calls sleep() method which gives Thread-2 the chance to execute immediately after Thread-1 has acquired lock on String.class and Thread-2 tries to acquire lock on String.class but lock is holded by Thread-1. Meanwhile, Thread-1 completes successfully. As Thread-1 has completed successfully it releases lock on String.class, Thread-2 can now acquire lock on String.class and complete successfully without any deadlock formation.

Conclusion: No deadlock is formed.

//Code called by Thread-1

public void run() {

synchronized (String.class) {

Thread.sleep(100);

synchronized (Object.class) {

}

}

}

//Code called by Thread-2

public void run() {

synchronized (String.class) {

Thread.sleep(100);

synchronized (Object.class) {

}

}

}

**Few important measures to avoid deadlock.**

* Lock specific member variables of class rather than locking whole class:
* If possible try to use join() method, although it may refrain us from taking full advantage of multithreading environment because threads will start and end sequentially, but it can be handy in avoiding deadlocks.

1. Have you ever generated thread dumps or analyzed Thread Dumps? (Important)?

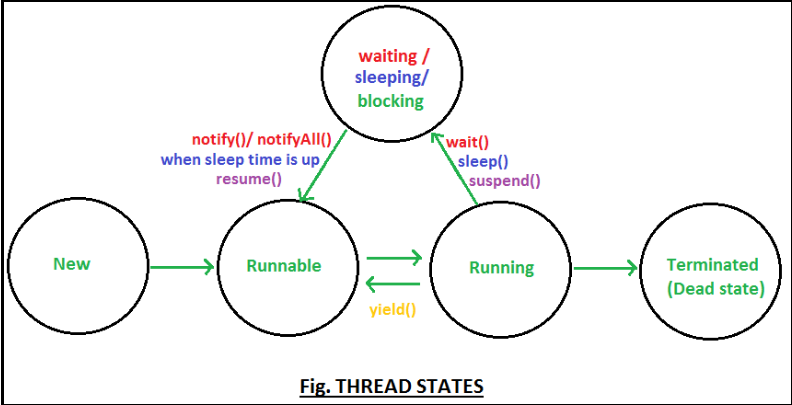
VisualVM is most popular way to generate Thread Dump.

jstack is very easy way to generate Thread dump.

1. What is life cycle of Thread, explain thread states? (Important)?

Thread have following states >

* **New: -** When instance of thread is created using new operator it is in new state, but the start() method has not been invoked on the thread yet, thread is not eligible to run yet.
* **Runnable: -** When start() method is called on thread it enters runnable state.
* **Running: -** Thread scheduler selects thread to go from runnable to running state. In running state Thread starts executing by entering run() method.
* **Waiting/blocked/sleeping: -** In this state a thread is not eligible to run.
* **Terminated (Dead):-** A thread is considered dead when its run() method completes.



1. How can Thread go from running to waiting state?

By calling wait()method, thread go from running to waiting state. In waiting state it will wait for other threads to release object monitor/lock.

1. How can Thread go from running to sleeping state?

By calling sleep() method, thread go from running to sleeping state. In sleeping state it will wait for sleep time to get over.

1. Are you aware of preemptive scheduling and time slicing?

In preemptive scheduling, the highest priority thread executes until it enters into the waiting or dead state.

In time slicing, a thread executes for a certain predefined time and then enters runnable pool. Than thread can enter running state when selected by thread scheduler.

1. What are daemon threads?

Daemon threads are low priority threads which runs intermittently in background for doing garbage collection.

* Thread scheduler schedules these threads only when CPU is idle.
* Daemon threads are service oriented threads, they serves all other threads.
* These threads are created before user threads are created and die after all other user threads dies.
* Priority of daemon threads is always 1 (i.e. MIN\_PRIORITY).
* User created threads are non daemon threads.
* We can use **isDaemon()** method to check whether thread is daemon thread or not.
* We can use **setDaemon(boolean on)** method to make any user method a daemon thread.
* If setDaemon(boolean on) is called on thread after calling start() method than IllegalThreadStateException is thrown.



**Servlet Interview Questions**

1. What is the life-cycle of a servlet?

The web container maintains the life cycle of a servlet instance.

* **Servlet class is loaded:** The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.
* **Servlet instance is created:** The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.
* **init method is invoked:** The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

public void init(ServletConfig config) throws ServletException

* **service method is invoked:** The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

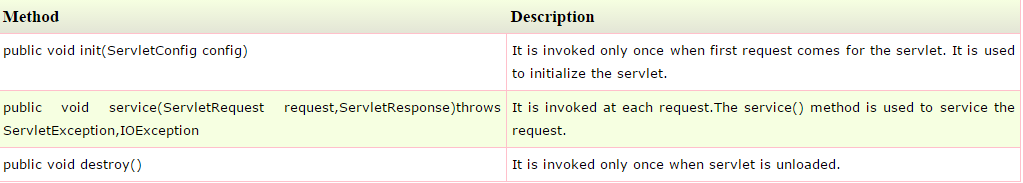
public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException

* **destroy method is invoked**: The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

public void destroy()

1. What are the life-cycle methods for a servlet?



1. How Servlet works?

The server checks if the servlet is requested for the first time.

If yes,

web container does the following tasks:

* + Loads the servlet class.
  + Instantiates the servlet class.
  + calls the init method passing the ServletConfig object

else

calls the service method passing request and response objects

The web container calls the destroy method when it needs to remove the servlet such as at time of stopping server or undeploying the project.

