Seneca Global Interview Questions

--------------------\*\*\*\*--------------------------

1. What is Threadpoolexecutor?

The java.util.concurrent.ThreadPoolExecutor is an implementation of the [**ExecutorService**](http://tutorials.jenkov.com/java-util-concurrent/executorservice.html) interface. The ThreadPoolExecutor executes the given task (Callable or Runnable) using one of its internally pooled threads.

The thread pool contained inside the ThreadPoolExecutor can contain a varying amount of threads. The number of threads in the pool is determined by these variables:

* corePoolSize
* maximumPoolSize

If less than corePoolSize threads are created in the the thread pool when a task is delegated to the thread pool, then a new thread is created, even if idle threads exist in the pool.

If the internal queue of tasks is full, and corePoolSize threads or more are running, but less than maximumPoolSize threads are running, then a new thread is created to execute the task.

|  |
| --- |
| Here is a diagram illustrating the ThreadPoolExecutor principles:A ThreadPoolExecutor. |
| **A ThreadPoolExecutor** |

2. What is JMS ?

JMS (Java Message Service) is an API that provides the facility to create, send and read messages. It provides loosely coupled, reliable and asynchronous communication.

JMS is also known as a messaging service.

Understanding Messaging

Messaging is a technique to communicate applications or software components.

JMS is mainly used to send and receive message from one application to another.

Requirement of JMS

Generally, user sends message to application. But, if we want to send message from one application to another, we need to use JMS API.

Consider a scenario, one application A is running in INDIA and another application B is running in USA. To send message from A application to B, we need to use JMS

3. Which implementation you use in JMS?

----no answer—

4. What is Singleton class?

Singleton Pattern says that just**"define a class that has only one instance and provides a global point of access to it".**

In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.

There are two forms of singleton design pattern

* **Early Instantiation:** creation of instance at load time.
* **Lazy Instantiation:** creation of instance when required.

#### **Advantage of Singleton design pattern**

* Saves memory because object is not created at each request. Only single instance is reused again and again.

#### **Usage of Singleton design pattern**

* Singleton pattern is mostly used in multi-threaded and database applications. It is used in logging, caching, thread pools, configuration settings etc.

6. What are the difference between Arraylist and linkedlist? When we use what?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ArrayList and LinkedList both implements List interface and maintains insertion order. Both are non synchronized classes.  But there are many differences between ArrayList and LinkedList classes that are given below.   |  |  | | --- | --- | | **ArrayList** | **LinkedList** | | 1) ArrayList internally uses **dynamic array** to store the elements. | LinkedList internally uses **doubly linked list** to store the elements. | | 2) Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory. | | 3) ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. | | 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data. |  **Example of ArrayList and LinkedList in Java** Let's see a simple example where we are using ArrayList and LinkedList both.  **import** java.util.\*;  **class** TestArrayLinked{  **public** **static** **void** main(String args[]){      List<String> al=**new** ArrayList<String>();//creating arraylist    al.add("Ravi");//adding object in arraylist    al.add("Vijay");    al.add("Ravi");    al.add("Ajay");      List<String> al2=**new** LinkedList<String>();//creating linkedlist    al2.add("James");//adding object in linkedlist    al2.add("Serena");    al2.add("Swati");    al2.add("Junaid");      System.out.println("arraylist: "+al);    System.out.println("linkedlist: "+al2);   }  }  [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestArrayLinked)  Output:  arraylist: [Ravi,Vijay,Ravi,Ajay]  linkedlist: [James,Serena,Swati,Junaid]  7. What are the types of exceptions raised in spring framework?  -------------NoAnswer--------  8. What are the repositories in Springframework?  ---------------NoAnswer----------  9. How you handle exceptions in P/L SQL?  ---------------------NoAnswer---------------------  10.what is connection pooling?  a **connection pool** is a [cache](https://en.wikipedia.org/wiki/Database_cache) of [database connections](https://en.wikipedia.org/wiki/Database_connection) maintained so that the connections can be reused when future requests to the database are required.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] Connection pools are used to enhance the performance of executing commands on a database. Opening and maintaining a database connection for each user, especially requests made to a dynamic database-driven [website](https://en.wikipedia.org/wiki/Website) application, is costly and wastes resources. In connection pooling, after a connection is created, it is placed in the pool and it is used again so that a new connection does not have to be established. If all the connections are being used, a new connection is made and is added to the pool.  11. How can we write our own immutable class?  To create immutable class in java, you have to do following steps.   1. Declare the class as final so it can’t be extended. 2. Make all fields private so that direct access is not allowed. 3. Don’t provide setter methods for variables 4. Make all **mutable fields final** so that it’s value can be assigned only once. 5. Initialize all the fields via a constructor performing deep copy. 6. Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.   12. Can we clone immutable class?  -----------------NoAnswer------------------  13. Can we provide threadsafe for Singleton design pattern and how?  [Singleton](http://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples) is one of the most widely used creational [design pattern](http://www.journaldev.com/1827/java-design-patterns-example-tutorial) to restrict the object creation by applications. In real world applications, resources like Database connections or Enterprise Information Systems (EIS) are limited and should be used wisely to avoid any resource crunch. To achieve this, we can implement Singleton design pattern to create a wrapper class around the resource and limit the number of object created at runtime to one.  There are three ways through which we can achieve thread safety.   1. **Create the instance variable at the time of class loading.**   **Pros**:   * + Thread safety without synchronization   + Easy to implement   **Cons**:   * + Early creation of resource that might not be used in the application.   + The client application can’t pass any argument, so we can’t reuse it. For example, having a generic singleton class for database connection where client application supplies database server properties.  1. **Synchronize the getInstance() method**   **Pros**:   * + Thread safety is guaranteed.   + Client application can pass parameters   + Lazy initialization achieved   **Cons**:   * + Slow performance because of locking overhead.   + Unnecessary synchronization that is not required once the instance variable is initialized.  1. **Use synchronized block inside the if loop**   **Pros**:   * + Thread safety is guaranteed   + Client application can pass arguments   + Lazy initialization achieved   + Synchronization overhead is minimal and applicable only for first few threads when the variable is null.   **Cons**:Extra if condition  14. What are the support issues you faced in your project?  ------------------NoAnswer--------------  15. How connection pooling works in java?  Connection pooling has become the standard for middleware database drivers. The process of creating a connection, always an expensive, time-consuming operation, is multiplied in these environments where a large number of users are accessing the database in short, unconnected operations. Creating connections over and over in these environments is simply too expensive.  The transaction profile for Web applications, probably the most common application in use today, is that of a large number of users performing short, discrete database operations. These applications usually perform work centered around creating a web page that will be sent back to the user's browser. Transactions are generally short-lived, and user sessions are often limited in time.  A connection pool operates by performing the work of creating connections ahead of time, In the case of a JDBC connection pool, a pool of **Connection** objects is created at the time the application server (or some other server) starts. These objects are then managed by a **pool manager**that disperses connections as they are requested by clients and returns them to the pool when it determines the client is finished with the **Connection**object. A great deal of housekeeping is involved in managing these connections.  When the connection pool server starts, it creates a predetermined number of **Connection**objects. A client application would then perform a JNDI lookup to retrieve a reference to a **DataSource** object that implements the **ConnectionPoolDataSource**interface. The client application would not need make any special provisions to use the pooled data source; the code would be no different from code written for a nonpooled **DataSource.**  When the client application requests a connetion from the **ConnetionPoolDataSource,**the data source implementation would retrieve a physical connection to the client application. the **ConnectionPoolDataSource**would return a **Connection**object that implemented the **PooledConnection**interface.  The **PooledConnection**interface dictates the use of *event listeners.*These event listeners allow the connection pool manager to capture important connection events, such as attempts by the client application to close the connection. When the driver traps a close-connection event, it intercedes and performs a pseudo-close operation that merely takes the **Connection**object, returns it to the pool of available connection, and performs any housekeeping that is necessary.  Nexii Labs   * 1. How many ways we can create an object?   There are total 5 core ways to create objects in Java which are explained below with their example followed by bytecode of the line which is creating the object. However, lots of Apis are out there are which creates objects for us but these Apis will also are using one of these 5 core ways indirectly e.g. Spring BeanFactory.  [5-different-ways-of-object-creation-in-Java-with-example-and-explanation](https://1.bp.blogspot.com/-FfxCYRWady8/WI2_7Y00PKI/AAAAAAAAKds/_pjbMvBpDsItJnBAxeuHDp375cIoOph6QCK4B/s1600/5-different-ways-of-object-creation-in-Java.png)  If you will execute program given in end, you will see method 1, 2, 3 uses the constructor to create the object while 4, 5 doesn’t call the constructor to create the object. **1. Using the new keyword** It is the most common and regular way to create an object and actually very simple one also. By using this method we can call whichever constructor we want to call (no-arg constructor as well as parametrised).  Employee emp1 = **new** Employee();  0: new *#19 // class org/programming/mitra/exercises/Employee*  3: dup  4: invokespecial *#21 // Method org/programming/mitra/exercises/Employee."":()V* **2. Using Class.newInstance() method** We can also use the newInstance() method of the Class class to create objects, This newInstance() method calls the no-arg constructor to create the object. We can create objects by newInstance() in following way.  Employee emp2 = (Employee) Class.forName("org.programming.mitra.exercises.Employee")  .newInstance();  Or  Employee emp2 = Employee.class.newInstance();  51: invokevirtual *#70 // Method java/lang/Class.newInstance:()Ljava/lang/Object;* **3. Using newInstance() method of Constructor class** Similar to the newInstance() method of Class class, There is one newInstance() method in the java.lang.reflect.Constructor class which we can use to create objects. We can also call parameterized constructor, and private constructor by using this newInstance() method.  Both newInstance() methods are known as reflective ways to create objects. In fact newInstance() method of Class class internally uses newInstance() method of Constructor class. That's why the later one is preferred and also used by different frameworks like Spring, Hibernate, Struts etc. To know differences between both newInstance() methods read [Creating objects through Reflection in Java with Example](https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html).  Constructor<Employee> constructor = Employee.class.getConstructor();  Employee emp3 = constructor.newInstance();  111: invokevirtual *#80 // Method java/lang/reflect/Constructor.newInstance:([Ljava/lang/Object;)Ljava/lang/Object;* **4. Using clone() method** Whenever we call clone() on any object JVM actually creates a new object for us and copy all content of the previous object into it. Creating an object using clone method does not invoke any constructor.  To use clone() method on an object we need to implements Cloneable and define clone() method in it.  Employee emp4 = (Employee) emp3.clone();  162: invokevirtual *#87 // Method org/programming/mitra/exercises/Employee.clone ()Ljava/lang/Object;*  Java cloning is the most debatable topic in Java community and it surely does have its drawbacks but it is still the most popular and easy way of creating a copy of any object until that object is full filling mandatory conditions of Java cloning. I have also written 3 more articles on [Java Cloning](https://programmingmitra.blogspot.in/search/label/Java%20Cloning) go ahead and read them if you want to know more about Java cloning (Note: Begin from the bottom). **5. Using deserialization** Whenever we serialize and then deserialize an object JVM creates a separate object for us. In deserialization, JVM doesn’t use any constructor to create the object. To deserialize an object we need to implement the Serializable interface in our class.  ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream("data.obj"));  Employee emp5 = (Employee) in.readObject();  261: invokevirtual *#118 // Method java/io/ObjectInputStream.readObject:()Ljava/lang/Object;*  As we can see in above bytecodes all 4 methods call get converted to invokevirtual (object creation is directly handled by these methods) except the first one which got converted to two calls one is new and other is invokespecial (call to the constructor).  2. What is cloning? **Object Cloning in Java** constructor in java  The **object cloning** is a way to create exact copy of an object. For this purpose, clone() method of Object class is used to clone an object.  The **java.lang.Cloneable interface** must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates **CloneNotSupportedException**.  The **clone() method** is defined in the Object class. Syntax of the clone() method is as follows:   1. **protected** Object clone() **throws** CloneNotSupportedException  **Why use clone() method ?** The **clone() method** saves the extra processing task for creating the exact copy of an object. If we perform it by using the new keyword, it will take a lot of processing to be performed that is why we use object cloning. **Advantage of Object cloning** Less processing task. **Example of clone() method (Object cloning)** Let's see the simple example of object cloning   1. **class** Student18 **implements** Cloneable{ 2. **int** rollno; 3. String name; 5. Student18(**int** rollno,String name){ 6. **this**.rollno=rollno; 7. **this**.name=name; 8. } 9. **public** Object clone()**throws** CloneNotSupportedException{ 10. **return** **super**.clone(); 11. } 13. **public** **static** **void** main(String args[]){ 14. **try**{ 15. Student18 s1=**new** Student18(101,"amit"); 17. Student18 s2=(Student18)s1.clone(); 19. System.out.println(s1.rollno+" "+s1.name); 20. System.out.println(s2.rollno+" "+s2.name); 22. }**catch**(CloneNotSupportedException c){} 24. } 25. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Student18)  Output:101 amit  101 amit  3. Types of cloning in java? |

