[Core Java](#CoreJava)

[Hibernate](#Hibernate)

OOPS Interview Questions

**1.What are the principle concepts of OOPS?**

There are four principle concepts upon which object oriented design and programming rest. They are:

* Abstraction
* Polymorphism
* Inheritance
* Encapsulation

(i.e. easily remembered as A-PIE).

**2.What is Abstraction?**

Abstraction refers to the act of representing essential features without including the background details or explanations.

**3.What is Encapsulation?**

Encapsulation is a technique used for hiding the properties and behaviors of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.

**4.What is the difference between abstraction and encapsulation?**

* **Abstraction** focuses on the outside view of an object (i.e. the interface) **Encapsulation** (information hiding) prevents clients from seeing it’s inside view, where the behavior of the abstraction is implemented.
* **Abstraction** solves the problem in the design side while **Encapsulation** is the Implementation.
* **Encapsulation** is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs.

**5.What is Inheritance?**

* Inheritance is the process by which objects of one class acquire the properties of objects of another class.
* A class that is inherited is called a superclass.
* The class that does the inheriting is called a subclass.
* Inheritance is done by using the keyword extends.
* The two most common reasons to use inheritance are:
  + To promote code reuse
  + To use polymorphism

**6.What is Polymorphism?**

Polymorphism is briefly described as "one interface, many implementations." Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form.

**7.How does Java implement polymorphism?**

(Inheritance, Overloading and Overriding are used to achieve Polymorphism in java).  
Polymorphism manifests itself in Java in the form of multiple methods having the same name.

* In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods).
* In other cases, multiple methods have the same name, same return type, and same formal argument list (overridden methods).

**8.Explain the different forms of Polymorphism.**

There are two types of polymorphism one is **Compile time polymorphism** and the other is run time polymorphism. Compile time polymorphism is method overloading. **Runtime time polymorphism** is done using inheritance and interface.  
**Note**: *From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:*

* *Method overloading*
* *Method overriding through inheritance*
* *Method overriding through the Java interface*

|  |
| --- |
|  |

**9.What is runtime polymorphism or dynamic method dispatch?**

In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

**10.What is Dynamic Binding?**

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

**11.What is method overloading?**

Method Overloading means to have two or more methods with same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same semantic operation but differ by argument number or type.  
**Note**:

* *Overloaded methods MUST change the argument list*
* *Overloaded methods CAN change the return type*
* *Overloaded methods CAN change the access modifier*
* *Overloaded methods CAN declare new or broader checked exceptions*
* *A method can be overloaded in the same class or in a subclass*

**12.What is method overriding?**

Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass. The key benefit of overriding is the ability to define behavior that’s specific to a particular subclass type.  
**Note**:

* *The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can’t override a method marked public and make it protected).*
* *You cannot override a method marked final*
* *You cannot override a method marked static*

**13.What are the differences between method overloading and method overriding?**

|  |  |  |
| --- | --- | --- |
|  | **Overloaded Method** | **Overridden Method** |
| **Arguments** | Must change | Must not change |
| **Return type** | Can change | Can’t change except for covariant returns |
| **Exceptions** | Can change | Can reduce or eliminate. Must not throw new or broader checked exceptions |
| **Access** | Can change | Must not make more restrictive (can be less restrictive) |
| **Invocation** | Reference type determines which overloaded version is selected. Happens at compile time. | Object type determines which method is selected. Happens at runtime. |

**14.Can overloaded methods be override too?**

Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. Compiler will not binding the method calls since it is overloaded, because it might be overridden now or in the future.

**15.Is it possible to override the main method?**

NO, because main is a static method. A static method can't be overridden in Java.

**16.How to invoke a superclass version of an Overridden method?**

To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance, or use the super prefix in the subclass itself. From the point of the view of the subclass, the super prefix provides an explicit reference to the superclass' implementation of the method.

// From subclass

super.overriddenMethod();

**17.What is super?**

super is a keyword which is used to access the method or member variables from the superclass. If a method hides one of the member variables in its superclass, the method can refer to the hidden variable through the use of the super keyword. In the same way, if a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword.   
**Note**:

* *You can only go back one level.*
* *In the constructor, if you use super(), it must be the very first code, and you cannot access any*this.xxx*variables or methods to compute its parameters.*

**18.How do you prevent a method from being overridden?**

To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means "this is the final implementation of this method", the end of its inheritance hierarchy.

public final void exampleMethod() {  
                         // Method statements  
                         }

**19.What is an Interface?**

An interface is a description of a set of methods that conforming implementing classes must have.  
**Note**:

* *You can’t mark an interface as final.*
* *Interface variables must be static.*
* *An Interface cannot extend anything but another interfaces.*

|  |
| --- |
|  |

**20.Can we instantiate an interface?**

You can’t instantiate an interface directly, but you can instantiate a class that implements an interface.

**21.Can we create an object for an interface?**

Yes, it is always necessary to create an object implementation for an interface. Interfaces cannot be instantiated in their own right, so you must write a class that implements the interface and fulfill all the methods defined in it.

**22.Do interfaces have member variables?**

Interfaces may have member variables, but these are implicitly public, static, and final- in other words, interfaces can declare only constants, not instance variables that are available to all implementations and may be used as key references for method arguments for example.

**23.What modifiers are allowed for methods in an Interface?**

Only public and abstract modifiers are allowed for methods in interfaces.

**24.What is a marker interface?**

Marker interfaces are those which do not declare any required methods, but signify their compatibility with certain operations. The java.io.Serializableinterface and Cloneable are typical marker interfaces. These do not contain any methods, but classes must implement this interface in order to be serialized and de-serialized.

**25.What is an abstract class?**

Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.   
**Note**:

* *If even a single method is abstract, the whole class must be declared abstract.*
* *Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.*
* *You can’t mark a class as both abstract and final.*

**26.Can we instantiate an abstract class?**

An abstract class can never be instantiated. Its sole purpose is to be extended (subclassed).

**27.What are the differences between Interface and Abstract class?**

|  |  |
| --- | --- |
| **Abstract Class** | **Interfaces** |
| An abstract class can provide complete, default code and/or just the details that have to be overridden. | An interface cannot provide any code at all,just the signature. |
| In case of abstract class, a class may extend only one abstract class. | A Class may implement several interfaces. |
| An abstract class can have non-abstract methods. | All methods of an Interface are abstract. |
| An abstract class can have instance variables. | An Interface cannot have instance variables. |
| An abstract class can have any visibility: public, private, protected. | An Interface visibility must be public (or) none. |
| If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly. | If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method. |
| An abstract class can contain constructors . | An Interface cannot contain constructors . |
| Abstract classes are fast. | Interfaces are slow as it requires extra indirection to find corresponding method in the actual class. |

**28.When should I use abstract classes and when should I use interfaces?**

**Use Interfaces when…**

* You see that something in your design will change frequently.
* If various implementations only share method signatures then it is better to use Interfaces.
* you need some classes to use some methods which you don't want to be included in the class, then you go for the interface, which makes it easy to just implement and make use of the methods defined in the interface.

**Use Abstract Class when…**

* If various implementations are of the same kind and use common behavior or status then abstract class is better to use.
* When you want to provide a generalized form of abstraction and leave the implementation task with the inheriting subclass.
* Abstract classes are an excellent way to create planned inheritance hierarchies. They're also a good choice for nonleaf classes in class hierarchies.

**29.When you declare a method as abstract, can other nonabstract methods access it?**

Yes, other nonabstract methods can access a method that you declare as abstract.

**30.Can there be an abstract class with no abstract methods in it?**

Yes, there can be an abstract class without abstract methods.

|  |
| --- |
|  |

**31.What is Constructor?**

* A constructor is a special method whose task is to initialize the object of its class.
* It is special because its name is the **same as the class name**.
* They do not have return types, not even **void** and therefore they cannot return values.
* They **cannot be inherited**, though a derived class can call the base class constructor.
* Constructor is invoked whenever an object of its associated class is created.

**32.How does the Java default constructor be provided?**

If a class defined by the code does **not** have any constructor, compiler will automatically provide one no-parameter-constructor (default-constructor) for the class in the byte code. The access modifier (public/private/etc.) of the default constructor is the same as the class itself.

**33.Can constructor be inherited?**

No, constructor cannot be inherited, though a derived class can call the base class constructor.

**34.What are the differences between Contructors and Methods?**

|  |  |  |
| --- | --- | --- |
|  | **Constructors** | **Methods** |
| **Purpose** | Create an instance of a class | Group Java statements |
| **Modifiers** | Cannot be *abstract, final, native, static*, or *synchronized* | Can be *abstract, final, native, static*, or *synchronized* |
| **Return Type** | No return type, not even void | void or a valid return type |
| **Name** | Same name as the class (first letter is capitalized by convention) -- usually a noun | Any name except the class. Method names begin with a lowercase letter by convention -- usually the name of an action |
| ***This*** | Refers to another constructor in the same class. If used, it must be the first line of the constructor | Refers to an instance of the owning class. Cannot be used by static methods. |
| ***Super*** | Calls the constructor of the parent class. If used, must be the first line of the constructor | Calls an overridden method in the parent class |
| **Inheritance** | Constructors are not inherited | Methods are inherited |

**35.How are this() and super() used with constructors?**

* Constructors use *this* to refer to another constructor in the same class with a different parameter list.
* Constructors use *super* to invoke the superclass's constructor. If a constructor uses *super*, it must use it in the first line; otherwise, the compiler will complain.