1. How web container handles the servlet request?

* Maps the request with the servlet in the web.xml file.
* creates request and response objects for this request
* calls the service method on the thread
* The public service method internally calls the protected service method
* The protected service method calls the doGet method depending on the type of request.
* The doGet method generates the response and it is passed to the client.
* After sending the response, the web container deletes the request and response objects. The thread is contained in the thread pool or deleted depends on the server implementation.

1. What is written inside the public service method?

The public service method converts the ServletRequest object into the HttpServletRequest type and ServletResponse object into the HttpServletResponse type. Then, calls the service method passing these objects.

1. What is written inside the protected service method?

The protected service method checks the type of request, if request type is get, it calls doGet method, if request type is post, it calls doPost method.

1. How many ways can you create a servlet?

There are three ways to create the servlet.

* By implementing the Servlet interface
* By inheriting the GenericServlet class
* By inheriting the HttpServlet class

1. How many objects of a servlet is created?

Only one object at the time of first request by servlet or web container.

1. Who is responsible to create the object of servlet?

The web container or servlet container.

1. When servlet object is created?

At the time of first request.

1. What is difference between Get and Post method?



1. What is difference between PrintWriter and ServletOutputStream?

PrintWriter is a character-stream class where as ServletOutputStream is a byte-stream class. The PrintWriter class can be used to write only character-based information whereas ServletOutputStream class can be used to write primitive values as well as character-based information.

1. What is difference between GenericServlet and HttpServlet?

The GenericServlet is protocol independent whereas HttpServlet is HTTP protocol specific. HttpServlet provides additional functionalities such as state management etc.

1. What is servlet collaboration?

When one servlet communicates to another servlet, it is known as servlet collaboration. There are many ways of servlet collaboration:

* RequestDispacher interface
* sendRedirect() method etc.

1. What is the purpose of RequestDispatcher Interface?

The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interceptor can also be used to include the content of another resource.

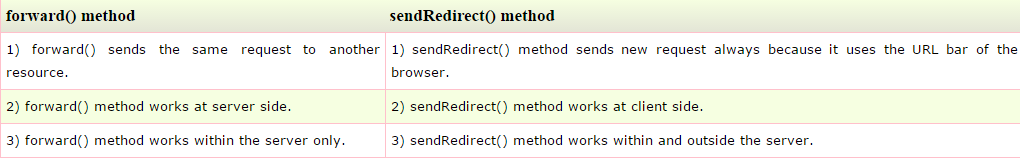
1. Can you call a jsp from the servlet?

Yes, one of the way is RequestDispatcher interface for example:

RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");

rd.forward(request,response);

1. Difference between forward() method and sendRedirect() method?



1. What is difference between ServletConfig and ServletContext?

The container creates object of ServletConfig for each servlet whereas object of ServletContext is created for each web application.

1. What is Session Tracking?

Session Tracking is a way to maintain state of an user.Http protocol is a stateless protocol.Each time user requests to the server, server treats the request as the new request.So we need to maintain the state of an user to recognize to particular user.

1. What are Cookies?

A cookie is a small piece of information that is persisted between the multiple client requests.

1. What is difference between Cookies and HttpSession?

Cookie works at client side whereas HttpSession works at server side.

1. What is the disadvantage of cookies?

It will not work if cookie is disabled from the browser.

1. What is filter?

A filter is an object that is invoked either at the preprocessing or postprocessing of a request. It is pluggable.

<https://www.tutorialspoint.com/servlets/servlets-writing-filters.htm>

1. How can we perform any action at the time of deploying the project?

By the help of ServletContextListener interface.

1. How can we upload the file to the server using servlet?

File upload and download are not supported by Servlet API. So we can use third party jar like **Apache FileUpload jar** or **cos.jar** from oreilly. Cos.jar provides a class named “MultipartRequest” which can be used to upload files onto server.

MultipartRequest m=new MultipartRequest(request,"d:/new");

out.print("successfully uploaded");

1. What is load-on-startup in servlet?

The load-on-startup element of servlet in web.xml is used to load the servlet at the time of deploying the project or server start. So it saves time for the response of first request.

<servlet>

<servlet-name>servlet1</servlet-name>

<servlet-class>com.javatpoint.FirstServlet</servlet-class>

**<load-on-startup>0</load-on-startup>**

</servlet>

If we pass negative value in <load-on-startup> attribute, servlet will be loaded at request time, at first request.

If we pass 0 or positive value, Container will load it on application startup.

If there are multiple servlets with load-on-startup value such as 0, 1, 2, 3, 4 then lower integer value servlet will be loaded first.

1. How to make sure a servlet is loaded at the application startup?

By Passing +ve value in load-on-startup value such as 0, 1, 2, 3, 4 in servlet configuration in web.xml.

If you specify the load-on-startup in web.xml (positive value), servlet will be loaded at project deployment time or server start. So, it will take less time for responding to first request.

1. Which event is fired at the time of project deployment and undeployment?

ServletContextEvent.

1. Which event is fired at the time of session creation and destroy?

HttpSessionEvent.

1. Which event is fired at the time of setting, getting or removing attribute from application scope?

ServletContextAttributeEvent.