**Cloning** is a process of creating an exact copy of an existing object in the memory. In java, **clone()** method of **java.lang.Object** class is used for cloning process. This method creates an exact copy of an object on which it is called through **field-by-field assignment** and returns the reference of that object. Not all the objects in java are eligible for cloning process. The objects which implement **Cloneable interface** are only eligible for cloning process. Cloneable interface is a [marker interface](http://javaconceptoftheday.com/marker-interface-java/) which is used to provide the marker to cloning process. Click [here](http://javaconceptoftheday.com/clone-method-java-lang-object-class/) to see more info on clone() method in java.

Both shallow copy and deep copy are related to this cloning process. The default version of clone() method creates the shallow copy of an object. To create the deep copy of an object, you have to override the clone() method. Let’s see how these shallow copy and deep copy work.

## Shallow Copy In Java :

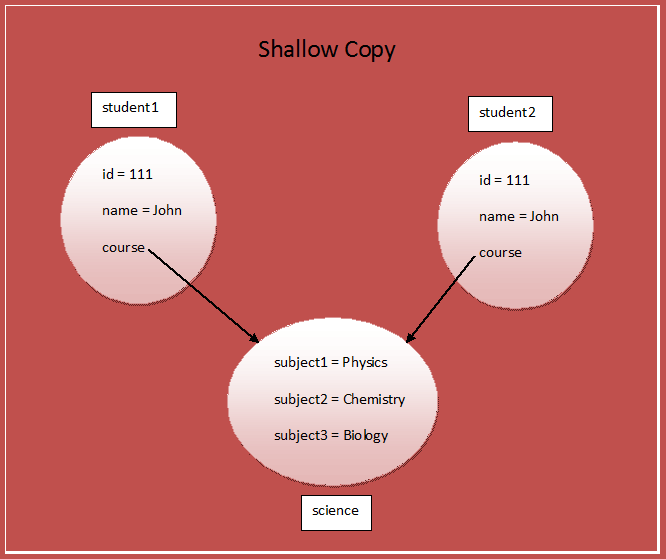
The default version of clone() method creates the shallow copy of an object. The shallow copy of an object will have exact copy of all the fields of original object. If original object has any references to other objects as fields, then only references of those objects are copied into clone object, copy of those objects are not created. That means any changes made to those objects through clone object will be reflected in original object or vice-versa. Shallow copy is not 100% disjoint from original object. Shallow copy is not 100% independent of original object.