**36.What are the differences between Class Methods and Instance Methods?**

|  |  |
| --- | --- |
| **Class Methods** | **Instance Methods** |
| Class methods are methods which are declared as static. The method can be called without creating an instance of the class | Instance methods on the other hand require an instance of the class to exist before they can be called, so an instance of a class needs to be created by using the new keyword. Instance methods operate on specific instances of classes. |
| Class methods can only operate on class members and not on instance members as class methods are unaware of instance members. | Instance methods of the class can also not be called from within a class method unless they are being called on an instance of that class. |
| Class methods are methods which are declared as static. The method can be called without creating an  instance of the class. | Instance methods are not declared as static. |

**37.How are this() and super() used with constructors?**

* Constructors use *this* to refer to another constructor in the same class with a different parameter list.
* Constructors use *super* to invoke the superclass's constructor. If a constructor uses super, it must use it in the first line; otherwise, the compiler will complain.

**38.What are Access Specifiers?**

One of the techniques in object-oriented programming is *encapsulation*. It concerns the hiding of data in a class and making this class available only through methods. Java allows you to control access to classes, methods, and fields via so-called *access specifiers*..

**39.What are Access Specifiers available in Java?**

Java offers four access specifiers, listed below in decreasing accessibility:

* **Public**- *public* classes, methods, and fields can be accessed from everywhere.
* **Protected**- *protected* methods and fields can only be accessed within the same class to which the methods and fields belong, within its subclasses, and within classes of the same package.
* **Default(no specifier)-**If you do not set access to specific level, then such a class, method, or field will be accessible from inside the same package to which the class, method, or field belongs, but not from outside this package.
* **Private**- *private* methods and fields can only be accessed within the same class to which the methods and fields belong. *private* methods and fields are not visible within subclasses and are not inherited by subclasses.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Situation** | public | protected | **default** | private |
| Accessible to class   from same package? | yes | yes | yes | no |
| Accessible to class   from different package? | yes | no, *unless it is a subclass* | no | no |

**40.What is final modifier?**

The final modifier keyword makes that the programmer cannot change the value anymore. The actual meaning depends on whether it is applied to a class, a variable, or a method.

* ***final* Classes**- A final class cannot have subclasses.
* ***final* Variables**- A final variable cannot be changed once it is initialized.
* ***final* Methods**- A final method cannot be overridden by subclasses.

|  |
| --- |
|  |

**41.What are the uses of final method?**

There are two reasons for marking a method as final:

* Disallowing subclasses to change the meaning of the method.
* Increasing efficiency by allowing the compiler to turn calls to the method into inline Java code.

**42.What is static block?**

Static block which exactly executed exactly once when the class is first loaded into JVM. Before going to the main method the static block will execute.

**43.What are static variables?**

Variables that have only one copy per class are known as static variables. They are not attached to a particular instance of a class but rather belong to a class as a whole. They are declared by using the static keyword as a modifier.

static type varIdentifier;

where, the name of the variable is varIdentifier and its data type is specified by type.  
**Note**: Static variables that are not explicitly initialized in the code are automatically initialized with a default value. The default value depends on the data type of the variables.

**44.What is the difference between static and non-static variables?**

A static variable is associated with the class as a whole rather than with specific instances of a class. Non-static variables take on unique values with each object instance.

|  |
| --- |
|  |

**45.What are static methods?**

Methods declared with the keyword static as modifier are called static methods or class methods. They are so called because they affect a class as a whole, not a particular instance of the class. Static methods are always invoked without reference to a particular instance of a class.  
**Note**:The use of a static method suffers from the following restrictions:

* *A static method can only call other static methods.*
* *A static method must only access static data.*
* *A static method****cannot****reference to the current object using keywords super or this.*

**46.What is an Iterator ?**

* The Iterator interface is used to step through the elements of a Collection.
* Iterators let you process each element of a Collection.
* Iterators are a generic way to go through all the elements of a Collection no matter how it is organized.
* Iterator is an Interface implemented a different way for every Collection.

**47.How do you traverse through a collection using its Iterator?**

To use an iterator to traverse through the contents of a collection, follow these steps:

* Obtain an iterator to the start of the collection by calling the collectionâ€™s ***iterator()*** method.
* Set up a loop that makes a call to ***hasNext()***. Have the loop iterate as long as ***hasNext()*** returns **true**.
* Within the loop, obtain each element by calling **next()**.

**48.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.

**49.What is the difference between Enumeration and Iterator?**

|  |  |
| --- | --- |
| **Enumeration** | **Iterator** |
| Enumeration doesn't have a remove() method | Iterator has a remove() method |
| Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects | Can be *abstract, final, native, static*, or *synchronized* |

**Note**: So Enumeration is used whenever we want to make Collection objects as Read-only.

**50.How is ListIterator?**

**ListIterator** is just like Iterator, except it allows us to access the collection in either the forward or backward direction and lets us modify an element

**51.What is the List interface?**

* The List interface provides support for ordered collections of objects.
* Lists may contain duplicate elements.

**52.What are the main implementations of the List interface ?**

The main implementations of the List interface are as follows :

* **ArrayList** : Resizable-array implementation of the List interface. The best all-around implementation of the List interface.
* **Vector** : Synchronized resizable-array implementation of the List interface with additional "legacy methods."
* **LinkedList** : Doubly-linked list implementation of the List interface. May provide better performance than the ArrayList implementation if elements are frequently inserted or deleted within the list. Useful for queues and double-ended queues (deques).

**53.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.

**54.Difference between ArrayList and Vector ?**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| ArrayList is **NOT** synchronized by default. | Vector List is synchronized by default. |
| ArrayList can use only Iterator to access the elements. | Vector list can use Iterator and Enumeration Interface to access the elements. |
| The ArrayList increases its array size by 50 percent if it runs out of room. | A Vector defaults to doubling the size of its array if it runs out of room |
| ArrayList has no default size. | While vector has a default size of 10. |

**55.How to obtain Array from an ArrayList ?**

Array can be obtained from an ArrayList using ***toArray()***method on ArrayList.

List arrayList = new ArrayList();  
 arrayList.add(â€¦

ObjectÂ  a[] = **arrayList.toArray()**;

**56.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.

|  |
| --- |
|  |

**57.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.

**58.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.

**59.What is the Set interface ?**

* The Set interface provides methods for accessing the elements of a finite mathematical set
* Sets do not allow duplicate elements
* Contains no methods other than those inherited from Collection
* It adds the restriction that duplicate elements are prohibited
* Two Set objects are equal if they contain the same elements

**60.What are the main Implementations of the Set interface ?**

The main implementations of the List interface are as follows:

* HashSet
* TreeSet
* LinkedHashSet
* EnumSet

**61.What is a HashSet ?**

* A HashSet is an unsorted, unordered Set.
* It uses the hashcode of the object being inserted (so the more efficient your hashcode() implementation the better access performance you’ll get).
* Use this class when you want a collection with no duplicates and you don’t care about order when you iterate through it.

**62.What is a TreeSet ?**

TreeSet is a Set implementation that keeps the elements in sorted order. The elements are sorted according to the natural order of elements or by the comparator provided at creation time.

**63.What is an EnumSet ?**

An EnumSet is a specialized set for use with enum types, all of the elements in the EnumSet type that is specified, explicitly or implicitly, when the set is created.

**64.Difference between HashSet and TreeSet ?**

|  |  |
| --- | --- |
| **HashSet** | **TreeSet** |
| HashSet is under set interface i.e. it  does not guarantee for either sorted order or sequence order. | TreeSet is under set i.e. it provides elements in a sorted  order (acceding order). |
| We can add any type of elements to hash set. | We can add only similar types  of elements to tree set. |

|  |
| --- |
|  |

**65.What is a Map ?**

* A map is an object that stores associations between keys and values (key/value pairs).
* Given a key, you can find its value. Both keys  and  values are objects.
* The keys must be unique, but the values may be duplicated.
* Some maps can accept a null key and null values, others cannot.

**66.What are the main Implementations of the Map interface ?**

The main implementations of the List interface are as follows:

* HashMap
* HashTable
* TreeMap
* EnumMap

**67.What is a TreeMap ?**

TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. TreeMap is based on the Red-Black tree data structure.

**68.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.

**69.Difference between HashMap and Hashtable ?**

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| HashMap lets you have null values as well as one null key. | HashTable  does not allows null values as key and value. |
| The iterator in the HashMap is fail-safe (If you change the map while iterating, you’ll know). | The enumerator for the Hashtable is not fail-safe. |
| HashMap is unsynchronized. | Hashtable is synchronized. |

**Note**: Only one NULL is allowed as a key in HashMap. HashMap does not allow multiple keys to be NULL. Nevertheless, it can have multiple NULL values.

**70.How does a Hashtable internally maintain the key-value pairs?**

TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. TreeMap is based on the Red-Black tree data structure.

|  |
| --- |
|  |

**71.What Are the different Collection Views That Maps Provide?**

Maps Provide Three Collection Views.

* **Key Set**- allow a map's contents to be viewed as a set of keys.
* **Values Collection** - allow a map's contents to be viewed as a set of values.
* **Entry Set** - allow a map's contents to be viewed as a set of key-value mappings.

**72.What is a KeySet View ?**

KeySet is a set returned by the ***keySet()*** method of the Map Interface, It is a set that contains all the keys present in the Map.