1. What is the use of welcome-file-list in web.xml?

The welcome-file-list element of web-app, is used to define a list of welcome files. A welcome file is the file that is invoked automatically by the server, if you don't specify any file name.

By default server looks for the welcome file in following order:

* welcome-file-list in web.xml
* index.html
* index.htm
* index.jsp

If none of these files are found, server renders 404 error.

If you have specified welcome-file in web.xml, and all the files index.html, index.htm and index.jsp exists, priority goes to welcome-file.

If welcome-file-list entry doesn't exist in web.xml file, priority goes to index.html file then index.htm and at last index.jsp file.

1. How to get server information in a servlet?

getServletContext().getServletInfo();

1. How to get the IP Address of client in servlet?

We can use request.getRemoteAddr() to get the client IP address in servlet.

1. How do we go with database connection and log4j integration in servet?

Best approach is to initialize database connection in Servlet Context Listener and set it as a context attribute for other servlets to use.

1. What is the effective way to make sure all the servlets are accessible only when user has a valid session?

We can use Authentication Filter to check if request contains a valid session or now.

1. How do we call one servlet from another servlet?

RequestDispather.forward() method can be used to forward the request to another servlet. If we want to include the another servlet’s output, we can use RequestDispather.include() method.

1. How can we invoke another servlet in a different application?

We can use ServletResponse.sendRedirect() method and provide the complete URL of another servlet. If we have to send some data also, we can use cookies that will be part of the servlet response and sent in the request to another servlet.

1. What is servlet attributes and their scope?

An attribute in servlet is an object that can be set, get or removed from one of the following scopes:

* request scope – ServletRequest interface
* session scope – HttpSession interface
* application scope – ServletRequest interface

1. Are Servlets Thread Safe? How to achieve thread safety in servlets?

Init() and destroy() method are called only once in servlet life cycle. So we don’t need to worry about their synchronization. But service method like doGet() or doPost() gets called in every client request. We should provide thread safety in these methods.

If there are local variables in service methods, we don’t need to worry about their thread safety because they are specific to each thread. But if we have a shared resource then we can use synchronization to achieve thread safety in servlets. The thread safety is similar to standalone applications.

1. What is SingleThreadModel interface?

SingleThreadModel interface was provided for Thread safety and it guarantees that no two threads will execute concurrently in the Servlet’s service method. However SingleThreadModel does not solve all thread safety issues. For Ex:- session attributes and static variables can still be accessed by multiple requests on multiple threads at the same time, even when SingleThreadModel servles are used.

Also it takes out all the benefits of multithreading support of servlets. That’s why this interface is deprecated in Servlet 2.4

1. What is the use of servlet wrapper classes?

Servlet HTTP API provides two wrapper classes. HttpServletRequestWrapper and HttpServletResponseWrapper. These wrapper classes are provided to help developers with custom implementation of servlet request and response. We can extend these classes and implement for custom request and response.

1. Can we get PrintWriter and ServletOutputStrem both in a servlet?

We can’t get instances of both PrintWriter and ServletOutputStrem in a single servlet. If we invoke both the methods; getWriter() and getOutputStrem()on response; we will get java.lang.IllegalStateException at runtime.

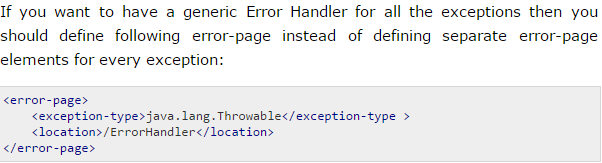
1. How can we create deadlock situation in servlet?

Just call doPost() method from doGet() and doPost() method from doGet() method.

1. Exception handling in servlet?

When a servlet throws an exception, the web container searches the configurations in web.xml that use the exception-type element for a match with the thrown exception type.

Web.xml:

We can handle these exceptions in ErrorHandler servlet.

1. How can we do auto refresh in servlet?

The simplest way of refreshing a web page is using method setIntHeader() of response object. Following is the signature of this method:

public void setIntHeader(String header, int headerValue)

This method sends back header "Refresh" to the browser along with an integer value which indicates time interval in seconds.

1. ServletContextEvent and ServletContextListener?

The ServletContextEvent is notified when web application is deployed on the server. If you want to perform some action at the time of deploying the web application such as creating database connection, creating all the tables of the project etc, you need to implement ServletContextListener interface and provide the implementation of its methods.

**Methods in ServletContextListener:**

public void contextInitialized(ServletContextEvent e): is invoked when application is deployed on the server.

public void contextDestroyed(ServletContextEvent e): is invoked when application is undeployed from the server.

1. HttpSessionEvent and HttpSessionListener?

The HttpSessionEvent is notified when session object is changed. The corresponding Listener interface for this event is HttpSessionListener.

We can perform some operations at this event such as counting total and current logged-in users, maintaing a log of user details such as login time, logout time etc.

**Methods in HttpSessionListener:**

public void sessionCreated(HttpSessionEvent e): is invoked when session object is created.

public void sessionDestroyed(HttpSessionEvent e): is invoked when session is invalidated.

**Hibernate Interview Questions**

1. What is JDBC?

JDBC stands for Java Database Connectivity and provides a set of Java API for accessing the relational databases from Java program. These Java APIs enables Java programs to execute SQL statements and interact with any SQL compliant database.

1. What is ORM?

ORM stands for Object-Relational Mapping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc.