Below is the example which creates the shallow copy of an object ‘**student1**‘.[?](http://javaconceptoftheday.com/difference-between-shallow-copy-vs-deep-copy-in-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  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | classCourse  {  String subject1;  String subject2;  String subject3;  publicCourse(String sub1, String sub2, String sub3)  {  this.subject1 = sub1;  this.subject2 = sub2;  this.subject3 = sub3;  }  }  classStudent implementsCloneable  {  intid;  String name;  Course course;  publicStudent(intid, String name, Course course)  {  this.id = id;  this.name = name;  this.course = course;  }  //Default version of clone() method. It creates shallow copy of an object.  protectedObject clone() throwsCloneNotSupportedException  {  returnsuper.clone();  }  }  publicclassShallowCopyInJava  {  publicstaticvoidmain(String[] args)  {  Course science = newCourse("Physics", "Chemistry", "Biology");  Student student1 = newStudent(111, "John", science);  Student student2 = null;  try  {  //Creating a clone of student1 and assigning it to student2  student2 = (Student) student1.clone();  }  catch(CloneNotSupportedException e)  {  e.printStackTrace();  }  //Printing the subject3 of 'student1'  System.out.println(student1.course.subject3);         //Output : Biology  //Changing the subject3 of 'student2'  student2.course.subject3 = "Maths";  //This change will be reflected in original student 'student1'  System.out.println(student1.course.subject3);       //Output : Maths  }  } |

In the above example, ‘**student1**‘ is an object of ‘**Student**‘ class which has three fields – **id**, **name** and **course**. ‘**course**‘ is a reference variable pointing to a ‘**Course**‘ type object. Clone of ‘**student1**‘ is created by calling clone method on it and assigned it to ‘**student2**‘. As default version of clone method creates the shallow copy, the ‘**course**‘ field of both ‘**student1**‘ and ‘**student2**‘ will be pointing to same ‘**Course**‘ object. So, any changes made to this object through ‘**student2**‘ will be reflected in ‘**student1**‘ or vice-versa.

See the below picture for more clear understanding.



## Deep Copy In Java :

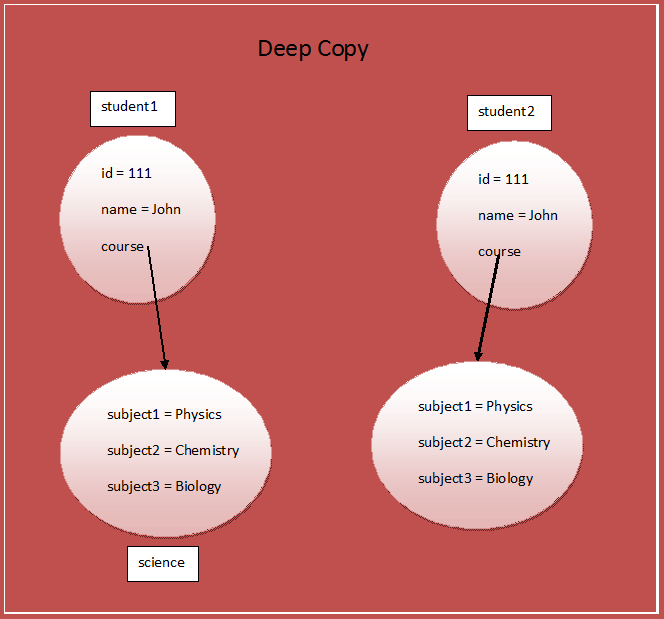
Deep copy of an object will have exact copy of all the fields of original object just like shallow copy. But in additional, if original object has any references to other objects as fields, then copy of those objects are also created by calling clone() method on them. That means clone object and original object will be 100% disjoint. They will be 100% independent of each other. Any changes made to clone object will not be reflected in original object or vice-versa.

To create a deep copy of an object, you have to override the clone() method as demonstrated in the below example.

[?](http://javaconceptoftheday.com/difference-between-shallow-copy-vs-deep-copy-in-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  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86 | classCourse implementsCloneable  {      String subject1;        String subject2;        String subject3;        publicCourse(String sub1, String sub2, String sub3)      {          this.subject1 = sub1;            this.subject2 = sub2;            this.subject3 = sub3;      }        protectedObject clone() throwsCloneNotSupportedException      {          returnsuper.clone();      }  }    classStudent implementsCloneable  {      intid;        String name;        Course course;        publicStudent(intid, String name, Course course)      {          this.id = id;            this.name = name;            this.course = course;      }        //Overriding clone() method to create a deep copy of an object.        protectedObject clone() throwsCloneNotSupportedException      {          Student student = (Student) super.clone();            student.course = (Course) course.clone();            returnstudent;      }  }    publicclassDeepCopyInJava  {      publicstaticvoidmain(String[] args)      {          Course science = newCourse("Physics", "Chemistry", "Biology");            Student student1 = newStudent(111, "John", science);            Student student2 = null;            try          {              //Creating a clone of student1 and assigning it to student2                student2 = (Student) student1.clone();          }          catch(CloneNotSupportedException e)          {              e.printStackTrace();          }            //Printing the subject3 of 'student1'            System.out.println(student1.course.subject3);         //Output : Biology            //Changing the subject3 of 'student2'            student2.course.subject3 = "Maths";            //This change will not be reflected in original student 'student1'            System.out.println(student1.course.subject3);       //Output : Biology      }  } |

Below picture shows how deep copy of ‘student1’ is created.



1. What is Synchronization?

### Synchronization

At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called **synchronization**. The synchronization keyword in java creates a block of code referred to as critical section.

**General Syntax :**

synchronized (object)

{

//statement to be synchronized

}

Every Java object with a critical section of code gets a lock associated with the object. To enter critical section a thread need to obtain the corresponding object's lock.

#### Why we use Syncronization ?

If we do not use syncronization, and let two or more threads access a shared resource at the same time, it will lead to distorted results.

Consider an example, Suppose we have two different threads **T1** and **T2**, T1 starts execution and save certain values in a file *temporary.txt* which will be used to calculate some result when T1 returns. Meanwhile, T2 starts and before T1 returns, T2 change the values saved by T1 in the file temporary.txt (temporary.txt is the shared resource). Now obviously T1 will return wrong result.

To prevent such problems, synchronization was introduced. With synchronization in above case, once T1 starts using *temporary.txt* file, this file will be **locked**(LOCK mode), and no other thread will be able to access or modify it until T1 returns.

#### Using Synchronized Methods

Using Synchronized methods is a way to accomplish synchronization. But lets first see what happens when we do not use synchronization in our program.

5.What is Constructor chaining?

Calling a constructor from the another constructor of same class is known as Constructor chaining. We will understand this with the help of below program.

## Example Program

In the below example the class “ChainingDemo” has 4 constructors and we are calling one constructor from another using **this() statement.**For e.g. in order to call a constructor with single string argument we have supplied a string in this() statement like **this(“hello”).**

**Note**: this() should always be the first statement in **constructor** otherwise you will get the below error message:

Exceptionin thread "main" java.lang.Error:Unresolved compilation problem:

Constructor call must be the first statement in a constructor

**Complete Code:**

package beginnersbook.com;

publicclassChainingDemo{

//default constructor of the class

publicChainingDemo(){

System.out.println("Default constructor");

}

publicChainingDemo(String str){

this();

System.out.println("Parametrized constructor with single param");

}

publicChainingDemo(String str,int num){

//It will call the constructor with String argument

this("Hello");

System.out.println("Parametrized constructor with double args");

}

publicChainingDemo(int num1,int num2,int num3){

// It will call the constructor with (String, integer) arguments

this("Hello",2);

System.out.println("Parametrized constructor with three args");

}

publicstaticvoid main(String args[]){

//Creating an object using Constructor with 3 int arguments

ChainingDemo obj =newChainingDemo(5,5,15);

}

}

Output:

Default constructor

Parametrized constructor with single param

Parametrized constructor withdouble args

Parametrized constructor with three args

6. What are Stereo type annotations in spring?

Now that we understand what these words mean it might give us some insight as to what role they play in our Spring projects. When a class is annotated with one of the 1following Stereotypes Spring will automatically register them in the application context. This makes the class available for dependency injection in other classes and this become vital to building out our applications. These classes can all be found under the [org.springframework.stereotype package](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/stereotype/package-summary.html).

|  |  |
| --- | --- |
| **Annotation** | **Description** |
| @Component | Indicates that an annotated class is a “component” |
| @Controller | Indicates that an annotated class is a “Controller” (e.g. |
| @Service | Indicates that an annotated class is a “Service”, originally defined by Domain-Driven Design (Evans, 2003) as “an operation offered as an interface that stands alone in the model, with no encapsulated state.” |
| @Repository | Indicates that an annotated class is a “Repository”, originally defined by Domain-Driven Design (Evans, 2003) as “a mechanism for encapsulating storage, retrieval, and search behavior which emulates a collection of objects”. |

\* You might have expected to see[*@RestController*](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html)in this list but it was omitted on purpose.  @RestController is a convenience annotation that is itself annotated with[*@Controller*](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/stereotype/Controller.html) and [*@ResponseBody*](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/ResponseBody.html).