**73.What is a Values Collection View ?**

Values Collection View is a collection returned by the ***values()*** method of the Map Interface, It contains all the objects present as values in the map.

**74.What is an EntrySet View ?**

Entry Set view is a set that is returned by the ***entrySet()*** method in the map and contains Objects of type Map. Entry each of which has both Key and Value.

**75.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).

**76.What is the Comparable interface ?**

The Comparable interface is used to sort collections and arrays of objects using the Collections.sort() and java.utils.Arrays.sort() methods respectively. The objects of the class implementing the Comparable interface can be ordered.

The Comparable interface in the generic form is written as follows:

interface Comparable<T>

*where T is the name of the type parameter.*  
  
All classes implementing the Comparable interface must implement the compareTo() method that has the return type as an integer. The signature of thecompareTo() method is as follows:

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.

**77.What are the differences between the Comparable and Comparator interfaces ?**

|  |  |
| --- | --- |
| **Comparable** | **Comparato** |
| It uses the *compareTo()* method.  *int objectOne.compareTo(objectTwo).* | t uses the *compare()*method.  *int compare(ObjOne, ObjTwo)* |
| It is necessary to modify the class whose instance is going to be sorted. | A separate class can be created in order to sort the instances. |
| Only one sort sequence can be created. | Many sort sequences can be created. |
| It is frequently used by the API classes. | It used by third-party classes to sort instances. |

I/o Questions:

# Program: All file operations.

|  |
| --- |
| **Description:** |
| Below example shows basic operations on File object.  canRead(): Tests whether the application can read the file  canWrite(): Tests whether the application can modify the file  createNewFile(): Tests whether the application can modify the file  delete(): Deletes the file or directory  exists(): Tests whether the file or directory exists.  getAbsolutePath(): Returns the absolute pathname string.  isDirectory(): Tests whether the file is a directory or not.  isHidden(): Tests whether the file is a hidden file or not.  list(): Returns an array of strings naming the  files and directories in the directory. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.File;  public class MyFileOperations {  public static void main(String[] a){    try{  File file = new File("fileName");  //Tests whether the application can read the file  System.out.println(file.canRead());  //Tests whether the application can modify the file  System.out.println(file.canWrite());  //Tests whether the application can modify the file  System.out.println(file.createNewFile());  //Deletes the file or directory  System.out.println(file.delete());  //Tests whether the file or directory exists.  System.out.println(file.exists());  //Returns the absolute pathname string.  System.out.println(file.getAbsolutePath());  //Tests whether the file is a directory or not.  System.out.println(file.isDirectory());  //Tests whether the file is a hidden file or not.  System.out.println(file.isHidden());  //Returns an array of strings naming the  //files and directories in the directory.  System.out.println(file.list());  } catch(Exception ex){    }  }  } |

# How to get list of all file names from a folder in java?

|  |
| --- |
| **Description:** |
| Below example shows how to get list of all file names from the given folder. First create File object by passing folder path. Call list() method on file object to get list of file names in the folder. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.File;  public class FileListFromFolder {    public static void main(String a[]){  File file = new File("C:/MyFolder/");  String[] fileList = file.list();  for(String name:fileList){  System.out.println(name);  }  }  } |

# How to get list of all file objects from a folder in java?

|  |
| --- |
| **Description:** |
| Below example shows how to get list of all file objects from the given folder. First create File object by passing folder path. Call listFiles() method on file object to get list of file names in the given folder. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.File;  public class FilesListFromFolder {    public static void main(String a[]){  File file = new File("C:/MyFolder/");  File[] files = file.listFiles();  for(File f: files){  System.out.println(f.getName());  }  }  } |

# How to filter the files by file extensions and show the file names?

|  |
| --- |
| **Description:** |
| Below example shows how to get specific files from a folder. Sometimes we need to pic only specific extensions from the given folder. Implement FilenameFilter class and override accept() method, and add your filter logic here. Pass this object to list() method to get specific file extensions. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.File;  import java.io.FilenameFilter;  public class MyFileFilter {  public static void main(String a[]){  File file = new File("C:/MyFolder/");  String[] files = file.list(new FilenameFilter() {    @Override  public boolean accept(File dir, String name) {  if(name.toLowerCase().endsWith(".csv")){  return true;  } else {  return false;  }  }  });  for(String f:files){  System.out.println(f);  }  }  } |

# How to read file content using byte array?

|  |
| --- |
| **Description:** |
| Write a program to read a file using byte array. You need to read fixed size of content each time, read multiple times, till end of the file. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.IOException;  import java.io.InputStream;  /\*\*  \* Reading contents from a file into byte array.  \*  \*/  public class FileToByteArray {  public static void main(String a[]){    String fileName = "C:/MyFile.txt";  InputStream is = null;  try {  is = new FileInputStream(fileName);  byte content[] = new byte[2\*1024];  int readCount = 0;  while((readCount = is.read(content)) > 0){  System.out.println(new String(content, 0, readCount-1));  }  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  } finally {  try{  if(is != null) is.close();  } catch(Exception ex){    }  }  }  } |

# ow to read file content line by line in java?

|  |
| --- |
| **Description:** |
| Below example shows how to read file content line by line. To get this, you have to use BufferedReader object. By calling readLine() method you can get file content line by line. readLine() returns one line at each iteration, we have to iterate it till it returns null. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.BufferedReader;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  public class ReadLinesFromFile {  public static void main(String a[]){  BufferedReader br = null;  String strLine = "";  try {  br = new BufferedReader( new FileReader("fileName"));  while( (strLine = br.readLine()) != null){  System.out.println(strLine);  }  } catch (FileNotFoundException e) {  System.err.println("Unable to find the file: fileName");  } catch (IOException e) {  System.err.println("Unable to read the file: fileName");  }  }  } |

# Program: How to read property file in static context?

|  |
| --- |
| **Description:** |
| Below example shows how to read any file (property file, image file, etc) from a static method, or static block using getResourceAsStream() method. You can read any file in this way if it is in your class path. Make sure that the rendering file is in the class path. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.IOException;  import java.io.InputStream;  import java.util.Properties;  public class ClassLoaderFileLoading {  private static Properties appProp = null;    static {  try{  InputStream is =  ClassLoaderFileLoading.class.getResourceAsStream("/FileName.properties");  appProp = new Properties();  appProp.load(is);  } catch(IOException ex){  ex.printStackTrace();  } catch(Exception ex){  ex.printStackTrace();  }  }  } |

# How to read input from java console in java?

|  |
| --- |
| **Description:** |
| Below example shows how to read input from java console. You have to pass System.in object to InputStreamReader class. Create BufferedReader object by passing InputStreamReader, readLine() method can helps you to get the typed commands. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.Reader;  public class ReadFromConsole {  public static void main(String a[]){  BufferedReader br = null;  Reader r = new InputStreamReader(System.in);  br = new BufferedReader(r);  String str = null;  try {  do{  System.out.println("Enter Input, exit to quit.");  str = br.readLine();  System.out.println(str);  } while (!str.equalsIgnoreCase("exit"));  } catch (IOException e) {  e.printStackTrace();  } finally{  try{  if(br != null) br.close();  }catch(Exception ex){}  }  }  } |

|  |
| --- |
| **Output:** |
| Enter Input, exit to quit.  Hello  Hello  Enter Input, exit to quit.  I am entering data through console  I am entering data through console  Enter Input, exit to quit.  exit  exit |

# Program: How to get file URI reference?

|  |
| --- |
| **Description:** |
| Below example shows how to get URI reference to the given file. By calling toURI() method on file object, you can get the URI reference. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.File;  public class MyFileUrl {    public static void main(String a[]){  File f = new File("C:/TestForm.txt");  System.out.println(f.toURI());  }  } |

# How to create and store property file dynamically?

|  |
| --- |
| **Description:** |
| We can also create property files dynamically. First creat Properties object, add all property key-value pair by using setProperty() method. Then write this file to FileOutputStream object to store is as a property file. Below example shows how to create it. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.OutputStream;  import java.util.Properties;  public class MyPropertyFileStore {  public static void main(String a[]) throws IOException{    OutputStream os = null;  Properties prop = new Properties();  prop.setProperty("name", "java2novice");  prop.setProperty("domain", "www.java2novice.com");  prop.setProperty("email", "java2novice@gmail.com");  try {  os = new FileOutputStream("MyProp.properties");  prop.store(os, "Dynamic Property File");  } catch (FileNotFoundException e) {  e.printStackTrace();  }    }  } |

# Program: How to store property file as xml file?

|  |
| --- |
| **Description:** |
| We can also create property files in xml format dynamically. First creat Properties object, add all property key-value pair by using setProperty() method. Then write this file to FileOutputStream object to store is as a property file as xml. Call storeToXML() method to store it to the file system. Below example shows how to create it. |

|  |
| --- |
| **Code:** |
| package com.java2novice.files;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.OutputStream;  import java.util.Properties;  public class MyFileXmlStore {    public static void main(String a[]) throws IOException{    OutputStream os = null;  Properties prop = new Properties();  prop.setProperty("name", "java2novice");  prop.setProperty("domain", "www.java2novice.com");  prop.setProperty("email", "java2novice@gmail.com");  try {  os = new FileOutputStream("MyProp.xml");  prop.storeToXML(os, "Dynamic Property File");  } catch (FileNotFoundException e) {  e.printStackTrace();  }    }  } |

|  |
| --- |
| **Output:** |
| <?xml version="1.0" encoding="UTF-8" standalone="no"?>  <properties>  <comment>Dynamic Property File</comment>  <entry key="email">java2novice@gmail.com</entry>  <entry key="name">java2novice</entry>  <entry key="domain">www.java2novice.com</entry>  </properties> |

|  |
| --- |
|  |
| Program: How to get file last modified time?  |  | | --- | | **Description:** | | Below example shows how to get last modified time of a file. lastModified() method gives you the last modified time of a file object. |  |  | | --- | | **Code:** | | package com.java2novice.files;  import java.io.File;  public class MyFileLastModified {    public static void main(String a[]) {    File file = new File("MyProp.xml");  System.out.println(file.lastModified());  }  } | |