1. What are the advantages of ORM over JDBC?

* **Fast Performance:** The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.
* **Database Independent query:** HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.
* **Automatic table creation:** Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.
* Let’s business code access objects rather than DB tables.

1. Name some of the ORM frameworks based on JAVA?

Enterprise JavaBeans Entity Beans, Spring DAO, Hibernate etc.

1. What is Hibernate?

Hibernate is an Object-Relational Mapping (ORM) solution for JAVA. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application. Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

1. What are the advantages of using Hibernate?

* Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.
* Provides simple APIs for storing and retrieving Java objects directly to and from the database.
* If there is change in Database or in any table then the only need to change XML file properties.
* Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects.
* Manipulates Complex associations of objects of your database.

1. Name some of the java based tools/frameworks that supports hibernate integration.

Eclipse plug-ins, Maven, J2EE, XDoclet Spring etc.

1. What are the core interfaces of Hibernate? / What are the key components/objects of hibernate?

**Configuration -** Represents a configuration or properties file required by the Hibernate.

**SessionFactory -** Configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated.

**Session -** Used to get a physical connection with a database.

**Transaction -** Represents a unit of work with the database and most of the RDBMS supports transaction functionality.

**Query -** Uses SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects.

**Criteria -** Used to create and execute object oriented criteria queries to retrieve objects.

1. What are the two key components of a hibernate configuration object?

The Configuration object provides two keys components:

**Database Connection:** This is handled through one or more configuration files supported by Hibernate. These files are hibernate.properties and hibernate.cfg.xml.

**Class Mapping Setup:** This component creates the connection between the Java classes and database tables.

1. What is a configuration object in hibernate?

The Configuration object is the first Hibernate object you create in any Hibernate application and usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate.

1. What is a SessionFactory in hibernate?

Configuration object is used to create a SessionFactory object which in turn configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.

**Note:-** ThreadSafe means…the SessionFactory object’s state cannot be changed by any thread accessing it. It does not mean threads cant access simultaneously.  
Multiple threads can safely access a ThreadSafe object-SessionFactory.

The SessionFactory is heavyweight object so usually it is created during application start up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So if you are using multiple databases then you would have to create multiple SessionFactory objects.

SessionFactory provides the instance of Session. It is a factory of Session.

The internal state of a SessionFactory is immutable. Once it is created this internal state is set. This internal state includes all of the metadata about Object/Relational Mapping.

It holds the data of second level cache that is not enabled by default.

Session is thread-safe object.

1. What is Session in hibernate?

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.

It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances. This holds first level cache that is enabled by default.

Session is not a thread-safe object, many threads can access it simultaneously

1. What is Transaction in hibernate?

A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality. This is an optional object and Hibernate applications may choose not to use this interface, instead managing transactions in their own application code.

Session.beginTransaction() method begins a unit of work and returns the associated Transaction object.

1. What is Query in hibernate?

Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects.

1. What is Criteria in hibernate?

Hibernate provides Criteria API that is more object oriented for querying the database and getting results. We can’t use Criteria to run update or delete queries or any DDL statements. It’s only used to fetch the results from the database using more object oriented approach.

Crietria c=session.createCriteria(Emp.class);

c.addOrder(Order.asc("salary"));

List list=c.list();

List employees = session.createCriteria(Employee.class)

.add(Restrictions.like("name", "a%") )

.add(Restrictions.like("address", "Boston"))

.addOrder(Order.asc("name") )

.list();

1. Name some of the properties you would require to configure for a databases in a standalone situation?



hibernate.dialect: This property makes Hibernate generate the appropriate SQL for the chosen database.

1. What are the three states of a persistent entity at a given point in time?

Instances may exist in one of the following three states at a given point in time:

**transient:** A new instance of a persistent class which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.

**persistent:** You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.

**detached:** Once we close the Hibernate Session, the persistent instance will become a detached instance.

1. Which method is used to create a HQL query?

Session.createQuery() creates a new instance of Query for the given HQL query string.

1. Which method is used to create a SQL query?

Session.createSQLQuery() creates a new instance of SQLQuery for the given SQL query string.

1. Which method is used to remove a persistent instance from the datastore?

Session.delete() removes a persistent instance from the datastore.

1. Which method is used to get a persistent instance from the datastore?

Session.get() returns the persistent instance of the given named entity with the given identifier, or null if there is no such persistent instance.

1. Which method is used to re-read the state of the given instance from the underlying database?

Session.refresh() re-reads the state of the given instance from the underlying database.

1. Which method is used to save the state of the given instance from the underlying database?

Session.save()

1. Which method is used to update the state of the given instance from the underlying database?

Session.update()

1. Which method is used to save or update the state of the given instance from the underlying database?

Session.saveOrUpdate either saves(Object) or updates(Object) the given instance.

1. What persistent classes are in hibernate?

Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate.

