**Basic Java Interview Questions**

### ****Q1. Explain JDK, JRE and JVM?****

|  |  |  |
| --- | --- | --- |
| ****JDK vs JRE vs JVM********JDK vs JRE vs JVM**** | | |
| **JDK** | **JRE** | **JVM** |
| It stands for Java Development Kit. | It stands for Java Runtime Environment. | It stands for Java Virtual Machine. |
| It is the tool necessary to compile, document and package Java programs. | JRE refers to a runtime environment in which java bytecode can be executed. | It is an abstract machine. It is a specification that provides run-time environment in which java bytecode can be executed. |
| Along with JRE, it includes an interpreter/loader, a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development. In short, it contains JRE + development tools. | It implements the JVM (Java Virtual Machine) and provides all the class libraries and other support files that JVM uses at runtime. So JRE is a software package that contains what is required to run a Java program. Basically, it’s an implementation of the JVM which physically exists. | JVM follows three notations: Specification(document that describes the implementation of the Java virtual machine), **Implementation**(program that meets the requirements of JVM specification) and **Runtime Instance**(instance of JVM is created whenever you write a java command on the command prompt and run class). |

### ****Q2. Explain public static void main(String args[]).****

* **public**: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
* **static** : It is a keyword in java which identifies it is class based i.e it can be accessed without creating the instance of a Class.
* **void** : It is the return type of the method. Void defines the method which will not return any value.
* **main**: It is the name of the methodwhich is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
* **String args[]** : It is the parameter passed to the main method.

### ****Q3. Why Java is platform independent?****

Platform independent practically means “write once run anywhere”. Java is called so because of its byte codes which can run on any system irrespective of its underlying operating system.

### ****Q4. Why java is not 100% Object-oriented?****

Java is not 100% Object-oriented because it makes use of eight primitive datatypes such as boolean, byte, char, int, float, double, long, short which are **not objects.**

### ****Q5. What are wrapper classes?****

Wrapper classes convert the java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they “wrap” the primitive data type into an object of that class. Refer to the below image which displays different primitive type, wrapper class and constructor argument.

### ****Q6. What are constructors in Java?****

In Java, constructor refers to a block of code which is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

There are two types of constructors:

1. Default constructor
2. Parameterized constructor

### ****Q7. What is singleton class and how can we make a class singleton?****

Singleton class is a class whose only one instance can be created at any given time, in one JVM. A class can be made singleton by making its constructor private.

### ****Q8. What is the difference between Array list and vector?****

|  |  |
| --- | --- |
| **Array List** | **Vector** |
| Array List is not synchronized. | Vector is synchronized. |
| Array List is fast as it’s non-synchronized. | Vector is slow as it is thread safe. |
| If an element is inserted into the Array List, it increases its Array size by 50%. | Vector defaults to doubling size of its array. |
| Array List does not define the increment size. | Vector defines the increment size. |
| Array List can only use Iterator for traversing an Array List. | Except Hashtable, Vector is the only other class which uses both Enumeration and Iterator. |

### ****Q9.What is the difference between HashMap and ConcurrentHashMap?****

|  |  |
| --- | --- |
| **HashMap** | **ConcurrentHashMap** |
| [HashMap](https://www.geeksforgeeks.org/java-util-hashmap-in-java/) is the Class which is under Traditional Collection | ConcurrentHashMap is a Class which is under Concurrent Collections |
| HashMap is non-Synchronized in nature i.e. HashMap is not Thread-safe | ConcurrentHashMap is Thread-safe in nature. |
| HashMap performance is relatively high because it is non-synchronized in nature and any number of threads can perform simultaneously | ConcurrentHashMap performance is low sometimes because sometimes Threads are required to wait on ConcurrentHashMap |
| While one thread is Iterating the HashMap object, if other thread try to add/modify the contents of Object then we will get Run-time exception saying **ConcurrentModificationException** | In ConcurrentHashMap we wont get any exception while performing any modification at the time of Iteration. |