- See more at: <http://www.java2novice.com/java-file-io-operations/file-last-modified-time/#sthash.NYC6rE33.dpuf>

# Program: How to set file permissions in java?

Below example shows how to set permissions to a file.

|  |
| --- |
| package com.java2novice.files;  import java.io.File;  public class MyFilePermissions {  public static void main(String a[]){    File scriptFile = new File("/home/java2novice/test.sh");  System.out.println("Current file permissions:");  System.out.println("Can Execute? "+scriptFile.canExecute());  System.out.println("Can Read? "+scriptFile.canRead());  System.out.println("Can Write? "+scriptFile.canWrite());  scriptFile.setExecutable(true);  scriptFile.setReadable(true);  scriptFile.setWritable(true);  System.out.println("Now file permissions:");  System.out.println("Can Execute? "+scriptFile.canExecute());  System.out.println("Can Read? "+scriptFile.canRead());  System.out.println("Can Write? "+scriptFile.canWrite());  }  } |

|  |
| --- |
| **Output:** |
| Current file permissions:  Can Execute? false  Can Read? true  Can Write? false  Now file permissions:  Can Execute? true  Can Read? true  Can Write? true |

- See more at: <http://www.java2novice.com/java-file-io-operations/set-file-permissions/#sthash.3gMan5t0.dpuf>

# Program: How to write string content to a file in java?

Below example shows how write string content to a local file. When you are dealing with characters, you need to use Writer.

- See more at: http://www.java2novice.com/java-file-io-operations/write-string-to-file/#sthash.lVyEKm0j.dpuf

package com.java2novice.files;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.Writer;

public class WriteToFile {

    public static void main(String[] args) {

        BufferedWriter bufferedWriter = null;

        try {

            String strContent = "This example shows how to write string content to a file";

            File myFile = new File("C:/MyTestFile.txt");

            // check if file exist, otherwise create the file before writing

            if (!myFile.exists()) {

                myFile.createNewFile();

            }

            Writer writer = new FileWriter(myFile);

            bufferedWriter = new BufferedWriter(writer);

            bufferedWriter.write(strContent);

        } catch (IOException e) {

            e.printStackTrace();

        } finally{

            try{

                if(bufferedWriter != null) bufferedWriter.close();

            } catch(Exception ex){

            }

        }

    }

}

Hibernate

**1) What is hibernate?**

Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate and retrieve data from the database.

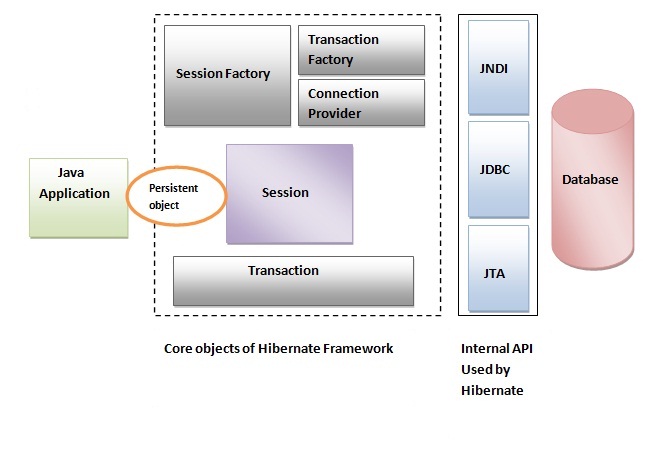
[more details...](http://www.javatpoint.com/hibernate-tutorial)

### 2) What is ORM?

ORM is an acronym for Object/Relational mapping. It is a programming strategy to map object with the data stored in the database. It simplifies data creation, data manipulation and data access.

### 3) Explain hibernate architecture?

Hibernate architecture comprises of many interfaces such as Configuration, SessionFactory, Session, Transaction etc.

[more details...](http://www.javatpoint.com/hibernate-architecture) 

### 4) What are the core interfaces of Hibernate?

The core interfaces of Hibernate framework are:

* Configuration
* SessionFactory
* Session
* Query
* Criteria
* Transaction

### 5) What is SessionFactory?

SessionFactory provides the instance of Session. It is a factory of Session. It holds the data of second level cache that is not enabled by default.

[more details...](http://www.javatpoint.com/hibernate-architecture)

### 6) Is SessionFactory a thread-safe object?

Yes, SessionFactory is a thread-safe object, many threads can access it simultaneously.

### 7) What is Session?

It maintains a connection between hibernate application and database.

It provides methods to store, update, delete or fetch data from the database such as persist(), update(), delete(), load(), get() etc.

It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances.

[more details...](http://www.javatpoint.com/hibernate-architecture)

### 8) Is Session a thread-safe object?

No, Session is not a thread-safe object, many threads can't access it simultaneously. In other words, you cannot share it between threads.

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

|  |  |  |
| --- | --- | --- |
| **No.** | **save()** | **persist()** |
| 1) | returns the identifier (Serializable) of the instance. | return nothing because its return type is void. |
| 2) | Syn: public Serializable save(Object o) | Syn: public void persist(Object o) |

### 10) What is the difference between get and load method?

The differences between get() and load() methods are given below.

|  |  |  |
| --- | --- | --- |
| **No.** | **get()** | **load()** |
| 1) | Returns **null** if object is not found. | Throws **ObjectNotFoundException** if object is not found. |
| 2) | get() method always **hit the database**. | load() method **doesn't hit** the database. |
| 3) | It returns real object **not proxy**. | It returns **proxy object.** |
| 4) | It should be used if **you are not sure** about the existence of instance. | It should be used if **you are sure** that instance exists. |

### 11) What is the difference between update and merge method?

The differences between update() and merge() methods are given below.

|  |  |  |
| --- | --- | --- |
| **No.** | **update() method** | **merge() method** |
| 1) | Update means to edit something. | Merge means to combine something. |
| 2) | update() should be used if session doesn't contain an already persistent state with same id. It means update should be used inside the session only. After closing the session it will throw error. | merge() should be used if you don't know the state of the session, means you want to make modification at any time. |

Let's try to understand the difference by the example given below:

1. ...
2. SessionFactory factory = cfg.buildSessionFactory();
3. Session session1 = factory.openSession();
5. Employee e1 = (Employee) session1.get(Employee.**class**, Integer.valueOf(101));//passing id of employee
6. session1.close();
8. e1.setSalary(70000);
10. Session session2 = factory.openSession();
11. Employee e2 = (Employee) session1.get(Employee.**class**, Integer.valueOf(101));//passing same id
13. Transaction tx=session2.beginTransaction();
14. session2.merge(e1);
16. tx.commit();
17. session2.close();

After closing session1, e1 is in detached state. It will not be in session1 cache. So if you call update() method, it will throw an error.

Then, we opened another session and loaded the same Employee instance. If we call merge in session2, changes of e1 will be merged in e2.

### 12) What are the states of object in hibernate?

There are 3 states of object (instance) in hibernate.

1. **Transient**: The object is in transient state if it is just created but has no primary key (identifier) and not associated with session.
2. **Persistent**: The object is in persistent state if session is open, and you just saved the instance in the database or retrieved the instance from the database.
3. **Detached**: The object is in detached state if session is closed. After detached state, object comes to persistent state if you call lock() or update() method.

### 13) What are the inheritance mapping strategies?

There are 3 ways of inheritance mapping in hibernate.

1. Table per hierarchy
2. Table per concrete class
3. Table per subclass

[more details...](http://www.javatpoint.com/hibernate-inheritance-mapping-tutorial)

### 14) How to make a immutable class in hibernate?

If you mark a class as mutable="false", class will be treated as an immutable class. By default, it is mutable="true".

### 15) What is automatic dirty checking in hibernate?

The automatic dirty checking feature of hibernate, calls update statement automatically on the objects that are modified in a transaction.

Let's understand it by the example given below:

1. ...
2. SessionFactory factory = cfg.buildSessionFactory();
3. Session session1 = factory.openSession();
4. Transaction tx=session2.beginTransaction();
6. Employee e1 = (Employee) session1.get(Employee.**class**, Integer.valueOf(101));
8. e1.setSalary(70000);
10. tx.commit();
11. session1.close();

Here, after getting employee instance e1 and we are changing the state of e1.

After changing the state, we are committing the transaction. In such case, state will be updated automatically. This is known as dirty checking in hibernate.

### 16) How many types of association mapping are possible in hibernate?

There can be 4 types of association mapping in hibernate.

1. One to One
2. One to Many
3. Many to One
4. Many to Many

### 17) Is it possible to perform collection mapping with One-to-One and Many-to-One?

No, collection mapping can only be performed with One-to-Many and Many-to-Many

### 18) What is lazy loading in hibernate?

Lazy loading in hibernate improves the performance. It loads the child objects on demand.