1. What are the best practices that hibernate recommends for persistent classes.

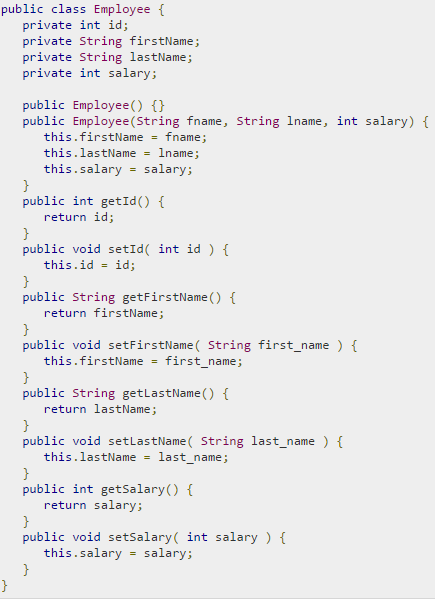
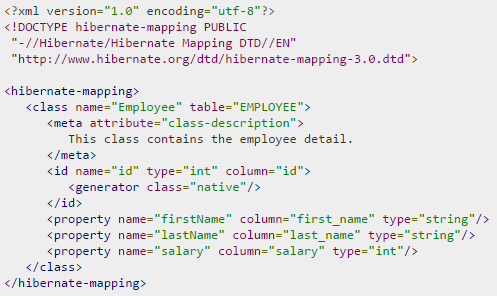
There are following main rules of persistent classes, however, none of these rules are hard requirements.

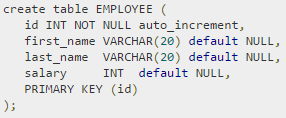
* All Java classes that will be persisted need a default constructor.
* All classes should contain an ID in order to allow easy identification of your objects within Hibernate and the database. This property maps to the primary key column of a database table.
* All attributes that will be persisted should be declared private and have getXXX and setXXX methods defined in the JavaBean style.

1. Where Object/relational mappings are defined in hibernate?

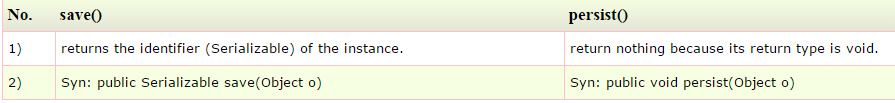
An Object/relational mappings are usually defined in an XML document. We should save the mapping document in a file with the format <classname>.hbm.xml.

1. Example of persistent class and it’s mapping?



1. What is the difference between session.save() and session.persist() method?



For Example:

System.out.println(session.save(aCoin)); - will print the generated primary key.

System.out.println(session.persist(aCoin)); - will throw a compile time error because persist()'s return type is void.

1. What is the difference between session.save() and session.saveOrUpdate() method?

* The main difference between save() and saveOrUpdate() method is that save() method performs an INSERT operation to store the object into the database, but INSERT will fail if the primary key is already persistent i.e. object already exists in the database. This is why, you should only call save() with an absolutely new object which doesn't have any database identifier. Calling save() with the detached object will fail. This is opposite of saveOrUpdate() method, which can do either INSERT or UPDATE SQL query depending upon whether an object exists in the database or not. The saveOrUpdate() method first executes a SELECT query to determine if it needs to do an INSERT or UPDATE operation.
* Another key difference between save() and saveOrUpdate() method is that former is used to bring a transient object to persistent state but saveOurUpdate() can bring both transient (new) and detached (existing) object into persistent state. It is often used to re-attach a detached object into Session.

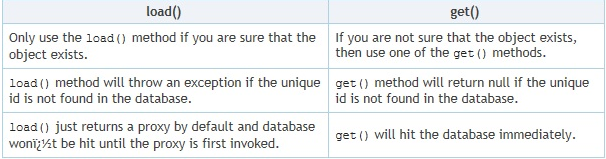
1. What is the difference between get and load method?

session.load() –

* It will always return a “proxy” (Hibernate term) without hitting the database. In Hibernate, proxy is an object with the given identifier value, its properties are not initialized yet, it just look like a temporary fake object.
* If no row found, it will throws an ObjectNotFoundException.

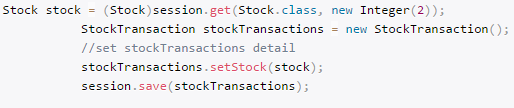
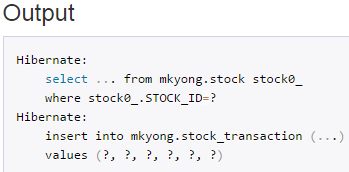
session.get() –

* It always hit the database and return the real object, an object that represent the database row, not proxy.
* If no row found, it return null.

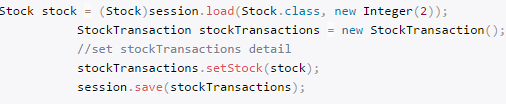
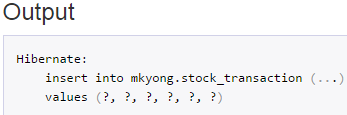


**Point#1 Session.load() is better performant Example** in a Stock application , Stock and StockTransactions should have a “one-to-many” relationship, when you want to save a stock transaction, it’s common to declared something like below.

**Session.get():**In session.get(), Hibernate will hit the database to retrieve the Stock object and put it as a reference to StockTransaction. However, this save process is extremely high demand, there may be thousand or million transactions per hour, do you think is this necessary to hit the database to retrieve the Stock object everything save a stock transaction record? After all you just need the Stock’s Id as a reference to StockTransaction.

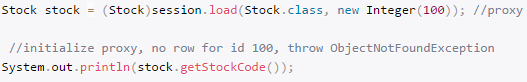
**Session.load():** **session.load()** will be your good solution.

In session.load(), Hibernate will not hit the database (no select statement in output) to retrieve the Stock object, it will return a Stock proxy object – a fake object with given identify value. In this scenario, a proxy object is enough for to save a stock transaction record.