### ****Q10. What is the difference between Array list and LinkedList?****

|  |  |
| --- | --- |
| **Array List** | **LinkedList** |
| [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/):-Implemented with the concept of dynamic array. | [LinkedList](http://geeksquiz.com/linked-list-in-java/):-Implemented with the concept of doubly linked list. |
| **Memory Allocation:**  ArrayList uses Array as underline datastructure to store the elements, and we know Array stores elements in consecutive manner | In LinkedList, elements can be stored at any available memory location as address of node is stored in previous node. |
| **Element Manipulation:**  Element manipulation with ArrayList is slow because it uses internally Array | For storing or removing of any new element needs to shift other elements in Array. |
| **Role as a Queue**:  ArrayList can be act as List only as it implements List interface only, | LinkedList can be act as List and Queue also as it implements List and Deque interface |
| Storing: | Insertions  are easy and fast in LinkedList as compared to ArrayList because there is no risk of resizing array |

### ****Q11. What are the differences between Heap and Stack Memory?****

|  |  |  |
| --- | --- | --- |
| **Features** | **Stack** | **Heap** |
| **Memory** | Stack memory is used only by one thread of execution. | Heap memory is used by all the parts of the application. |
| **Access** | Stack memory can’t be accessed by other threads. | Objects stored in the heap are globally accessible. |
| **Memory Management** | Follows LIFO manner to free memory. | Memory management is based on generation associated to each object. |
| **Lifetime** | Exists until the end of execution of the thread. | Heap memory lives from the start till the end of application execution. |
| **Usage** | Stack memory only contains local primitive and reference variables to objects in heap space. | Whenever an object is created, it’s always stored in the Heap space. |

### ****Q11.What is difference between object oriented programming language and object based programming language?****

Object based programming languages follow all the features of OOPs except Inheritance. Examples of object based programming languages are JavaScript, VBScript etc.

### ****Q12.What is static variable?****

* static variable is used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
* static variable gets memory only once in class area at the time of class loading.

### ****What is the 5 objects oriented design principle from SOLID?**** SOLID S for Single Responsibility Principle O for Open closed design principle L for Liskov substitution principle I for Interface segregation principle D for Dependency inversion principle

### *****How HashMap works in Java ?*****

[A HashMap in Java stores key-value pairs](http://www.javacodegeeks.com/2014/03/how-hashmap-works-in-java.html). The [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) requires a hash function and uses hashCode and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) are its capacity, its load factor and the threshold resizing.

## ****OOPS Java Interview Questions:****

### ****Q1. 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. There are two types of polymorphism:

1. Compile time polymorphism
2. Run time polymorphism

### ****Q2. What is the difference between abstract classes and interfaces?****

|  |  |
| --- | --- |
| **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 |

### ****Q3. What is method overloading and method overriding?****

#### ****Method Overloading :****

* In Method Overloading, Methods of the same class shares the same name but each method must have different number of parameters or parameters having different types and order.
* Method Overloading is to “add” or “extend” more to method’s behavior.
* It is a compile time polymorphism.
* The methods must have different signature.
* It may or may not need inheritance in Method Overloading.