There is a reason that @Component was at the top of this list. @Component is a generic stereotype annotation for any Spring-managed component. This means that if you want to register a class in the application context you can use @Component to do so. The other stereotypes (Controller, Service, Repository) are simply specializations of the @Component class. In fact, if you look at their source code you will see they themselves are annotated with @Component. If this is true, why wouldn’t we just annotate everything with @Component?

First, it helps us and other developers classify a particular class. We can look at a class with the @Controller annotation and understand right away what its purpose is. Second, as we will see in the demo, it also helps us define well crafted point-cut demarcations when we get into using AOP. This should all make a little more sense as we move through a demo, so let’s get to it.

7. What are the difference between sleep, yeild and join?

# Static Thread class methods

* sleep()
* yield()

# Instance Thread class method

* join()

# sleep()

The sleep() method causes the **currently executing thread** to sleep for X milliseconds.  
An example for sleeping for 2 seconds (2000 milliseconds).



|  |  |
| --- | --- |
| 1 | Thread.sleep(2000); |

# yield()

Currently running thread goes back to its runnable state (so it’s ready to run again, but **is currently not running**), so other threads have a chance to run. It’s only a hint for the thread scheduler (it can be ignored), so the given behavior (goin back to runnable state) is not guaranteed.



|  |  |
| --- | --- |
| 1 | Thread.yield(); |

# join



|  |  |
| --- | --- |
| 1  2 | Threadt=newThread();  t.join(); |

This instance method causes the currently running thread to join the end of the thread “t”. You will get it after this example…

An example:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | classRunnableImplimplementsRunnable{    *@Override*      publicvoidrun(){          System.out.println("one");              try{                  Thread.sleep(1000);              }catch(InterruptedExceptione){                  e.printStackTrace();              }          System.out.println("two");      }    }    publicclassStarter{        publicstaticvoidmain(String[]args)throwsInterruptedException{          Threadt=newThread(newRunnableImpl());          Thread t2=newThread(newRunnableImpl());            t.start();          t.join();// the main thread waits till the "t" thread is finished          t2.start();      }    } |

The output is:



|  |  |
| --- | --- |
| 1  2  3  4 | one  two  one  two |

Without the t.join() the output would be:



|  |  |
| --- | --- |
| 1  2  3  4 | one  one  two  two |

1. How can we send attachments using javamail?

For sending email with attachment, JavaMail API provides some useful classes like BodyPart, MimeBodyPart etc.

|  |
| --- |
| For better understanding of this example, learn the steps of sending email using JavaMail API first. |
| For sending the email using JavaMail API, you need to load the two jar files:   * **mail.jar** * **activation.jar**   [download these jar files](https://www.javatpoint.com/src/mail/mailactivation.zip) (or) go to the Oracle site to download the latest version. |

### **Sending email with attachment using JavaMail API**

There are total 7 steps for sending attachment with email. They are:

1. **Get the session object**
2. **compose message**
3. **create MimeBodyPart object and set your message text**
4. **create new MimeBodyPart object and set DataHandler object to this object**
5. **create Multipart object and add MimeBodyPart objects to this object**
6. **set the multiplart object to the message object**
7. **send message**

### **Example of sending email with attachment in Java**

1. **import** java.util.\*;
2. **import** javax.mail.\*;
3. **import** javax.mail.internet.\*;
4. **import** javax.activation.\*;
6. **class** SendAttachment{
7. **public** **static** **void** main(String [] args){
9. String to="sonoojaiswal1987@gmail.com";//change accordingly
10. **final** String user="sonoojaiswal@javatpoint.com";//change accordingly
11. **final** String password="xxxxx";//change accordingly
13. //1) get the session object
14. Properties properties = System.getProperties();
15. properties.setProperty("mail.smtp.host", "mail.javatpoint.com");
16. properties.put("mail.smtp.auth", "true");
18. Session session = Session.getDefaultInstance(properties,
19. **new** javax.mail.Authenticator() {
20. **protected** PasswordAuthentication getPasswordAuthentication() {
21. **return** **new** PasswordAuthentication(user,password);
22. }
23. });
25. //2) compose message
26. **try**{
27. MimeMessage message = **new** MimeMessage(session);
28. message.setFrom(**new** InternetAddress(user));
29. message.addRecipient(Message.RecipientType.TO,**new** InternetAddress(to));
30. message.setSubject("Message Aleart");
32. //3) create MimeBodyPart object and set your message text
33. BodyPart messageBodyPart1 = **new** MimeBodyPart();
34. messageBodyPart1.setText("This is message body");
36. //4) create new MimeBodyPart object and set DataHandler object to this object
37. MimeBodyPart messageBodyPart2 = **new** MimeBodyPart();
39. String filename = "SendAttachment.java";//change accordingly
40. DataSource source = **new** FileDataSource(filename);
41. messageBodyPart2.setDataHandler(**new** DataHandler(source));
42. messageBodyPart2.setFileName(filename);

45. //5) create Multipart object and add MimeBodyPart objects to this object
46. Multipart multipart = **new** MimeMultipart();
47. multipart.addBodyPart(messageBodyPart1);
48. multipart.addBodyPart(messageBodyPart2);
50. //6) set the multiplart object to the message object
51. message.setContent(multipart );
53. //7) send message
54. Transport.send(message);
56. System.out.println("message sent....");
57. }**catch** (MessagingException ex) {ex.printStackTrace();}
58. }
59. }

9.Why wait, notify, notifyAll methods in Object class?

-----------------NoAnswer--------------

10.What is copy constructor?

--------NoAnswer----------

11. Can we use both this and super keywords in constructor?

If there is no **super**() in a **constructor** the compiler **will** add one implicitly. So **both** are **constructor** calls. **Constructor** must always be the first statement. So **we can** not have two statements as first statement, hence either **we can** call **super**() or **we can** call this() from the**constructor**, but not **both.**

12. How can we change maven war name?

------------------------NoAnswer----------------

13. What is SVN synchronise?

--------------------NoAnswer-----------

14. How can we place project in SVN?

The repository should already be created.  
You should have Tortoise SVN installed on your pc.  
You should have the URL to the repository and the username and password, if necessary, to access it,you have to follow the remaing procedure.

CTS Interview Questions

* 1. What is Externalization?

Thus externalization comes to give the programmers full control in reading and writing objects during serialization.

# **The Externalizable Interface**

When you want to control the process of reading and writing objects during the serialization and de-serialization process, have the object’s class implemented the **java.io.Externalizable** interface. Then you implement your own code to write object’s states in the **writeExternal()** method and read object’s states in the **readExternal()** method. These methods are defined by the Externalizable interface as follows:

* **writeExternal(ObjectOutput out)**: The object implements this method to save its contents by calling the methods of DataOutput for its primitive values or calling the writeObject method of ObjectOutput for objects, strings, and arrays.

* **readExternal(ObjectInput in)**: The object implements this method to restore its contents by calling the methods of DataInput for primitive types and readObject for objects, strings and arrays.