**Point#2 Exception case**

session.load() - It will always return a proxy object with the given identity value, even the identity value is not exists in database. However, when you try to initialize a proxy by retrieve it’s properties from database, it will hit the database with select statement. If no row is found, **ObjectNotFoundException** will throw.

session.load() - It will always return null , if the identity value is not found in database.



1. What is the difference between update and merge method?

At first look both update() and merge() methods seems similar because both of them are used to convert the object which is in detached state into persistence state but there are some differences as well.

* update() and merge() both methods are used to update the db row if it already exist. But if object does not exist in db already then the update() method will throw the exception while merge() will create the new entry in db for that object.
* If session does not contains an already persistent instance with the same identifier and if you are sure about that then use update to save the data. But merge() method can save your modifications at any time with out having the knowledge about the state of session.



1. What is difference between openSession and getCurrentSession?

Hibernate SessionFactory getCurrentSession() method returns the session bound to the context. But for this to work, we need to configure it in hibernate configuration file. Since this session object belongs to the hibernate context, we don’t need to close it. Once the session factory is closed, this session object gets closed.



Hibernate SessionFactory openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment.

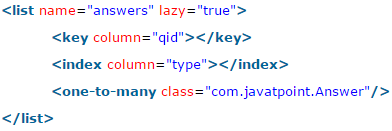
1. What is SessionFactory.openStatelessSession?

* Hibernate SessionFactory openStatelessSession() method returns instance of StatelessSession. There is another overloaded method where we can pass java.sql.Connection object to get a stateless session object from hibernate.
* StatelessSession in Hibernate does not implement first-level cache and it doesn’t interact with any second-level cache. Collections are also ignored by a stateless session.
* However, stateless session can be a good fit in certain situations. For example where we are loading bulk data into database and we don’t want hibernate session to hold huge data in first-level cache memory.

1. What is lazy loading?

Lazy loading is a technique in which objects are loaded on demand basis. Since Hibernate 3, lazy loading is by default, enabled so that child objects are not loaded when parent is loaded. Lazy collection loads the child objects on demand, it is used to improve performance.

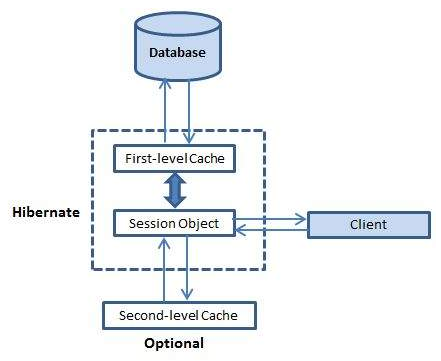
To use lazy collection, you may optionally use lazy="true" attribute in your collection. It is by default true, so you don't need to do this. If you set it to false, all the child objects will be loaded initially which will decrease performance in case of big data.



1. What is caching in hibernate?

Caching is all about application performance optimization and it sits between your application and the database to avoid the number of database hits as many as possible to give a better performance for performance critical applications.

Caching is important to Hibernate as well which utilizes a multilevel caching schemes as explained below:



**First-level cache:**

The first-level cache is the Session cache and is a mandatory cache through which all requests must pass. The Session object keeps an object under its own power before committing it to the database.

If you issue multiple updates to an object, Hibernate tries to delay doing the update as long as possible to reduce the number of update SQL statements issued. If you close the session, all the objects being cached are lost and either persisted or updated in the database.

Hibernate first level cache is associated with the Session object. Hibernate first level cache is enabled by default and there is no way to disable it. However hibernate provides methods through which we can delete selected objects from the cache or clear the cache completely.

**Second-level cache:**

Second level cache is an optional cache and first-level cache will always be consulted before any attempt is made to locate an object in the second-level cache. The second-level cache can be configured on a per-class and per-collection basis and mainly responsible for caching objects across sessions.

Any third-party cache can be used with Hibernate. An **org.hibernate.cache.CacheProvider** interface is provided, which must be implemented to provide Hibernate with a handle to the cache implementation.

**SessionFactory** holds the second level cache data. It is global for all the session objects and not enabled by default.

Different vendors have provided the implementation of Second Level Cache.

EH Cache

OS Cache

Swarm Cache

JBoss Cache

**Concurrency or Caching strategies:**

There are some caching strategies which you have to keep in mind when using a second-level cache:

* **Read Only** This strategy should be used for persistent objects that will be never updated. It is good for reading and caching application configuration and other static data. This is the simplest strategy with best performance because there is no overload to check if an entity is updated in database or not.
* **Read-Write** This strategy is good for entities which are updated by the application. However if the data is updated either through the database or other applications, then there is no way Hibernate could tell if there was a change or not and your data might be stale.
* **Nonrestricted Read-Write** If the application only occasionally updates data and strict transaction isolation is not required, this caching strategy might be appropriate.
* **Transactional** This caching strategy provides support for fully transactional cache providers such as JBoss TreeCache. Such a cache can only be used in a JTA environment and you must specify transaction.manager\_lookup\_class.