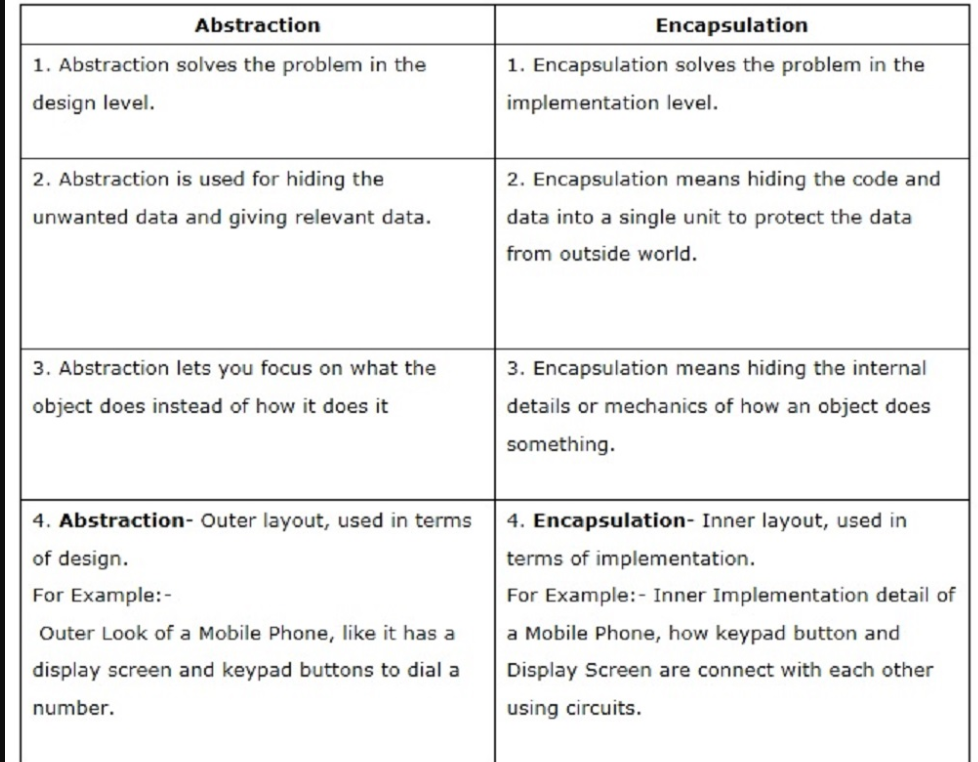
#### ****Method Overriding:****

* In Method Overriding, sub class have the same method with same name and exactly the same number and type of parameters and same return type as a super class.
* Method Overriding is to “Change” existing behavior of method.
* It is a run time polymorphism.
* The methods must have same signature.
* It always requires inheritance in Method Overriding.

### ****Q4. What is multiple inheritance? Is it supported by Java?****

* If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.
* The problem with multiple inheritance is that if multiple parent classes have a same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.
* Therefore, Java doesn’t support multiple inheritance. The problem is commonly referred as **Diamond Problem**.

### ****Q5. What is the difference between abstraction and encapsulation?****



## ****Strings****

**What is String in Java? String is a data type?**

String is a Class in java and defined in java.lang package. It’s not a primitive data type like int and long. String class represents character Strings. String is used in almost all the Java applications and there are some interesting facts we should know about String. String in [immutable](https://www.journaldev.com/129/how-to-create-immutable-class-in-java) and final in Java and JVM uses String Pool to store all the String objects.

Some other interesting things about String is the way we can instantiate a String object using double quotes and overloading of “+” operator for concatenation.

### ****Difference between String, StringBuffer and StringBuilder?****

String is immutable and final in java, so whenever we do String manipulation, it creates a new String. String manipulations are resource consuming, so java provides two utility classes for String manipulations – StringBuffer and StringBuilder.  
StringBuffer and StringBuilder are mutable classes. StringBuffer operations are thread-safe and synchronized where StringBuilder operations are not thread-safe. So when multiple threads are working on same String, we should use StringBuffer but in single threaded environment we should use StringBuilder.  
StringBuilder performance is fast than StringBuffer because of no overhead of synchronization.

### ****Why String is immutable or final in Java****

There are several benefits of String because it’s immutable and final.

* String Pool is possible because String is immutable in java.
* It increases security because any hacker can’t change its value and it’s used for storing sensitive information such as database username, password etc.
* Since String is immutable, it’s safe to use in multi-threading and we don’t need any synchronization.
* Strings are used in java classloader and immutability provides security that correct class is getting loaded by Classloader.

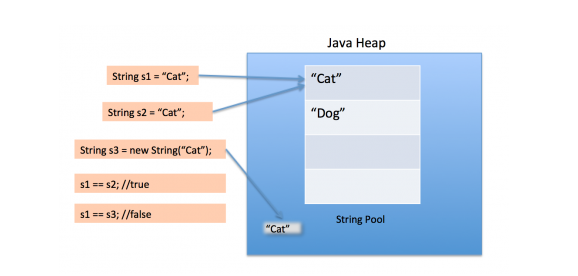
### ****How do you check if two Strings are equal in Java?****

There are two ways to check if two Strings are equal or not – using “==” operator or using equals method. When we use “==” operator, it checks for value of String as well as reference but in our programming, most of the time we are checking equality of String for value only. So we should use equals method to check if two Strings are equal or not.  
There is another function equalsIgnoreCase that we can use to ignore case.

|  |
| --- |
| String s1 = "abc";  String s2 = "abc";  String s3= new String("abc");  System.out.println("s1 == s2 ? "+(s1==s2)); //true  System.out.println("s1 == s3 ? "+(s1==s3)); //false  System.out.println("s1 equals s3 ? "+(s1.equals(s3))); //true |

### What is String Pool?

* As the name suggests, String Pool is a pool of Strings stored in [Java heap memory](https://www.journaldev.com/4098/java-heap-space-vs-stack-memory). We know that String is special class in java and we can create String object using new operator as well as providing values in double quotes.
* When we use **double quotes** to create a String, it first looks for String with same value in the String pool, if found it just returns the reference else it creates a new String in the pool and then returns the reference.