2. Immutable in concurrency environmet?

---------------------NoAnswer---------------

3. Difference btw static synchronized and non-static synchronized map?

---------------------NoAnswer---------------

4.What is hash collision?

-------------------NoAnswer-------------

5. What complexity involved in hashmap by calling get and put methods through hash collision?

-------------------NoAnswer-------------

6.Difference btw enumeration and iterator?

Though both Iterator and Enumeration allows you to traverse over elements of Collections in Java, there is some significant difference between them e.g. Iterator also allows you to remove elements from collection during traversal but Enumeration doesn't allow that, it doesn't got the remove() method. Enumeration is also a legacy class and not all Collection supports it e.g. Vector supports Enumeration but ArrayList doesn't. On the other hand Iterator is the standard class for iteration and traversal. By the way,  what is difference between Enumeration and Iterator in Java? This question is from early ages of  Java interview , I have not seen this question on recent interviews but it was quite common during 2000 to 2007 period, now days questions like [implementation of HashMapor ConcurrentHashMap](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html) etc has taken its place, nonetheless its very important to know the fundamental *difference between Iterator and Enumeration*. Some time its also asked as Iterator vs Enumeration or Enumeration vs Iterator which is same. Important point to note is that both Iterator and Enumeration provides way to traverse or navigate through entire collection in Java.

## Iterator vs Enumeration

[difference between iterator and enumeration in java](http://javarevisited.blogspot.com/2011/01/how-classpath-work-in-java.html)Between Enumeration and Iterator, Enumeration is older and its there from JDK1.0, while iterator was introduced later. Iterator can be used with ArrayList, HashSet and other collection classes.  Another similarity between Iterator and Enumeration in Java is that  *functionality of Enumeration interface is duplicated by the Iterator interface*.

Only major difference between Enumeration and iterator is Iterator has a remove() method while Enumeration doesn't. Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects, where as by using Iterator we can manipulate the objects like adding and removing the objects from collection e.g. Arraylist.

Also **Iterator**is more secure and safe as compared to **Enumeration**because it  does not allow other thread to modify the collection object while some thread is iterating over it and throws **ConcurrentModificationException.**This is by far most important fact for me for deciding between Iterator vs Enumeration in Java.

In Summary both Enumeration and Iterator will give successive elements, but Iterator is new and improved version where method names are shorter, and has new method called remove. Here is a short comparison:

**Enumeration**

hasMoreElement()

nextElement()

N/A

**Iterator**

hasNext()

next()

remove()

So **Enumeration** is used when ever we want to make Collection objects as Read-only.

7. What is immutable?(Releted to Strings)

In java, **string objects are immutable**. Immutable simply means unmodifiable or unchangeable.

Once string object is created its data or state can't be changed but a new string object is created.

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

1. **class** Testimmutablestring{
2. **public** **static** **void** main(String args[]){
3. String s="Sachin";
4. s.concat(" Tendulkar");//concat() method appends the string at the end
5. System.out.println(s);//will print Sachin because strings are immutable objects
6. }
7. }

5. What complexity involved in hashmap by calling get and put methods through hash collision?

--------------------NoAnswer-----------------

1. Do you know UML?

UML stands for Unified Modeling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

* Improved integration between structural models like class diagrams and behavior models like activity diagrams.
* Added the ability to define a hierarchy and decompose a software system into components and sub-components.
* The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed statechart diagrams to state machine diagrams, also known as state diagrams.

1. What are desgin patterns you know?

Design patterns represent the best practices used by experienced object-oriented software developers. Design patterns are solutions to general problems that software developers faced during software development. These solutions were obtained by trial and error by numerous software developers over quite a substantial period of time.

#### Singleton

This is the most used pattern. A lot of framework already implement this pattern, such as Spring, CDI (via @ApplicationScoped) or EJBs (using @Singleton). Still, it is nice to know how to implement it the old way. ;)

publicclassSingletonSample{

privatestaticSingletonSample instance =null;

privateSingletonSample(){

}

publicstaticSingletonSample getInstance(){

if(instance ==null){

instance =newSingletonSample();

}

return instance;

}}

There is no secret. You need an object that only needs to be instantiate once, so, you can use a singleton. The class needs to declare a private constructor to prevent people to instantiate it from outside the class. Also, you need to declare a static field of the type of the singleton. The method getInstance() assures that only one instance of this class is created at runtime.

#### Initialization on Demand Holder

This pattern is much like the Singleton, but the Initialization on Demand Holder has critical advantage over the Singleton: It is thread safe.   
The getInstance() method from the Singleton pattern is not thread safe, not if you don't make it synchronized, so, it can be instantiate more than once. If you do make your method synchronized, you are making your getInstance() method slower than it could be if it were not.

publicclassSingletonSample{

privateSingletonSample(){

}

publicstaticSingletonSample getInstance(){

returnSingletonSampleHolder.INSTANCE;

}

privatestaticclassSingletonSampleHolder{

privatestaticfinalSingletonSample INSTANCE =newSingletonSample();

}

}

As the name says, this type of Singleton does not initialize the instance until the getInstance() is called, with the advantage that it is thread safe.   
Since the JVM does not load the SingletonSampleHolder at startup because there is no static reference of this class anywhere in the code, the instance is not created until the first call of the getInstance() method.

#### The Strategy and The Factory pattern

Both are well-known design patterns. For sure two of the most useful design patterns, specially using them together. When both are combined, you can create objects from a given qualifier. The example is right below:

publicinterfaceBuilding{

String getType();

}

publicclassHouseimplementsBuilding{

publicString getType(){

return"house"

}

}

publicclassEdificeimplementsBuilding{

publicString getType(){

return"edifice"

}

}

publicclassBuildingFactory{

privatestaticMap<String,Building> instances;

static{

instances =newHashMap<>();

instances.put("house",newHouse());

instances.put("edifice",newEdifice());

}

publicstatic<T extendsBuilding> T getBuilding(String type){

return(T) instances.get(type);

}

}

Building building =BuildingFactory.getBuilding("house");

If you need a specific building, you need just to give a building type and one will be returned, or null there is no instance for this type. Very useful, besides, it gives us a chance to use the most of polymorphism.

#### Fluent Builder

Some objects require lots of parameters to be created. In this case, either using the constructor to create this object or using the setters will make our code ugly and hard to understand.

The builder pattern can help us in this case.

publicclassProduct{

privateString id;

privateString name;

privateString description;

privateDouble value;

privateProduct(Builder builder){

setId(builder.id);

setName(builder.name);

setDescription(builder.description);

setValue(builder.value);

}

publicstaticBuilder newProduct(){

returnnewBuilder();

}

publicString getId(){

return id;

}

publicvoid setId(String id){

this.id = id;

}

publicString getName(){

return name;

}

publicvoid setName(String name){

this.name = name;

}

publicString getDescription(){

return description;

}

publicvoid setDescription(String description){

this.description = description;

}

publicDouble getValue(){

return value;

}

publicvoid setValue(Double value){

this.value = value;

}

publicstaticfinalclassBuilder{

privateString id;

privateString name;

privateString description;

privateDouble value;

privateBuilder(){

}

publicBuilder id(String id){

this.id = id;

returnthis;

}

publicBuilder name(String name){

this.name = name;

returnthis;

}

publicBuilder description(String description){

this.description = description;

returnthis;

}

publicBuilder value(Double value){

this.value = value;

returnthis;

}

publicProduct build(){

returnnewProduct(this);

}

}

}

Product product =Product.newProduct()

.id(1l)

.description("TV 46'")

.value(2000.00)

.name("TV 46'")

.build();

Okay, this class is not that big and it does not have a lot of fields, but, this is just for an example, so... :P

#### Chain of Responsibility

You always need to build applications which require a lot of business logic. Behind this much of logic, there is always high complexity. This high complexity makes our code harder to understand, as well as harder to track, to log and so on.   
The CoR pattern makes us break our code into little pieces and organize them into sequential steps.

publicinterfaceCommand<T>{

boolean execute(T context);

}

publicclassFirstCommandimplementsCommand<Map<String,Object>>{

publicboolean execute(Map<String,Object> context){

//doing something in here

}

}

publicclassSecondCommandimplementsCommand<Map<String,Object>>{

publicboolean execute(Map<String,Object> context){

//doing something in here

}

}

publicclassChain{

publicList<Command> commands;

publicChain(Command... commands){

this.commands =Arrays.asList(commands);

}

publicvoid start(Object context){

for(Command command : commands){

boolean shouldStop = command.execute(context);

if(shouldStop){

return;

}

}

}

}

Chain chain =newChain(newFirstCommand(),newSecondCommand());

Map<String,Object> context =newHashMap<>();

context.put("some parameter","some value");

chain.start(context);

Now we can break our code into Commands and separate each logic in just one place. Also, we can reorganize as we want, making our code more decoupled, thus solving our complexity problem.