**3 extra steps for second level cache example using EH cache**

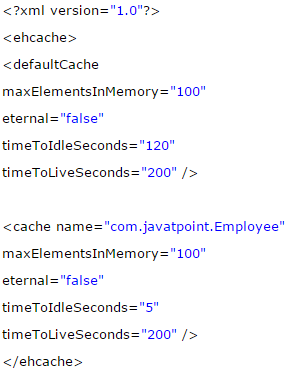
* Add 2 configuration setting in hibernate.cfg.xml file



* Add cache usage setting in hbm file



* Create ehcache.xml file



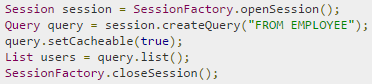
**Query-level cache:**

Hibernate also implements a cache for query resultsets that integrates closely with the second-level cache.

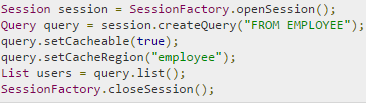
This is an optional feature and requires two additional physical cache regions that hold the cached query results and the timestamps when a table was last updated. This is only useful for queries that are run frequently with the same parameters.

To use the query cache, you must first activate it using the **hibernate.cache.use\_query\_cache="true"** property in the configuration file. By setting this property to true, you make Hibernate create the necessary caches in memory to hold the query and identifier sets.

Next, to use the query cache, you use the **setCacheable(Boolean)** method of the Query class. For example:



Hibernate also supports very fine-grained cache support through the concept of a cache region. A cache region is part of the cache that's given a name.



This code uses the method to tell Hibernate to store and look for the query in the employee area of the cache.

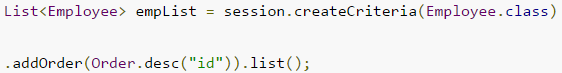
1. What will happen if we don’t have no-args constructor in Entity bean?

Hibernate uses Reflection API to create instance of Entity beans, usually when you call get() or load() methods. The method Class.newInstance() is used for this and it requires no-args constructor. So if you won’t have no-args constructor in entity beans, hibernate will fail to instantiate it and you will get HibernateException.

1. What is difference between sorted collection and ordered collection, which one is better?

When we use Collection API sorting algorithms to sort a collection, it’s called sorted list. For small collections, it’s not much of an overhead but for larger collections it can lead to slow performance and OutOfMemory errors. Also the entity beans should implement Comparable or Comparator interface for it to work, read more at java object list sorting.

If we are using Hibernate framework to load collection data from database, we can use it’s Criteria API to use “order by” clause to get ordered list. Below code snippet shows you how to get it.



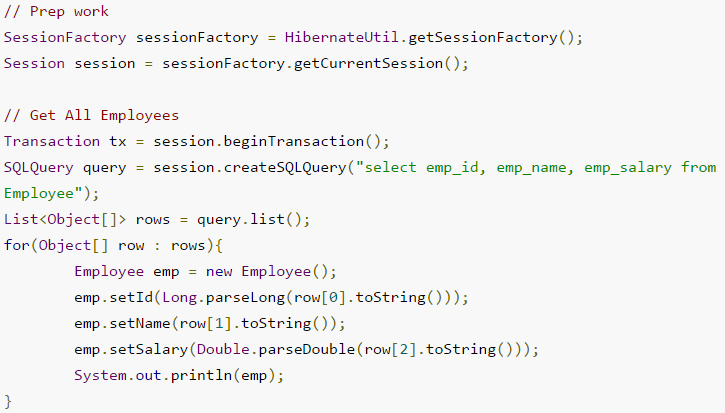
Ordered list is better than sorted list because the actual sorting is done at database level, that is fast and doesn’t cause memory issues.

1. Why we should not make Entity Class final?

Hibernate use proxy classes for lazy loading of data, only when it’s needed. This is done by extending the entity bean, if the entity bean will be final then lazy loading will not be possible, hence low performance.

1. Can we execute native sql query in hibernate?

Yes. For Hibernate Native SQL Query, we use Session.createSQLQuery(String query) to create the SQLQuery object and execute it. For example, if you want to read all the records from Employee table, we can do it through below code.



1. How to log hibernate generated sql queries in log files?

We can set below property for hibernate configuration to log SQL queries.

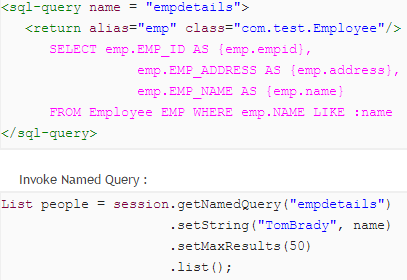


However we should use it only in Development or Testing environment and turn it off in production environment.

1. What is Named SQL Query?

Hibernate provides Named Query that we can define at a central location and use them anywhere in the code. We can created named queries for both HQL and Native SQL.

Hibernate Named Queries can be defined in Hibernate mapping files or through the use of JPA annotations @NamedQuery and @NamedNativeQuery.



1. What are the benefits of Named SQL Query?

* Hibernate Named Query helps us in grouping queries at a central location rather than letting them scattered all over the code.
* Hibernate Named Query syntax is checked when the hibernate session factory is created, thus making the application fail fast in case of any error in the named queries.
* Hibernate Named Query is global, means once defined it can be used throughout the application.

However one of the major disadvantage of Named query is that it’s hard to debug, because we need to find out the location where it’s defined.

1. What is Hibernate Proxy and how it helps in lazy loading?

Hibernate uses proxy object to support lazy loading. Basically when you load data from tables, hibernate doesn’t load all the mapped objects. As soon as you reference a child or lookup object via getter methods, if the linked entity is not in the session cache, then the proxy code will go to the database and load the linked object. It uses javassist to effectively and dynamically generate sub-classed implementations of your entity objects.