Since Hibernate 3, lazy loading is enabled by default, you don't need to do lazy="true". It means not to load the child objects when parent is loaded.

### 19) What is HQL (Hibernate Query Language)?

Hibernate Query Language is known as an object oriented query language. It is like structured query language (SQL).

The main advantage of HQL over SQL is:

1. You don't need to learn SQL
2. Database independent
3. Simple to write query

### 20) What is the difference between first level cache and second level cache?

|  |  |  |
| --- | --- | --- |
| **No.** | **First Level Cache** | **Second Level Cache** |
| 1) | First Level Cache is **associated with Session**. | Second Level Cache is associated with **SessionFactory**. |
| 2) | It is **enabled** by default. | It is **not enabled** by default. |

### What is Hibernate Framework?

**Object-relational mapping** or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.

Hibernate provides reference implementation of Java Persistence API, that makes it a great choice as ORM tool with benefits of loose coupling. We can use Hibernate persistence API for CRUD operations. Hibernate framework provide option to map plain old java objects to traditional database tables with the use of JPA annotations as well as XML based configuration.

Similarly hibernate configurations are flexible and can be done from XML configuration file as well as programmatically. For a quick overview of hibernate framework usage, you can go through [Hibernate Beginners Tutorial](http://www.journaldev.com/2882/hibernate-tutorial-for-beginners-using-xml-annotations-and-property-configurations).

### What is Java Persistence API (JPA)?

Java Persistence API (JPA) provides specification for managing the relational data in applications. Current JPA version 2.1 was started in July 2011 as JSR 338. JPA 2.1 was approved as final on 22 May 2013.

JPA specifications is defined with annotations in javax.persistence package. Using JPA annotation helps us in writing implementation independent code.

### What are the important benefits of using Hibernate Framework?

Some of the important benefits of using hibernate framework are:

* 1. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business logic.
  2. Hibernate framework provides support for XML as well as JPA annotations, that makes our code implementation independent.
  3. Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism and association.
  4. Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentations and help is easily available in forums.
  5. Hibernate is easy to integrate with other Java EE frameworks, it’s so popular that Spring Framework provides built-in support for integrating hibernate with Spring applications.
  6. Hibernate supports lazy initialization using proxy objects and perform actual database queries only when it’s required.
  7. Hibernate cache helps us in getting better performance.
  8. For database vendor specific feature, hibernate is suitable because we can also execute native sql queries.

Overall hibernate is the best choice in current market for ORM tool, it contains all the features that you will ever need in an ORM tool.

### What are the advantages of Hibernate over JDBC?

Some of the important advantages of Hibernate framework over JDBC are:

* 1. Hibernate removes a lot of boiler-plate code that comes with JDBC API, the code looks more cleaner and readable.
  2. Hibernate supports inheritance, associations and collections. These features are not present with JDBC API.
  3. Hibernate implicitly provides transaction management, in fact most of the queries can’t be executed outside transaction. In JDBC API, we need to write code for transaction management using commit and rollback. Read more at [JDBC Transaction Management](http://www.journaldev.com/2483/jdbc-transaction-management-and-savepoint-example-tutorial).
  4. JDBC API throws SQLException that is a checked exception, so we need to write a lot of try-catch block code. Most of the times it’s redundant in every JDBC call and used for transaction management. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException un-checked exception, so we don’t need to write code to handle it. Hibernate built-in transaction management removes the usage of try-catch blocks.
  5. Hibernate Query Language (HQL) is more object oriented and close to java programming language. For JDBC, we need to write native sql queries.
  6. Hibernate supports caching that is better for performance, JDBC queries are not cached hence performance is low.
  7. Hibernate provide option through which we can create database tables too, for JDBC tables must exist in the database.
  8. Hibernate configuration helps us in using JDBC like connection as well as JNDI DataSource for connection pool. This is very important feature in enterprise application and completely missing in JDBC API.
  9. Hibernate supports JPA annotations, so code is independent of implementation and easily replaceable with other ORM tools. JDBC code is very tightly coupled with the application.

### Name some important interfaces of Hibernate framework?

Some of the important interfaces of Hibernate framework are:

* 1. **SessionFactory (org.hibernate.SessionFactory)**: SessionFactory is an immutable thread-safe cache of compiled mappings for a single database. We need to initialize SessionFactory once and then we can cache and reuse it. SessionFactory instance is used to get the Session objects for database operations.
  2. **Session (org.hibernate.Session)**: Session is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It wraps JDBC java.sql.Connectionand works as a factory for org.hibernate.Transaction. We should open session only when it’s required and close it as soon as we are done using it. Session object is the interface between java application code and hibernate framework and provide methods for CRUD operations.
  3. **Transaction (org.hibernate.Transaction)**: Transaction is a single-threaded, short-lived object used by the application to specify atomic units of work. It abstracts the application from the underlying JDBC or JTA transaction. A org.hibernate.Session might span multiple org.hibernate.Transaction in some cases.

### What is hibernate configuration file?

Hibernate configuration file contains database specific configurations and used to initialize SessionFactory. We provide database credentials or JNDI resource information in the hibernate configuration xml file. Some other important parts of hibernate configuration file is Dialect information, so that hibernate knows the database type and mapping file or class details.

### What is hibernate mapping file?

Hibernate mapping file is used to define the entity bean fields and database table column mappings. We know that JPA annotations can be used for mapping but sometimes XML mapping file comes handy when we are using third party classes and we can’t use annotations.

### Name some important annotations used for Hibernate mapping?

Hibernate supports JPA annotations and it has some other annotations in org.hibernate.annotationspackage. Some of the important JPA and hibernate annotations used are:

* 1. **javax.persistence.Entity**: Used with model classes to specify that they are entity beans.
  2. **javax.persistence.Table**: Used with entity beans to define the corresponding table name in database.
  3. **javax.persistence.Access**: Used to define the access type, either field or property. Default value is field and if you want hibernate to use getter/setter methods then you need to set it to property.
  4. **javax.persistence.Id**: Used to define the primary key in the entity bean.
  5. **javax.persistence.EmbeddedId**: Used to define composite primary key in the entity bean.
  6. **javax.persistence.Column**: Used to define the column name in database table.
  7. **javax.persistence.GeneratedValue**: Used to define the strategy to be used for generation of primary key. Used in conjunction with javax.persistence.GenerationType enum.
  8. **javax.persistence.OneToOne**: Used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany
  9. **org.hibernate.annotations.Cascade**: Used to define the cascading between two entity beans, used with mappings. It works in conjunction with org.hibernate.annotations.CascadeType
  10. **javax.persistence.PrimaryKeyJoinColumn**: Used to define the property for foreign key. Used withorg.hibernate.annotations.GenericGenerator and org.hibernate.annotations.Parameter

Here are two classes showing usage of these annotations.

|  |  |
| --- | --- |
| Employee.java | |
| 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  33 | package com.journaldev.hibernate.model;    import javax.persistence.Access;  import javax.persistence.AccessType;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.OneToOne;  import javax.persistence.Table;    import org.hibernate.annotations.Cascade;    @Entity  @Table(name = "EMPLOYEE")  @Access(value=AccessType.FIELD)  public class Employee {        @Id      @GeneratedValue(strategy = GenerationType.IDENTITY)      @Column(name = "emp\_id")      private long id;        @Column(name = "emp\_name")      private String name;        @OneToOne(mappedBy = "employee")      @Cascade(value = org.hibernate.annotations.CascadeType.ALL)      private Address address;        //getter setter methods  } |
| Address.java | | |
| 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  33  34  35 | package com.journaldev.hibernate.model;    import javax.persistence.Access;  import javax.persistence.AccessType;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.Id;  import javax.persistence.OneToOne;  import javax.persistence.PrimaryKeyJoinColumn;  import javax.persistence.Table;    import org.hibernate.annotations.GenericGenerator;  import org.hibernate.annotations.Parameter;    @Entity  @Table(name = "ADDRESS")  @Access(value=AccessType.FIELD)  public class Address {        @Id      @Column(name = "emp\_id", unique = true, nullable = false)      @GeneratedValue(generator = "gen")      @GenericGenerator(name = "gen", strategy = "foreign", parameters = { @Parameter(name = "property", value = "employee") })      private long id;        @Column(name = "address\_line1")      private String addressLine1;        @OneToOne      @PrimaryKeyJoinColumn      private Employee employee;        //getter setter methods  } | |

### What is Hibernate SessionFactory and how to configure it?

SessionFactory is the factory class used to get the Session objects. SessionFactory is responsible to read the hibernate configuration parameters and connect to the database and provide Session objects. Usually an application has a single SessionFactory instance and threads servicing client requests obtain Session instances from this factory.

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.

SessionFactory also provide methods to get the Class metadata and Statistics instance to get the stats of query executions, second level cache details etc.

### Hibernate SessionFactory is thread safe?

Internal state of SessionFactory is immutable, so it’s thread safe. Multiple threads can access it simultaneously to get Session instances.

### What is Hibernate Session and how to get it?

Hibernate Session is the interface between java application layer and hibernate. This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.

Session provide methods to perform create, read, update and delete operations for a persistent object. We can execute HQL queries, SQL native queries and create criteria using Session object.

### Hibernate Session is thread safe?

Hibernate Session object is not thread safe, every thread should get it’s own session instance and close it after it’s work is finished.

### 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.

|  |  |
| --- | --- |
| 1 | <property name="hibernate.current\_session\_context\_class">thread</property> |

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.