#### Template method

This pattern defines a skeleton in a method for a operation. It is very useful when your have common method calls but different behaviors. This pattern is totally based in polymorphism.

publicabstractclassAnimal{

publicabstractvoid makeSound();

publicabstractvoid eatFood();

publicabstractvoid sleep();

publicvoid doEveryday(){

makeSound();

eatFood();

sleep();

}

}

publicclassDogextendsAnimal{

publicvoid makeSound(){

//bark!

}

publicvoid eatFood(){

//eat dog food

}

publicvoid sleep(){

//sleep a lot!

}

}

publicclassCatextendsAnimal{

publicvoid makeSound(){

//meow!

}

publicvoid eatFood(){

//eat cat food

}

publicvoid sleep(){

//sleep just a little bit

}

}

#### State pattern

A lot of objects have states. For example, a radio. A radio have basically two states: on and off. Can we represent this in object oriented programming? Well, yes:

publicclassRadio{

privateboolean on;

privateRadioState state;

publicRadio(RadioState state){

this.state = state;

}

publicvoid execute(){

state.execute(this);

}

publicvoid setState(RadioState state){

this.state = state;

}

publicvoid setOn(boolean on){

this.on = on;

}

publicboolean isOn(){

return on;

}

publicboolean isOff(){

return!on;

}

}

publicinterfaceRadioState{

void execute(Radio radio);

}

publicclassOnRadioStateimplementsRadioState{

publicvoid execute(Radio radio){

//throws exception if radio is already on

radio.setOn(true);

}

}

publicclassOffRadioStateimplementsRadioState{

publicvoid execute(Radio radio){

//throws exception if radio is already off

radio.setOn(false);

}

}

Radio radio =newRadio(newOffRadioState());//initial status

radio.setState(newOnRadioState());

radio.execute();//radio on

radio.setState(newOffRadioState());

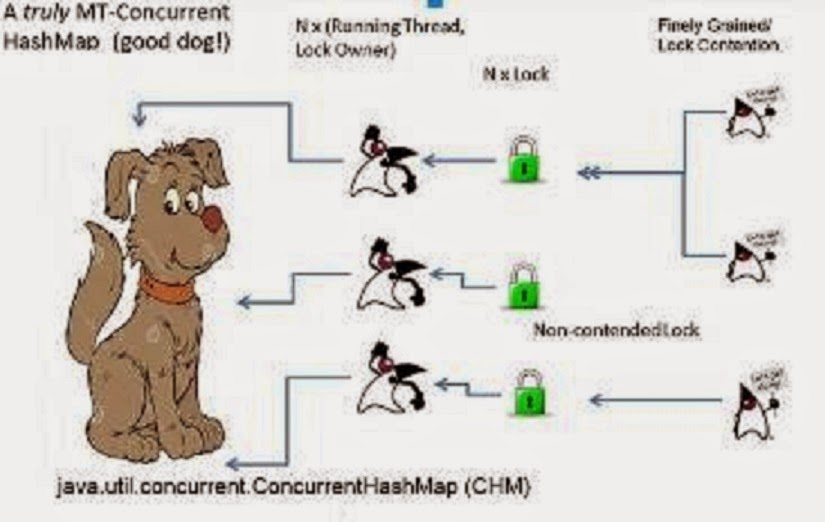
radio.execute();//radio off

10. Difference btw concurrent hashmap and synchronized hashmap?

the difference between Hashtable and ConcurrentHashMap, both can be used in the multithreaded environment but once the size of Hashtable becomes considerable large performance degrade because for iteration it has to be locked for a longer duration.

Since ConcurrentHashMap introduced the concept of segmentation, how large it becomes only certain part of it get locked to provide thread safety so many other readers can still access map without waiting for iteration to complete.

In Summary, ConcurrentHashMap only locked certain portion of Map while Hashtable locks full map while doing iteration. This will be clearer by looking at this diagram which explains the internal working of ConcurrentHashMap in Java.

[](http://3.bp.blogspot.com/-f2O3cncdnjk/VJ5UDxij_FI/AAAAAAAACSU/ut1ybg3WR2E/s1600/ConcurrentHashMap+in+Java+with+Example.png)

### The difference between ConcurrentHashMap and Collections.synchronizedMap

ConcurrentHashMap is designed for concurrency and improve performance while HashMap which is non-synchronized by nature can be synchronized by applying a wrapper using synchronized Map. Here are some of the common differences between ConcurrentHashMap and synchronized map in Java  
  
ConcurrentHashMap does not allow null keys or null values while synchronized HashMap allows one null key.

11.Can we String in switch class/

------------------NoAnswer------------

12.What are difference btw hashmap and hashtable?

HashMap and Hashtable both are used to store data in key and value form. Both are using hashing technique to store unique keys.

But there are many differences between HashMap and Hashtable classes that are given below.

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| 1) HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code. | Hashtable is **synchronized**. It is thread-safe and can be shared with many threads. |
| 2) HashMap **allows one null key and multiple null values**. | Hashtable **doesn't allow any null key or value**. |
| 3) HashMap is a **new class introduced in JDK 1.2**. | Hashtable is a **legacy class**. |
| 4) HashMap is **fast**. | Hashtable is **slow**. |
| 5) We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap); | Hashtable is internally synchronized and can't be unsynchronized. |
| 6) HashMap is **traversed by Iterator**. | Hashtable is **traversed by Enumerator and Iterator**. |
| 7) Iterator in HashMap is **fail-fast**. | Enumerator in Hashtable is **not fail-fast**. |
| 8) HashMap inherits **AbstractMap** class. | Hashtable inherits **Dictionary** class. |

project related questions

------------------------------

1.What technlogies do u use in your project?Tell me the versions?

---------------NoAnswer------------------

2.How will u deploy your project by using tomcat?

---------------NoAnswer------------------

3.How will u get log files?

---------------NoAnswer------------------

4.Which type of difficulties do u find at the time of deployments?

---------------NoAnswer------------------

5.What do u work do done in last week?

---------------NoAnswer------------------

6.How will u partcipate in your project?Roles and responcibilities?

---------------NoAnswer------------------

7.How to maintain security in your project?

---------------NoAnswer------------------

8.Which type of protocol do u use?Is it (http/https)

---------------NoAnswer------------------

9.How to maintain security in database?

---------------NoAnswer------------------

10.Which type of precations do u take for data storing ?hackers

---------------NoAnswer------------------

12.How will u store the data in the form of encryption or decryption?

---------------NoAnswer------------------

13.How will u done authentications and autherization?

---------------NoAnswer------------------

14.What is the last bug fixing u done?

---------------NoAnswer------------------

15.If u login in "X" module but u cant browse "Y" module?How will u do that?

---------------NoAnswer------------------

16.How do u fix bugs by using logfiles?

---------------NoAnswer------------------

18.How will u avoid conflicts?What will u do?

---------------NoAnswer------------------

19.What is the difficult situations r u ever face in working on your project?

---------------NoAnswer------------------

20.How will u get the work?

---------------NoAnswer------------------

21.Do u get any exceptions in deployments?

---------------NoAnswer------------------

22.Who will deploy the project on dev environment?