### ****Swap two Strings without using third user defined variable in Java****

|  |  |
| --- | --- |
| public static void main(String args[])      {          // Declare two strings          String a = "Hello";          String b = "World";            // Print String before swapping          System.out.println("Strings before swap: a = " +                                         a + " and b = "+b);            // append 2nd string to 1st          a = a + b;            // store intial string a in string b          b = a.substring(0,a.length()-b.length());            // store initial string b in string a          a = a.substring(b.length());            // print String after swapping          System.out.println("Strings after swap: a = " +                                       a + " and b = " + b);      } | 1) Append second string to first string and  store in first string:  a = a + b  2) call the method substring(int beginindex, int endindex)  by passing beginindex as 0 and endindex as,  a.length() - b.length():  b = substring(0,a.length()-b.length());  3) call the method substring(int beginindex) by passing  b.length() as argument to store the value of initial  b string in a  a = substring(b.length()); |

## ****Java Threads****

#### What is the difference between processes and threads ?

A process is an execution of a program, while a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is a single execution sequence within a process. A process can contain multiple threads. A [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is sometimes called a lightweight process.

#### Explain different ways of creating a thread. Which one would you prefer and why ?

There are three ways that can be used in order for a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) to be created:

* A class may extend the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class.
* A class may implement the [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface.
* An application can use the [Executor](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html) framework, in order to create a thread pool.

The [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface is preferred, as it does not require an object to inherit the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class. In case your application design requires multiple inheritance, only interfaces can help you. Also, the thread pool is very efficient and can be implemented and used very easily.

#### What is the difference between a synchronized method and a synchronized block ?

In Java programming, each object has a lock. A thread can acquire the lock for an object by using the synchronized keyword. The synchronized keyword can be applied in a method level (coarse grained lock) or block level of code (fine grained lock).

## ****Exceptional Handling****

#### What are the two types of Exceptions in Java ? Which are the differences between them ?

Java has two types of exceptions: checked exceptions and unchecked exceptions. Unchecked exceptions do not need to be declared in a method or a constructor’s throws clause, if they can be thrown by the execution of the method or the constructor, and propagate outside the method or constructor boundary. On the other hand, checked exceptions must be declared in a method or a constructor’s throws clause

#### What is the difference between Exception and Error in java ?

[Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) and [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) classes are both subclasses of the [Throwable](http://docs.oracle.com/javase/7/docs/api/java/lang/Throwable.html) class. The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) class is used for exceptional conditions that a user’s program should catch. The [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) class defines exceptions that are not excepted to be caught by the user program.

#### What is the difference between throw and throws ?

The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

#### What is the importance of finally block in exception handling ?

A finally block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

#### What will happen to the Exception object after exception handling ?

The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) object will be garbage collected in the next garbage collection.

#### How does finally block differ from finalize() method ?

#### A finally block will be executed whether or not an exception is thrown and is used to release those resources held by the application. Finalize is a protected method of the Object class, which is called by the Java Virtual Machine (JVM) just before an object is garbage collected.

## ****Hibernate****

### 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

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

Java Persistence API (JPA) provides specification for managing the relational data in applications.

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

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.

### Name some important interfaces of Hibernate framework?

1. **SessionFactory (org.hibernate.SessionFactory)**: SessionFactory is an [immutable](https://www.journaldev.com/129/how-to-create-immutable-class-in-java) 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.Connection and 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?

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 with org.hibernate.annotations.GenericGenerator and org.hibernate.annotations.Parameter

### 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.

### 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.

### 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.

### Explain Hibernate second level cache?

It is related to the session factory, tools like ehcache can be used in order to achieve the second level cache

### 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.

### 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.

### What are the collection types in Hibernate?

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

1. Bag
2. Set
3. List
4. Array
5. Map

### How to implement Joins in Hibernate?

There are various ways to implement joins in hibernate.