There is another method openStatelessSession() that returns stateless session, for more details with examples please read [Hibernate openSession vs getCurrentSession](http://www.journaldev.com/3522/hibernate-sessionfactory-opensession-vs-getcurrentsession-vs-openstatelesssession).

### What is difference between Hibernate Session get() and load() method?

Hibernate session comes with different methods to load data from database. get and load are most used methods, at first look they seems similar but there are some differences between them.

* 1. get() loads the data as soon as it’s called whereas load() returns a proxy object and loads data only when it’s actually required, so load() is better because it support lazy loading.
  2. Since load() throws exception when data is not found, we should use it only when we know data exists.
  3. We should use get() when we want to make sure data exists in the database.

For clarification regarding the differences, please read [Hibernate get vs load](http://www.journaldev.com/3472/hibernate-session-get-vs-load-difference-with-examples).

### What is hibernate caching? Explain Hibernate first level cache?

As the name suggests, hibernate caches query data to make our application faster. Hibernate Cache can be very useful in gaining fast application performance if used correctly. The idea behind cache is to reduce the number of database queries, hence reducing the throughput time of the application.

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.  
Any object cached in a session will not be visible to other sessions and when the session is closed, all the cached objects will also be lost.

For better explanation, please read [Hibernate First Level Cache](http://www.journaldev.com/2969/hibernate-first-level-cache-example-with-explanation).

### How to configure Hibernate Second Level Cache using EHCache?

EHCache is the best choice for utilizing hibernate second level cache. Following steps are required to enable EHCache in hibernate application.

* 1. Add hibernate-ehcache dependency in your maven project, if it’s not maven then add corresponding jars.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>          <groupId>org.hibernate</groupId>          <artifactId>hibernate-ehcache</artifactId>          <version>4.3.5.Final</version>  </dependency> |

* 1. Add below properties in hibernate configuration file.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <property name="hibernate.cache.region.factory\_class">org.hibernate.cache.ehcache.EhCacheRegionFactory</property>    <!-- For singleton factory -->  <!-- <property name="hibernate.cache.region.factory\_class">org.hibernate.cache.ehcache.SingletonEhCacheRegionFactory</property>  -->    <!-- enable second level cache and query cache -->  <property name="hibernate.cache.use\_second\_level\_cache">true</property>  <property name="hibernate.cache.use\_query\_cache">true</property>  <property name="net.sf.ehcache.configurationResourceName">/myehcache.xml</property> |

* 1. Create EHCache configuration file, a sample file would look like below.

|  |  |
| --- | --- |
| myehcache.xml | |
| 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 | <?xml version="1.0" encoding="UTF-8"?>  <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:noNamespaceSchemaLocation="ehcache.xsd" updateCheck="true"      monitoring="autodetect" dynamicConfig="true">        <diskStore path="java.io.tmpdir/ehcache" />        <defaultCache maxEntriesLocalHeap="10000" eternal="false"          timeToIdleSeconds="120" timeToLiveSeconds="120" diskSpoolBufferSizeMB="30"          maxEntriesLocalDisk="10000000" diskExpiryThreadIntervalSeconds="120"          memoryStoreEvictionPolicy="LRU" statistics="true">          <persistence strategy="localTempSwap" />      </defaultCache>        <cache name="employee" maxEntriesLocalHeap="10000" eternal="false"          timeToIdleSeconds="5" timeToLiveSeconds="10">          <persistence strategy="localTempSwap" />      </cache>        <cache name="org.hibernate.cache.internal.StandardQueryCache"          maxEntriesLocalHeap="5" eternal="false" timeToLiveSeconds="120">          <persistence strategy="localTempSwap" />      </cache>        <cache name="org.hibernate.cache.spi.UpdateTimestampsCache"          maxEntriesLocalHeap="5000" eternal="true">          <persistence strategy="localTempSwap" />      </cache>  </ehcache> |

* 1. Annotate entity beans with @Cache annotation and caching strategy to use. For example,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import org.hibernate.annotations.Cache;  import org.hibernate.annotations.CacheConcurrencyStrategy;    @Entity  @Table(name = "ADDRESS")  @Cache(usage=CacheConcurrencyStrategy.READ\_ONLY, region="employee")  public class Address {    } |

That’s it, we are done. Hibernate will use the EHCache for second level caching, read [Hibernate EHCache Example](http://www.journaldev.com/2980/hibernate-ehcache-second-level-caching-example-tutorial) for a complete example with explanation.

### What are different states of an entity bean?

An entity bean instance can exist is one of the three states.

* 1. **Transient**: When an object is never persisted or associated with any session, it’s in transient state. Transient instances may be made persistent by calling save(), persist() or saveOrUpdate(). Persistent instances may be made transient by calling delete().
  2. **Persistent**: When an object is associated with a unique session, it’s in persistent state. Any instance returned by a get() or load() method is persistent.
  3. **Detached**: When an object is previously persistent but not associated with any session, it’s in detached state. Detached instances may be made persistent by calling update(), saveOrUpdate(), lock() or replicate(). The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge().

### What is use of Hibernate Session merge() call?

Hibernate merge can be used to update existing values, however this method create a copy from the passed entity object and return it. The returned object is part of persistent context and tracked for any changes, passed object is not tracked. For example program, read [Hibernate merge](http://www.journaldev.com/3481/hibernate-save-vs-saveorupdate-vs-persist-vs-merge-vs-update-explanation-with-examples).

### What is difference between Hibernate save(), saveOrUpdate() and persist() methods?

Hibernate save can be used to save entity to database. Problem with save() is that it can be invoked without a transaction and if we have mapping entities, then only the primary object gets saved causing data inconsistencies. Also save returns the generated id immediately.

Hibernate persist is similar to save with transaction. I feel it’s better than save because we can’t use it outside the boundary of transaction, so all the object mappings are preserved. Also persist doesn’t return the generated id immediately, so data persistence happens when needed.

Hibernate saveOrUpdate results into insert or update queries based on the provided data. If the data is present in the database, update query is executed. We can use saveOrUpdate() without transaction also, but again you will face the issues with mapped objects not getting saved if session is not flushed. For example usage of these methods, read [Hibernate save vs persist](http://www.journaldev.com/3481/hibernate-save-vs-saveorupdate-vs-persist-vs-merge-vs-update-explanation-with-examples).

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

Hibernate uses [Reflection API](http://www.journaldev.com/1789/java-reflection-tutorial-for-classes-methods-fields-constructors-annotations-and-much-more) 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 getHibernateException.

### 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](http://www.journaldev.com/780/java-comparable-and-comparator-example-to-sort-objects).

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.

|  |  |
| --- | --- |
| 1  2 | List<Employee> empList = session.createCriteria(Employee.class)                          .addOrder(Order.desc("id")).list(); |

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.

### What are the collection types in Hibernate?

There are five collection types in hibernate used for one-to-many relationship mappings.

Bag

Set

List

Array

Map

### How to implement Joins in Hibernate?

There are various ways to implement joins in hibernate.

* 1. Using associations such as one-to-one, one-to-many etc.
  2. Using JOIN in the HQL query. There is another form “join fetch” to load associated data simultaneously, no lazy loading.
  3. We can fire native sql query and use join keyword.

### 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.

### What is HQL and what are it’s benefits?

Hibernate Framework comes with a powerful object-oriented query language – Hibernate Query Language (HQL). It’s very similar to SQL except that we use Objects instead of table names, that makes it more close to object oriented programming.

Hibernate query language is case-insensitive except for java class and variable names. So SeLeCT is the same as sELEct is the same as SELECT, but com.journaldev.model.Employee is not same as com.journaldev.model.EMPLOYEE.

The HQL queries are cached but we should avoid it as much as possible, otherwise we will have to take care of associations. However it’s a better choice than native sql query because of Object-Oriented approach. Read more at [HQL Example](http://www.journaldev.com/2954/hibernate-query-language-hql-example-tutorial).

### What is Query Cache in Hibernate?

Hibernate implements a cache region for queries resultset that integrates closely with the hibernate second-level cache.

This is an optional feature and requires additional steps in code. This is only useful for queries that are run frequently with the same parameters. First of all we need to configure below property in hibernate configuration file.

|  |  |
| --- | --- |
|  | <property name="hibernate.cache.use\_query\_cache">true</property> |

And in code, we need to use setCacheable(true) method of Query, quick example looks like below.

|  |  |
| --- | --- |
|  | Query query = session.createQuery("from Employee");  query.setCacheable(true);  query.setCacheRegion("ALL\_EMP"); |

### Can we execute native sql query in hibernate?

Hibernate provide option to execute native SQL queries through the use of SQLQuery object.

For normal scenarios, it is however not the recommended approach because we loose benefits related to hibernate association and hibernate first level caching. Read more at [Hibernate Native SQL Query Example](http://www.journaldev.com/3422/hibernate-native-sql-example-addscalar-addentity-addjoin-parameter-example).

### What is the benefit of native sql query support in hibernate?

Native SQL Query comes handy when we want to execute database specific queries that are not supported by Hibernate API such as query hints or the CONNECT keyword in Oracle Database.

### 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.

### 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.

### What is the benefit of Hibernate Criteria API?

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.

Some of the common usage of Criteria API are:

* 1. Criteria API provides Projection that we can use for aggregate functions such as sum(), min(), max() etc.
  2. Criteria API can be used with ProjectionList to fetch selected columns only.
  3. Criteria API can be used for join queries by joining multiple tables, useful methods are createAlias(), setFetchMode() and setProjection()
  4. Criteria API can be used for fetching results with conditions, useful methods are add() where we can add Restrictions.
  5. Criteria API provides addOrder() method that we can use for ordering the results.