---------------NoAnswer------------------

23.What is the build process?

---------------NoAnswer------------------

24.What is maven?Why do u use?

Maven is a powerful *project management tool* that is based on POM (project object model). It is used for projects build, dependency and documentation.

It simplifies the build process like ANT. But it is too much advanced than ANT.

Current version of Maven is 3.

25.What is the advantages your progect developed by using spring MVC?What knowledge do u get?

---------------NoAnswer------------------

26.What is difference between spring-MVC and another MVCs?

---------------NoAnswer------------------

VIRINCHI(Technical F2F)

----------------------------------

>Introduction about interviewer

1.Tell me about Yourself?

---------------NoAnswer------------------

2.How much rating will u give on core java?

---------------NoAnswer------------------

3.Do u know collections?which collecton classes do u use in ur project?tell me perticular scenario where do u

use exactly in ur project?

---------------NoAnswer------------------

4.which version eclipse do u use?

Neon

5.Which version JBOSS u used?Do u deploy it?

---------------NoAnswer------------------

6.What is your current project?What is the flow?

---------------NoAnswer------------------

7.What is spring life cycle?Tell me states parallel to ur project?

---------------NoAnswer------------------

8.What is SingleTon?Did u write singleton in ur project?Tell me exactly where do u

use in your project?

---------------NoAnswer------------------

9.What is connection pooling in JDBC?

-------Repeated---------------

10.What is log4j?What is it purpose?How do u fix bugs using log4j file?What it contains?

---------------NoAnswer------------------

11.What is ANT?

12.What is Sturts?Do u have any idea about it?

---------------NoAnswer------------------

13.What is JSON ?

JSON: **J**ava**S**cript **O**bject **N**otation.

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

## Exchanging Data

When exchanging data between a browser and a server, the data can only be text.

JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.

14.How U implemented jsp paje in ur project?Dont U write JS in that ?Dont u get chanse to write

Js in that?

---------------NoAnswer------------------

15.What u done recently in ur project?

---------------NoAnswer------------------

16.What Exceptions u get in ur project?How do u fix it?Which Scenarios do u get these Exception?

---------------NoAnswer------------------

17.How to handle Exception by using jsp pages?

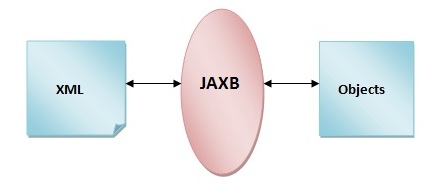
---------------NoAnswer------------------

18.What is jax-b?

# **JAXB Tutorial**

JAXB tutorial provides concepts and API to convert object into XML and XML into object. Our JAXB tutorial is designed for beginners and professionals.

**JAXB** stands for Java Architecture for XML Binding. It provides mechanism to marshal (write) java objects into XML and unmarshal (read) XML into object. Simply, you can say it is used to convert java object into xml and vice-versa.



## Features of JAXB 2.0

JAXB 2.0 includes several features that were not present in JAXB 1.x. They are as follows:

**1) Annotation support**: JAXB 2.0 provides support to annotation so less coding is required to develop JAXB application. The javax.xml.bind.annotation package provides classes and interfaces for JAXB 2.0.

**2) Support for all W3C XML Schema features**: it supports all the W3C schema unlike JAXB 1.0.

**3) Additional Validation Capabilities**: it provides additional validation support by JAXP 1.3 validation API.

**4) Small Runtime Library**: it required small runtime library that JAXB 1.0.

**5) Reduction of generated schema-derived classes**: it reduces a lot of generated schema-derived classes

19.Do u use web services in your project?

Yes…

20.Do u work on UI technologies?

Yes……

21.Tell me your project purpose?Which type of services it provided?

---------------NoAnswer------------------

22.Tell me your first project?What is it purpose?What is your role in that?Which module did u implement?

---------------NoAnswer------------------

23.Which build tool do u use?

---------------NoAnswer------------------

24.How did u fix bugs in ur project?

---------------NoAnswer------------------

25.Where is your office location?

---------------NoAnswer------------------

POLARIS (Telephonic interview )

-----------------------------------

1.Tell me briefly about yourself,What is your roles& Responsibilities in your project,Which technologies You

Worked on?

---------------NoAnswer------------------

2.How much knowledge do u have in core java?

---------------NoAnswer------------------

3 What are the oops princeples?

# **Java OOPs Concepts**

1. [Object Oriented Programming](https://www.javatpoint.com/java-oops-concepts#oops)
2. [Advantage of OOPs over Procedure-oriented programming language](https://www.javatpoint.com/java-oops-concepts#oopsadvantage)
3. [Difference between Objcet-oriented and Objcet-based programming language.](https://www.javatpoint.com/java-oops-concepts#oopsdifference)

In this page, we will learn about basics of OOPs. Object Oriented Programming is a paradigm that provides many concepts such as **inheritance**, **data binding**, **polymorphism** etc.

**Simula** is considered as the first object-oriented programming language. The programming paradigm where everything is represented as an object, is known as truly object-oriented programming language.

**Smalltalk** is considered as the first truly object-oriented programming language.

## OOPs (Object Oriented Programming System)



**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

## Object

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

## Class

**Collection of objects** is called class. It is a logical entity.

#### **Inheritance**

**When one object acquires all the properties and behaviours of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.



#### **Polymorphism**

When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

#### **Abstraction**

**Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.



#### **Encapsulation**

**Binding (or wrapping) code and data together into a single unit is known as encapsulation**. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

4.What are the types of inheritence?

# **Inheritance in Java**

1. [Inheritance](https://www.javatpoint.com/inheritance-in-java)
2. [Types of Inheritance](https://www.javatpoint.com/inheritance-in-java#inheritancetypes)
3. [Why multiple inheritance is not possible in java in case of class?](https://www.javatpoint.com/inheritance-in-java#inheritancenotmultiple)

**Inheritance in java** is a mechanism in which one object acquires all the properties and behaviors of parent object.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

Inheritance represents the **IS-A relationship**, also known as parent-child relationship.

### **Why use inheritance in java**

* For Method Overriding (so runtime polymorphism can be achieved).
* For Code Reusability.

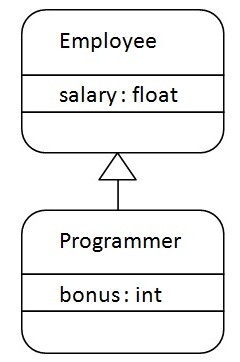
### **Syntax of Java Inheritance**

1. **class** Subclass-name **extends** Superclass-name
2. {
3. //methods and fields
4. }

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

### **Java Inheritance Example**



As displayed in the above figure, Programmer is the subclass and Employee is the superclass. Relationship between two classes is **Programmer IS-A Employee**.It means that Programmer is a type of Employee.

1. **class** Employee{
2. **float** salary=40000;
3. }
4. **class** Programmer **extends** Employee{
5. **int** bonus=10000;
6. **public** **static** **void** main(String args[]){
7. Programmer p=**new** Programmer();
8. System.out.println("Programmer salary is:"+p.salary);
9. System.out.println("Bonus of Programmer is:"+p.bonus);
10. }
11. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Programmer)

Programmer salary is:40000.0

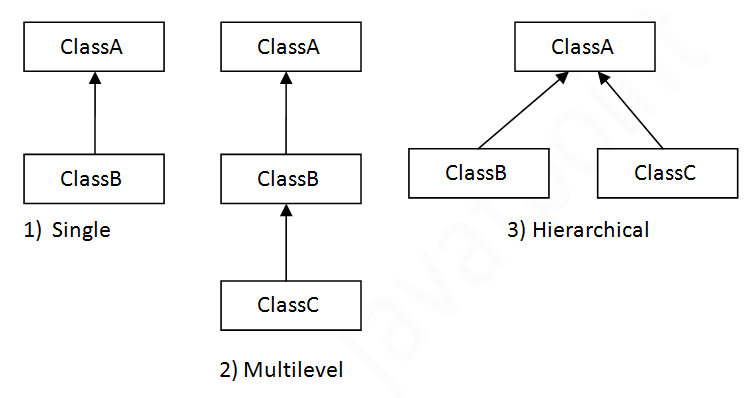
Bonus of programmer is:10000

In the above example, Programmer object can access the field of own class as well as of Employee class i.e. code reusability.