* Using associations such as one-to-one, one-to-many etc.
* Using JOIN in the HQL query. There is another form “join fetch” to load associated data simultaneously, no lazy loading.
* 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 approa

### 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 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:

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

### 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.

1. [Hibernate One to One Mapping](https://www.journaldev.com/2916/hibernate-one-to-one-mapping-example-annotation)
2. [Hibernate One to Many Mapping](https://www.journaldev.com/2924/hibernate-one-to-many-mapping-annotation)
3. [Hibernate Many to Many Mapping](https://www.journaldev.com/2934/hibernate-many-to-many-mapping-join-tables)

### 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.

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.

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

Some of the [design patterns](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) used in Hibernate Framework are:

* Domain Model Pattern – An object model of the domain that incorporates both behavior and data.
* Data Mapper – A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.
* [Proxy Pattern](https://www.journaldev.com/1572/proxy-design-pattern) for lazy loading
* [Factory pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java) in SessionFactory

### 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

**List some of the properties of hibernate?**

* hibernate.dialect
* hibernate.connection.driver\_class
* hibernate.connection.url
* hibernate.connection.username
* hibernate.connection.password
* hibernate.connection.pool\_size
* hibernate.connection.autocommit

## ****Spring****

### ****What is Spring Framework?****

Spring is the most broadly used framework for the development of Java Enterprise Edition applications. The core features of Spring can be used in developing any Java application.

We can use its extensions for building various web applications on top of the Java EE platform, or we may just use its dependency injection provisions in simple standalone applications.

### ****What are the benefits of using Spring?****

Spring targets to make Java EE development easier. Here are the advantages of using it:

* **Lightweight:**there is a slight overhead of using the framework in development
* **Inversion of Control (IoC):** Spring container takes care of wiring dependencies of various objects, instead of creating or looking for dependent objects
* **Aspect Oriented Programming (AOP):**Spring supports AOP to separate business logic from system services
* **IoC container:** it manages Spring Bean life cycle and project specific configurations
* **MVC framework:** that is used to create web applications or RESTful web services, capable of returning XML/JSON responses
* **Transaction management:** reduces the amount of boiler-plate code in JDBC operations, file uploading, etc., either by using Java annotations or by Spring Bean XML configuration file
* **Exception Handling:**Spring provides a convenient API for translating technology-specific exceptions into unchecked exceptions

### ****What Spring sub-projects do you know? Describe them briefly.****

* **Core** – a key module that provides fundamental parts of the framework, like IoC or DI
* **JDBC** – this module enables a JDBC-abstraction layer that removes the need to do JDBC coding for specific vendor databases
* **ORM integration** – provides integration layers for popular object-relational mapping APIs, such as JPA, JDO, and Hibernate
* **Web** – a web-oriented integration module, providing multipart file upload, Servlet listeners, and web-oriented application context functionalities
* **MVC framework** – a web module implementing the Model View Controller design pattern
* **AOP module** – aspect-oriented programming implementation allowing the definition of clean method-interceptors and pointcuts

### ****What is Inversion of Control?****

Inversion of Control is a principle in software engineering by which the control of objects or portions of a program is transferred to a container or framework. It’s most often used in the context of object-oriented programming

### ****What is Dependency Injection?****

Dependency Injection, an aspect of Inversion of Control (IoC), is a general concept stating that you do not create your objects manually but instead describe how they should be created. An IoC container will instantiate required classes if needed.

### ****How dependency injection in Spring?****

A few different options exist:

* 1. Constructor-based

|  |
| --- |
|  |

* 1. Setter-based
  2. Field-based

### ****Which is the best way of injecting beans and why?****

The recommended approach is to use constructor arguments for mandatory dependencies and setters for optional ones. Constructor injection allows injecting values to immutable fields and makes testing easier.

### ****What is the difference between****BeanFactory****and****ApplicationContext****?****

|  |  |
| --- | --- |
| BeanFactory | ApplicationContext |
| **Annotation based dependency Injection** is not supported by BeanFactory | ApplicationContext supports using annotation **@PreDestroy, @Autowired**. |
| BeanFactory uses**lazy initialization** approach | ApplicationContext uses **eager initialization** approach |
| BeanFactory used to be explicitly provided a resource object | ApplicationContext creates and manages resources objects on its own |
| BeanFactory do not. | ApplicationContext supports **internationalization** |
| Doesn’t support | Support  many enterprise services such JNDI access, EJB integration, remoting. |

### ****What is a Spring Bean?****

The Spring Beans are Java Objects that are initialized by the Spring IoC container.

### ****What is the default bean scope in Spring framework?****

By default, a Spring Bean is initialized as a singleton.