Learn some quick examples at [Hibernate Criteria Example](http://www.journaldev.com/2963/hibernate-criteria-example-tutorial).

### How to log hibernate generated sql queries in log files?

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

|  |  |
| --- | --- |
| 1 | <property name="hibernate.show\_sql">true</property> |

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

### 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.

### 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.

[Hibernate One to One Mapping](http://www.journaldev.com/2916/hibernate-one-to-one-mapping-annotation-and-xml-configuration-example)

[Hibernate One to Many Mapping](http://www.journaldev.com/2924/hibernate-one-to-many-mapping-annotation-and-xml-configuration-example-tutorial)

[Hibernate Many to Many Mapping](http://www.journaldev.com/2934/hibernate-many-to-many-mapping-join-tables-annotation-and-xml-configuration-example)

### 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.

Overall hibernate transaction management is better than JDBC transaction management because we don’t need to rely on exceptions for rollback. Any exception thrown by session methods automatically rollback the transaction.

### What is cascading and what are different types of cascading?

When we have relationship between entities, then we need to define how the different operations will affect the other entity. This is done by cascading and there are different types of it.

Here is a simple example of applying cascading between primary and secondary entities.

|  |  |
| --- | --- |
|  | import org.hibernate.annotations.Cascade;    @Entity  @Table(name = "EMPLOYEE")  public class Employee {    @OneToOne(mappedBy = "employee")  @Cascade(value = org.hibernate.annotations.CascadeType.ALL)  private Address address;    } |

Note that Hibernate CascadeType enum constants are little bit different from JPA javax.persistence.CascadeType, so we need to use the Hibernate CascadeType and Cascade annotations for mappings, as shown in above example.  
Commonly used cascading types as defined in CascadeType enum are:

* 1. None: No Cascading, it’s not a type but when we don’t define any cascading then no operations in parent affects the child.
  2. ALL: Cascades save, delete, update, evict, lock, replicate, merge, persist. Basically everything
  3. SAVE\_UPDATE: Cascades save and update, available only in hibernate.
  4. DELETE: Corresponds to the Hibernate native DELETE action, only in hibernate.
  5. DETATCH, MERGE, PERSIST, REFRESH and REMOVE – for similar operations
  6. LOCK: Corresponds to the Hibernate native LOCK action.
  7. REPLICATE: Corresponds to the Hibernate native REPLICATE action.

### 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.

* 1. Add log4j dependencies for maven project, if not maven then add corresponding jar files.
  2. 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.
  3. For standalone projects, use static block to configure log4j using DOMConfigurator orPropertyConfigurator. 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. For complete example code, you should go through [Hibernate log4j example](http://www.journaldev.com/2984/hibernate-4-log4j-configuration-example) and [Servlet log4j example](http://www.journaldev.com/1997/servlet-example-in-java-with-database-connection-and-log4j-integration).

### 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.

|  |  |
| --- | --- |
| 1 | <property name="hibernate.connection.datasource">java:comp/env/jdbc/MyLocalDB</property> |

For a complete example, go through [Hibernate JNDI DataSource Example](http://www.journaldev.com/2905/hibernate-tomcat-jndi-datasource-example-tutorial).

### How to integrate Hibernate and Spring frameworks?

Spring is one of the most used Java EE Framework and Hibernate is the most popular ORM framework. That’s why Spring Hibernate combination is used a lot in enterprise applications. The best part with using Spring is that it provides out-of-box integration support for Hibernate with **Spring ORM** module. Following steps are required to integrate Spring and Hibernate frameworks together.

* 1. Add hibernate-entitymanager, hibernate-core and spring-orm dependencies.
  2. Create Model classes and corresponding DAO implementations for database operations. Note that DAO classes will use SessionFactory that will be injected by Spring Bean configuration.
  3. If you are using Hibernate 3, you need to configureorg.springframework.orm.hibernate3.LocalSessionFactoryBean ororg.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean in Spring Bean configuration file. For Hibernate 4, there is single classorg.springframework.orm.hibernate4.LocalSessionFactoryBean that should be configured.
  4. Note that we don’t need to use Hibernate Transaction Management, we can leave it to Spring declarative transaction management using @Transactional annotation.

For complete example go through [Spring Hibernate Integration](http://www.journaldev.com/3524/spring-hibernate-integration-example-tutorial-spring-4-hibernate-3-and-hibernate-4) and [Spring MVC Hibernate Integration](http://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial).

### What is HibernateTemplate class?

When Spring and Hibernate integration started, Spring ORM provided two helper classes –HibernateDaoSupport and HibernateTemplate. The reason to use them was to get the Session from Hibernate and get the benefit of Spring transaction management. However from Hibernate 3.0.1, we can use SessionFactory getCurrentSession() method to get the current session and use it to get the spring transaction management benefits. If you go through above examples, you will see how easy it is and that’s why we should not use these classes anymore.

One other benefit of HibernateTemplate was exception translation but that can be achieved easily by using @Repository annotation with service classes, shown in above spring mvc example. This is a trick question to judge your knowledge and whether you are aware of recent developments or not.

### How to integrate Hibernate with Servlet or Struts2 web applications?

Hibernate integration with Servlet or Struts2 needs to be done using ServletContextListener, a complete example can be found at [Hibernate Struts2 Integration Example](http://www.journaldev.com/3557/struts2-hibernate-integration-example-tutorial).

### Which design patterns are used in Hibernate framework?

Some of the design patterns used in Hibernate Framework are:

* 1. Domain Model Pattern – An object model of the domain that incorporates both behavior and data.
  2. 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.
  3. [Proxy Pattern](http://www.journaldev.com/1572/proxy-design-pattern-in-java-example-tutorial) for lazy loading
  4. [Factory pattern](http://www.journaldev.com/1392/factory-design-pattern-in-java) in SessionFactory

### What are best practices to follow with Hibernate framework?

Some of the best practices to follow in Hibernate are:

* 1. Always check the primary key field access, if it’s generated at the database layer then you should not have a setter for this.
  2. By default hibernate set the field values directly, without using setters. So if you want hibernate to use setters, then make sure proper access is defined as @Access(value=AccessType.PROPERTY).
  3. If access type is property, make sure annotations are used with getter methods and not setter methods. Avoid mixing of using annotations on both filed and getter methods.
  4. Use native sql query only when it can’t be done using HQL, such as using database specific feature.
  5. If you have to sort the collection, use ordered list rather than sorting it using Collection API.
  6. 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.
  7. For web applications, always try to use JNDI DataSource rather than configuring to create connection in hibernate.
  8. Avoid Many-to-Many relationships, it can be easily implemented using bidirectional One-to-Many and Many-to-One relationships.
  9. For collections, try to use Lists, maps and sets. Avoid array because you don’t get benefit of lazy loading.
  10. 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.
  11. Prefer DAO pattern for exposing the different methods that can be used with entity bean
  12. Prefer lazy fetching for associations

### What is Hibernate Validator Framework?

Data validation is integral part of any application. You will find data validation at presentation layer with the use of Javascript, then at the server side code before processing it. Also data validation occurs before persisting it, to make sure it follows the correct format.

Validation is a cross cutting task, so we should try to keep it apart from our business logic. That’s why JSR303 and JSR349 provides specification for validating a bean by using annotations. Hibernate Validator provides the reference implementation of both these bean validation specs. Read more at[Hibernate Validation Example](http://www.journaldev.com/3626/hibernate-validator-jsr303-example-tutorial).

### What is the benefit of Hibernate Tools Eclipse plugin?

Hibernate Tools plugin helps us in writing hibernate configuration and mapping files easily. The major benefit is the content assist to help us with properties or xml tags to use. It also validates them against the Hibernate DTD files, so we know any mistakes before hand. Learn how to install and use at[Hibernate Tools Eclipse Plugin](http://www.journaldev.com/2940/hibernate-tools-eclipse-plugin-for-generating-hibernate-mapping-and-configuration-files).

**1. What's Hibernate?**

Hibernate is a popular framework of Java which allows an efficient Object Relational mapping

using configuration files in XML format. After java objects mapping to database tables, database

is used and handled using Java objects without writing complex database queries.

**2. What is ORM?**

ORM (Object Relational Mapping) is the fundamental concept of Hibernate framework which

maps database tables with Java Objects and then provides various API's to perform different

types of operations on the data tables.

**3. How properties of a class are mapped to the columns of a database table in Hibernate?**

Mappings between class properties and table columns are specified in XML file as in the below

example:

**4. What's the usage of Configuration Interface in hibernate?**

Configuration interface of hibernate framework is used to configure hibernate. It's also used to

bootstrap hibernate. Mapping documents of hibernate are located using this interface.

**5. How can we use new custom interfaces to enhance functionality of built-in interfaces**

**of hibernate?**

We can use extension interfaces in order to add any required functionality which isn't supported

by built-in interfaces.

**6. 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.

**7. How do we create session factory in hibernate?**

1 / 9

**http://career.guru99.com/**

To create a session factory in hibernate, an object of configuration is created first which refers to

the path of configuration file and then for that configuration, session factory is created as given

in the example below:

[crayon-54f5aebad732e943779819/]

**8. What are POJOs and what's their significance?**

POJOs( Plain Old Java Objects) are java beans with proper getter and setter methods for each

and every properties.