1. How transaction management works in Hibernate?

Transaction management is very easy in hibernate because most of the operations are not permitted outside of a transaction. So after getting the session from SessionFactory, we can call session beginTransaction() to start the transaction. This method returns the Transaction reference that we can use later on to either commit or rollback the transaction.

1. How to integrate log4j logging in hibernate application?

Hibernate 4 uses JBoss logging rather than slf4j used in earlier versions. For log4j configuration, we need to follow below steps.

* Add log4j dependencies for maven project, if not maven then add corresponding jar files.
* Create log4j.xml configuration file or log4j.properties file and keep it in the classpath. You can keep file name whatever you want because we will load it in next step.
* For standalone projects, use static block to configure log4j using DOMConfigurator or PropertyConfigurator. For web applications, you can use ServletContextListener to configure it.

That’s it, our setup is ready. Create org.apache.log4j.Logger instance in the java classes and start logging.

1. How to use application server JNDI DataSource with Hibernate framework?

For web applications, it’s always best to allow servlet container to manage the connection pool. That’s why we define JNDI resource for DataSource and we can use it in the web application. It’s very easy to use in Hibernate, all we need is to remove all the database specific properties and use below property to provide the JNDI DataSource name.



<http://www.journaldev.com/2905/hibernate-tomcat-jndi-datasource-example-tutorial>

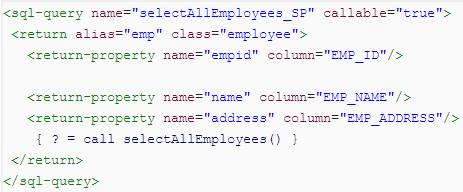
1. How to integrate Hibernate with Servlet or Struts2 web applications?

Hibernate integration with Servlet or Struts2 needs to be done using ServletContextListener.

1. Should all the mapping files of hibernate have .hbm.xml extension to work properly?

No, having .hbm.xml extension is a convention and not a requirement for hibernate mapping file names. We can have any extension for these mapping files.

1. How do you invoke Stored Procedures?



1. Which design patterns are used in Hibernate framework?

Some of the design patterns used in Hibernate Framework are:

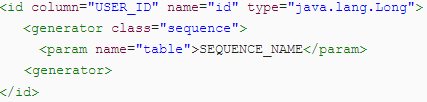
* Domain Model Pattern – An object model of the domain that incorporates both behavior and data.
* Data Mapper – A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.
* Proxy Pattern - for lazy loading
* Factory pattern - in SessionFactory

1. What are best practices to follow with Hibernate framework?

* Use native sql query only when it can’t be done using HQL, such as using database specific feature.
* If you have to sort the collection, use ordered list rather than sorting it using Collection API.
* Use named queries wisely, keep it at a single place for easy debugging. Use them for commonly used queries only. For entity specific query, you can keep them in the entity bean itself.
* For web applications, always try to use JNDI DataSource rather than configuring to create connection in hibernate.
* For collections, try to use Lists, maps and sets. Avoid array because you don’t get benefit of lazy loading.
* Do not treat exceptions as recoverable, roll back the Transaction and close the Session. If you do not do this, Hibernate cannot guarantee that in-memory state accurately represents the persistent state.
* Prefer DAO pattern for exposing the different methods that can be used with entity bean
* Prefer lazy fetching for associations.

1. How do you define sequence generated primary key in hibernate?

Using <generator> tag.



1. How can a whole class be mapped as immutable?

Mark the class as mutable="false" (Default is true). This specifies that instances of the class are (not) mutable. Immutable classes, means the updates to this class will be ignored, but no exception is thrown, only the add and delete operation are allow.



1. What is automatic dirty checking?

Automatic dirty checking is a feature that saves us the effort of explicitly asking Hibernate to update the database when we modify the state of an object inside a transaction.

1. What are Callback interfaces?

Callback interfaces allow the application to receive a notification when something interesting happens to an object—for example, when an object is loaded, saved, or deleted. Hibernate applications don't need to implement these callbacks, but they're useful for implementing certain kinds of generic functionality.

1. How can Hibernate be configured to access an instance variable directly and not through a setter method?

By mapping the property with access="field" in Hibernate metadata. This forces hibernate to bypass the setter method and access the instance variable directly while initializing a newly loaded object.

1. What are the types of inheritance models in Hibernate?

We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

**Table Per Hierarchy:** In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table.

**Table Per Concrete class:** In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables.

**Table Per Subclass:** In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.

1. What are the Collection types in Hibernate?

Bag

Set

List

Array

Map

1. How can we get hibernate statistics?

We can get hibernate statistics using getStatistics() method of SessionFactory class as shown below:  
SessionFactory.getStatistics()

1. What’s the use of session.lock() in hibernate?

session.lock() method of session class is used to reattach an object which has been detached earlier. This method of reattaching doesn’t check for any data synchronization in database while reattaching the object and hence may lead to lack of synchronization in data.

1. What is cascading and what are different types of cascading?
2. How to implement relationships in hibernate?

We can easily implement one-to-one, one-to-many and many-to-many relationships in hibernate. It can be done using JPA annotations as well as XML based configurations. For better understanding, you should go through following tutorials.