### ****How to define the scope of a bean?****

To set Spring Bean’s scope, we can use @Scope annotation or “scope” attribute in XML configuration files. There are five supported scopes:

* **singleton**
* **prototype**
* **request**
* **session**
* **global-session**

### ****What is Spring Security?****

Spring Security is a separate module of the Spring framework that focuses on providing authentication and authorization methods in Java applications. It also takes care of most of the common security vulnerabilities such as CSRF attacks.

To use Spring Security in web applications, you can get started with a simple annotation: @EnableWebSecurity.

### ****Name some of the Design Patterns used in the Spring Framework?****

* **Singleton Pattern:** Singleton-scoped beans
* **Factory Pattern:** Bean Factory classes
* **Prototype Pattern:** Prototype-scoped beans
* **Adapter Pattern:** Spring Web and Spring MVC
* **Proxy Pattern:** Spring Aspect Oriented Programming support
* **Template Method Pattern:** JdbcTemplate, HibernateTemplate, etc.
* **Front Controller:** Spring MVC DispatcherServlet
* **Data Access Object:** Spring DAO support
* **Model View Controller:**Spring MVC

### ****How does the scope****Prototype****work?****

Scope prototype means that every time you call for an instance of the Bean, Spring will create a new instance and return it. This differs from the default singleton scope, where a single object instance is instantiated once per Spring IoC container.

### ****What is the role of the****@Autowired****annotation?****

The @Autowired annotation can be used with fields or methods for injecting a bean by type. This annotation allows Spring to resolve and inject collaborating beans into your bean.

### ****What is the role of the****@Required****annotation?****

The @Required annotation is used on setter methods, and it indicates that the bean property that has this annotation must be populated at configuration time. Otherwise, the Spring container will throw a BeanInitializationException exception.

### ****What is the Role of the****@Qualifier****Annotation?****

It is used simultaneously with the @Autowired annotation to avoid confusion when multiple instances of a bean type are present.

### ****How to handle exceptions in Spring MVC environment?****

There are three ways to handle exceptions in Spring MVC:

1. **Using @ExceptionHandler at controller level** – this approach has a major feature – the @ExceptionHandler annotated method is only active for that particular controller, not globally for the entire application
2. **Using HandlerExceptionResolver** – this will resolve any exception thrown by the application
3. **Using @ControllerAdvice**– Spring 3.2 brings support for a global @ExceptionHandler with the @ControllerAdvice annotation, which enables a mechanism that breaks away from the older MVC model and makes use of ResponseEntity along with the type safety and flexibility of @ExceptionHandler

### ****How to validate if the bean was initialized using valid values?****

Spring supports [JSR-303](http://beanvalidation.org/1.0/spec/)annotation-based validations. JSR-303 is a specification of the Java API for bean validation, part of JavaEE and JavaSE, which ensures that properties of a bean meet specific criteria, using annotations such as @NotNull, @Min, and @Max

### ****What is Spring MVC Interceptor and how to use it?****

Spring MVC Interceptors allow us to intercept a client request and process it at three places – before handling, after handling, or after completion (when the view is rendered) of a request.

The interceptor can be used for cross-cutting concerns and to avoid repetitive handler code like logging, changing globally used parameters in Spring model, etc.

### ****What is a Controller in Spring MVC?****

Simply put, all the requests processed by the DispatcherServlet are directed to classes annotated with @Controller. Each controller class maps one or more requests to methods that process and execute the requests with provided inputs.

### ****How does the****@RequestMapping****annotation work?****

The @RequestMapping annotation is used to map web requests to Spring Controller methods. In addition to simple use cases, we can use it for mapping of HTTP headers, binding parts of the URI with @PathVariable, and working with URI parameters and the @RequestParam annotation.

### ****What’s the Difference Between****@Controller****,****@Component****,****@Repository,****and****@Service****Annotations in Spring?****

According to the official Spring documentation, *@Component* is a generic stereotype for any Spring-managed component. *@Repository*, *@Service*, and *@Controller* are specializations of *@Component* for more specific use cases, for example, in the persistence, service, and presentation layers, respectively.