Use of POJOs instead of simple java classes results in an efficient and well constructed code.

**9. What's HQL?**

HQL is the query language used in Hibernate which is an extension of SQL. HQL is very

efficient, simple and flexible query language to do various type of operations on relational

database without writing complex database queries.

**10. How can we invoke stored procedures in hibernate?**

In hibernate we can execute stored procedures using code as below:

[xml]

<sql-query name=”getStudents” callable=”true”>  
<return alias=”st” class=”Student”>  
<return-property name=”std\_id” column=”STD\_ID”/>  
<return-property name=”s\_name” column=”STD\_NAME”/>  
<return-property name=”s\_dept” column=”STD\_DEPARTMENT”/>  
{ ? = call selectStudents() }  
</return>  
</sql-query>

[/xml]

**11. What is criteria API?**

Criteria is a simple yet powerful API of hibernate which is used to retrieve entities through criteria object composition.

**12. What are the benefits of using Hibernate template?**

Following are some key benefits of using Hibernate template:

a. Session closing is automated.

b. Interaction with hibernate session is simplified.

c. Exception handling is automated.

**13. How can we see hibernate generated SQL on console?**

We need to add following in hibernate configuration file to enable viewing SQL on the console for debugging purposes:

[xml]

<property name=”show\_sql”>true</property>

[/xml]

**14. What are the two types of collections in hibernate?**

Following are the two types of collections in hibernate:

a. Sorted Collection

b. Order Collection

**15. What's the difference between session.save() and session.saveOrUpdate() methods**

**in hibernate?**

Session save() method saves a record only if it's unique with respect to its primary key and will fail to insert if primary key already exists in the table.

saveOrUpdate() method inserts a new record if primary key is unique and will update an existing record if primary key exists in the table already.

**16. What the benefits are of hibernate over JDBC?**

a. Hibernate can be used seamlessly with any type of database as its database independent

while in case of JDBC, developer has to write database specific queries.

b. Using hibernate, developer doesn't need to be an expert of writing complex queries as HQL

simplifies query writing process while in case of JDBC, its job of developer to write and tune

queries.

c. In case of hibernate, there is no need to create connection pools as hibernate does all

connection handling automatically while in case of JDBC, connection pools need to be created.

**17. How can we get hibernate statistics?**

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

below:

SessionFactory.getStatistics()

**18. What is transient instance state in Hibernate?**

If an instance is not associated with any persistent context and also, it has never been

associated with any persistent context, then it's said to be in transient state.

**19. How can we reduce database write action times in Hibernate?**

Hibernate provides dirty checking feature which can be used to reduce database write times.

Dirty checking feature of hibernate updates only those fields which require a change while keeps others unchanged.

**20. What's the usage of callback interfaces in hibernate?**

Callback interfaces of hibernate are useful in receiving event notifications from objects. For example, when an object is loaded or deleted, an event is generated and notification is sent using callback interfaces.

**21. When an instance goes in detached state in hibernate?**

When an instance was earlier associated with some persistent context (e.g. a table) and is no longer associated, it's called to be in detached state.

**22. What the four ORM levels are in hibernate?**

Following are the four ORM levels in hibernate:

a. Pure Relational

b. Light Object Mapping

c. Medium Object Mapping

d. Full Object Mapping

**23. What's transaction management in hibernate? How it works?**

Transaction management is the process of managing a set of statements or commands. In hibernate; transaction management is done by transaction interface as shown in below code:

[java]

Session s = null;

Transaction tr = null;

try {

s = sessionFactory.openSession();

tr = s.beginTransaction();

doTheAction(s);

tr.commit();

} catch (RuntimeException exc) {

tr.rollback();

} finally {

s.close();

}

**24. What the two methods are of hibernate configuration?**

We can use any of the following two methods of hibernate configuration:

a. XML based configuration ( using hibernate.cfg.xml file)

b. Programmatic configuration ( Using code logic)

**25. What is the default cache service of hibernate?**

Hibernate supports multiple cache services like EHCache, OSCache, SWARMCache and TreeCache and default cache service of hibernate is EHCache.

**26. What are the two mapping associations used in hibernate?**

In hibernate; we have following two types of mapping associations between entities:

a. One-to-One Association

b. Many-to-Many Association

**27. What's the usage of Hibernate QBC API?**

Hibernate Query By Criteria (QBC) API is used to create queries by manipulation of criteria objects at runtime.

**28. In how many ways, objects can be fetched from database in hibernate?**

Hibernate provides following four ways to fetch objects from database:

a. Using HQL

b. Using identifier

c. Using Criteria API

d. Using Standard SQL

**29. How primary key is created by using hibernate?**

Database primary key is specified in the configuration file hbm.xml. Generator can also be used

to specify how primary key is being created in the database.

In the below example, deptId acts as primary key:

[xml]

<id name=”deptId” type=”string” >  
<column name=”columnId” length=”30″/>  
<generator/>  
</id>

[/xml]

**30. How can we reattach any detached objects in Hibernate?**

Objects which have been detached and are no longer associated with any persistent entities can be reattached by calling session.merge() method of session class.

**31. What are different ways to disable hibernate second level cache?**

Hibernate second level cache can be disabled using any of the following ways:

a. By setting use\_second\_level\_cache as false.

b. By using CACHEMODE.IGNORE

c. Using cache provider as org.hibernate.cache.NoCacheProvider

**32. What is ORM metadata?**

All the mapping between classes and tables, properties and columns, Java types and SQL

types etc is defined in ORM metadata.

**33. Which one is the default transaction factory in hibernate?**

With hibernate 3.2, default transaction factory is JDBCTransactionFactory.

**34. What's the role of JMX in hibernate?**

Java Applications and components are managed in hibernate by a standard API called JMX

API. JMX provides tools for development of efficient and robust distributed, web based

solutions.

**35. How can we bind hibernate session factory to JNDI ?**

Hibernate session factory can be bound to JNDI by making configuration changes in

hibernate.cfg file.

**36. In how many ways objects can be identified in Hibernate?**

Object identification can be done in hibernate in following three ways:

a. Using Object Identity: Using == operator.

b. Using Object Equality: Using equals() method.

c. Using database identity: Relational database objects can be identified if they represent same row.

**37. What different fetching strategies are of hibernate?**

Following fetching strategies are available in hibernate:

1. Join Fetching

2. Batch Fetching

3. Select Fetching

4. Sub-select Fetching

**38. How mapping of java objects is done with database tables?**

To map java objects with database tables, we need to have Java beans properties names same as column names of a database table. Then mapping is provided in hbm.xml file as given below:

[xml]

<hibernate-mapping>  
<class name=”Student”  table=”tbl\_student”>  
<property  column=”studentname” length=”255″  
name=”studentName” not-null=”true”  type=”java.lang.String”/>  
<property  column=”studentDisciplne” length=”255″  
name=”studentDiscipline” not-null=”true”  type=”java.lang.String”/>  
</class>  
</hibernate-mapping>

[/xml]

**39. What are derived properties in hibernate?**

Derived properties are those properties which are not mapped to any columns of a database table. Such properties are calculated at runtime by evaluation of any expressions.

**40. What is meant by a Named SQL Query in hibernate and how it's used?**

Named SQL queries are those queries which are defined in mapping file and are called as required anywhere.

For example, we can write a SQL query in our XML mapping file as follows:

[xml]

SELECT std.STUDENT\_ID AS {std.STUDENT\_ID},

std.STUDENT\_DISCIPLINE AS {std.discipline},

FROM Student std WHERE std.NAME LIKE :name

Then this query can be called as follows:

[java]

List students = session.getNamedQuery(“studentdetails”)

.setString(“TomBrady”, name)

.setMaxResults(50)

.list();

[/java]

**41. What's the difference between load() and get() method in hibernate?**

Load() methods results in an exception if the required records isn't found in the database while

get() method returns null when records against the id isn't found in the database.

So, ideally we should use Load() method only when we are sure about existence of records

against an id.

**42. What's the use of version property in hibernate?**

Version property is used in hibernate to know whether an object is in transient state or in detached state.

**43. What is attribute oriented programming?**

In Attribute oriented programming, a developer can add Meta data (attributes) in the java source code to add more significance in the code. For Java (hibernate), attribute oriented programming is enabled by an engine called XDoclet.

**44. 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.

**45. Does hibernate support polymorphism?**

Yes, hibernate fully supports polymorphism. Polymorphism queries and polymorphism

associations are supported in all mapping strategies of hibernate.

**46. What the three inheritance models are of hibernate?**

Hibernate has following three inheritance models:

a. Tables Per Concrete Class

b. Table per class hierarchy

c. Table per sub-class

**47. How can we map the classes as immutable?**

If we don't want an application to update or delete objects of a class in hibernate, we can make the class as immutable by setting mutable=false

**48. What's general hibernate flow using RDBMS?**

General hibernate flow involving RDBMS is as follows:

a. Load configuration file and create object of configuration class.

b. Using configuration object, create sessionFactory object.

c. From sessionFactory, get one session.

d. Create HQL query.

e. Execute HQL query and get the results. Results will be in the form of a list.

**49. What is Light Object Mapping?**

Light Object Mapping is one of the levels of ORM quality in which all entities are represented as classes and they are mapped manually.

**50. What's difference between managed associations and hibernate associations?**

Managed associations relate to container management persistence and are bi-directional while hibernate associations are unidirectional.