Let’s take a look at specific use cases of last three:

* **@*Controller*** – indicates that the class serves the role of a controller, and detects *@RequestMapping*annotations within the class
* **@*Service*** – indicates that the class holds business logic and calls methods in the repository layer
* **@*Repository*** – indicates that the class defines a data repository; its job is to catch platform-specific exceptions and re-throw them as one of Spring’s unified unchecked exceptions

### ****What are****DispatcherServlet****and****ContextLoaderListener****?****

Simply put, in the Front Controller design pattern, a single controller is responsible for directing incoming HttpRequests to all of an application’s other controllers and handlers.

**Spring’s DispatcherServlet implements this pattern and is, therefore, responsible for correctly coordinating the HttpRequests to the right handlers.**

On the other hand, ContextLoaderListener starts up and shuts down Spring’s root WebApplicationContext. It ties the lifecycle of ApplicationContext to the lifecycle of the ServletContext. We can use it to define shared beans working across different Spring contexts.

### ****What is****ViewResolver****in Spring?****

The ViewResolver enables an application to render models in the browser – without tying the implementation to a specific view technology – by mapping view names to actual views.

### ****What is a****MultipartResolver****and when is it used?****

The MultipartResolver interface is used for uploading files. The Spring framework provides one MultipartResolver implementation for use with Commons FileUpload and another for use with Servlet 3.0 multipart request parsing.

### ****What is Spring****JDBCTemplate****class and how to use it?****

The Spring JDBC template is the primary API through which we can access database operations logic that we’re interested in:

* creation and closing of connections
* executing statements and stored procedure calls
* iterating over the *ResultSet* and returning results

### ****How would you enable t****ransactions****in Spring and what are their benefits?****

There are two distinct ways to configure Transactions – with annotations or by using Aspect Oriented Programming (AOP) – each with their advantages.

The benefits of using Spring Transactions, according to the [official docs](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/transaction.html), are:

* Provide a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO
* Support declarative transaction management
* Provide a simpler API for programmatic transaction management than some complex transaction APIs such as JTA
* Integrate very well with Spring’s various data access abstractions

### ****What is Spring DAO?****

Spring Data Access Object is Spring’s support provided to work with data access technologies like JDBC, Hibernate, and JPA in a consistent and easy way.

### ****What is Aspect-Oriented Programming?****

Aspects enable the modularization of cross-cutting concerns such as transaction management that span multiple types and objects by adding extra behavior to already existing code without modifying affected classes.

### ****What are****Aspect****,****Advice****,****Pointcut,****and****JoinPoint****in AOP?****

* ***Aspect***: a class that implements cross-cutting concerns, such as transaction management
* **Advice**: the methods that get executed when a specific JoinPoint with matching Pointcut is reached in the application
* ***Pointcut***: a set of regular expressions that are matched with JoinPoint to determine whether Adviceneeds to be executed or not
* ***JoinPoint***: a point during the execution of a program, such as the execution of a method or the handling of an exception

### ****What is****Weaving****?****

According to the [official docs](https://docs.spring.io/spring/docs/current/spring-framework-reference/html/aop.html), weaving is a process that links aspects with other application types or objects to create an advised object. This can be done at compile time, load time, or at runtime. Spring AOP, like other pure Java AOP frameworks, performs weaving at runtime.

## ****Design Pattern****

### Singleton Pattern (Creational [Design Pattern](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) category. )

Singleton

* Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists in the java virtual machine.
* The singleton class must provide a global access point to get the instance of the class.
* Singleton pattern is used for [logging](https://www.journaldev.com/977/logger-in-java-logging-example), drivers objects, caching and [thread pool](https://www.journaldev.com/1069/threadpoolexecutor-java-thread-pool-example-executorservice).
* Singleton design pattern is also used in other design patterns like [Abstract Factory](https://www.journaldev.com/1418/abstract-factory-design-pattern-in-java), [Builder](https://www.journaldev.com/1425/builder-design-pattern-in-java), [Prototype](https://www.journaldev.com/1440/prototype-design-pattern-in-java), [Facade](https://www.journaldev.com/1557/facade-design-pattern-in-java) etc.
* Singleton design pattern is used in core java classes also, for example java.lang.Runtime, java.awt.Desktop.

### Java Singleton Pattern

To implement Singleton pattern, we have different approaches but all of them have following common concepts.

* Private constructor to restrict instantiation of the class from other classes.
* Private static variable of the same class that is the only instance of the class.
* Public static method that returns the instance of the class, this is the global access point for outer world to get the instance of the singleton class.

**Spring boot**