## Types of inheritance in java

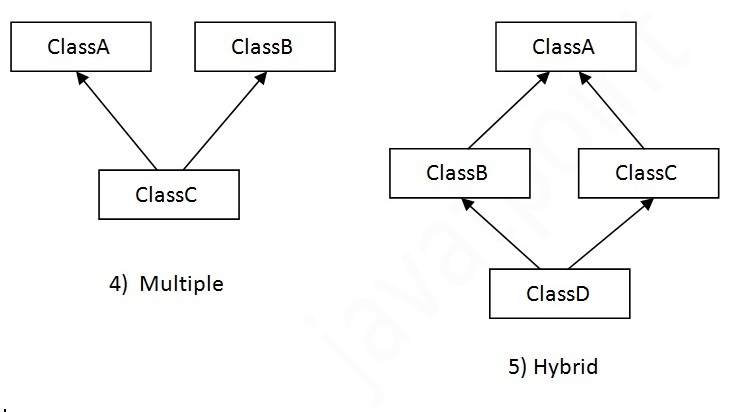
On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



#### Note: Multiple inheritance is not supported in java through class.

When a class extends multiple classes i.e. known as multiple inheritance. For Example:



## Single Inheritance Example

*File: TestInheritance.java*

1. **class** Animal{
2. **void** eat(){System.out.println("eating...");}
3. }
4. **class** Dog **extends** Animal{
5. **void** bark(){System.out.println("barking...");}
6. }
7. **class** TestInheritance{
8. **public** **static** **void** main(String args[]){
9. Dog d=**new** Dog();
10. d.bark();
11. d.eat();
12. }}

Output:

barking...

eating...

## Multilevel Inheritance Example

*File: TestInheritance2.java*

1. **class** Animal{
2. **void** eat(){System.out.println("eating...");}
3. }
4. **class** Dog **extends** Animal{
5. **void** bark(){System.out.println("barking...");}
6. }
7. **class** BabyDog **extends** Dog{
8. **void** weep(){System.out.println("weeping...");}
9. }
10. **class** TestInheritance2{
11. **public** **static** **void** main(String args[]){
12. BabyDog d=**new** BabyDog();
13. d.weep();
14. d.bark();
15. d.eat();
16. }}

Output:

weeping...

barking...

eating...

## Hierarchical Inheritance Example

*File: TestInheritance3.java*

1. **class** Animal{
2. **void** eat(){System.out.println("eating...");}
3. }
4. **class** Dog **extends** Animal{
5. **void** bark(){System.out.println("barking...");}
6. }
7. **class** Cat **extends** Animal{
8. **void** meow(){System.out.println("meowing...");}
9. }
10. **class** TestInheritance3{
11. **public** **static** **void** main(String args[]){
12. Cat c=**new** Cat();
13. c.meow();
14. c.eat();
15. //c.bark();//C.T.Error
16. }}

Output:

meowing...

eating...

5.What is the difference between abstract/ interface?Which scenarios we go for abstract/interface?

## Difference between abstract class and interface in Java

[When to use Abstract Class vs Interface in Java](http://2.bp.blogspot.com/-wrzDeQGAe1I/TWu8pLuLr4I/AAAAAAAAADE/V017G-6Q61w/s1600/java_logo_50_50.jpg)While deciding *when to use interface and abstract class*, it’s important to know difference between abstract class and interface in Java. In my opinion, following two differences between them drives decision about when to use abstract class or interface in Java.

1) Interface in Java can only contains declaration. You can not declare any concrete methods inside interface. On the other hand abstract class may contain both abstract and concrete methods, which makes abstract class an ideal place to provide common or default functionality. I suggest reading my post [10 things to know about interface in Java](http://javarevisited.blogspot.com/2012/04/10-points-on-interface-in-java-with.html) to know more about interfaces, particularly in Java programming language.

2) Java interface can extend multiple interface also Java class can implement multiple interfaces, Which means interface can provide more Polymorphism support than abstract class . By extending abstract class, a class can only participate in one Type hierarchy but by using interface it can be part of multiple type hierarchies. E.g. a class can be Runnable and Displayable at same time. One example I can remember of this is writing GUI application in J2ME, where  class extends Canvas and implements CommandListener to provide both graphic and event-handling functionality..

3) In order to implement interface in Java, until your class is abstract, you need to provide implementation of all methods, which is very painful. On the other hand abstract class may help you in this case by providing default implementation. Because of this reason, I prefer to have minimum methods in interface, starting from just one, I don't like idea of [marker interface](http://javarevisited.blogspot.com/2012/01/what-is-marker-interfaces-in-java-and.html), once annotation is introduced in Java 5. If you look JDK or any framework like Spring, which I does to understand OOPS and design patter better, you will find that most of interface contains only one or two methods e.g. Runnable, Callable, ActionListener etc.

I haven't included all syntactical difference between abstract class and interface in Java here, because focus here to learn when to use abstract class and interface and choosing one over other. Nevertheless you can see [difference between interface and abstract class](http://java67.blogspot.com/2012/09/what-is-difference-between-interface-abstract-class-java.html) to find  all those syntactical differences.

When to use interface and abstract class in Java

As I said earlier, it's easy to answer questions like difference between abstract class and interface in Java, but difficult to answer follow-ups. Though most of  Java Interview starts with former one, later it goes to see if you have really used abstract class and interface or not. In order to answer this question, you need to have good understanding of OOPS concepts

like [Polymorphism](http://javarevisited.blogspot.com/2011/08/what-is-polymorphism-in-java-example.html), [Encapsulation](http://javarevisited.blogspot.com/2012/03/what-is-encapsulation-in-java-and-oops.html), [Abstraction](http://javarevisited.blogspot.com/2010/10/abstraction-in-java.html) and [Inheritance](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html). Also familiarity with coupling and cohesion is important. You at least should know that effort of designing should lead to reduce coupling and increased cohesion, ease of maintenance etc. In this part, we will see some scenarios, guidelines, rules which can help you to decide *when to use abstract class and interface in Java*.

1) In Java particularly, decision between choosing Abstract class and interface may influence by the fact that multiple inheritance is not supported in Java. One class can only extend another class in Java. If you choose abstract class over interface than you lost your chance to extend another class, while at the same time you can implement multiple interfaces to show that you have multiple capability. One of the common example, in favor of interface over abstract class is [Thread vs Runnable](http://javarevisited.blogspot.com/2012/01/difference-thread-vs-runnable-interface.html) case. If you want to execute a task and need run() method it's better to implement Runnable interface than extending Thread class.

2) Let's see another case where an abstract class suits better than interface. Since abstract class can include concrete methods, it’s great for maintenance point of view, particularly when your base class is evolving and keep changing. If you need a functionality across all your implementation e.g. a common method, than, you need to change every single implementation to include that change if  you have chosen interface to describe your base class. Abstract class comes handy in this case because you can just define new functionality in abstract super class and every sub class will automatically gets it. In short, abstract class are great in terms of evolving functionality. If you are using interface, you need to exercise extra care while defining contracts because its not easy to change them once published.

3) Interface in Java is great for defining Types. Programming for interfaces than implementation is also one of the useful [Object oriented design principle](http://javarevisited.blogspot.de/2012/03/10-object-oriented-design-principles.html) which suggests benefit of using interface as argument to function, return type etc.

4) One more general rule of when to use abstract class and interface is to find out whether a certain class will form a IS-A hierarchy or CAN-DO-THIS hierarchy. If you know that you will be creating classes e.g. Circle, Square than it's better to create an abstract class Shape which can have area() and perimeter() as abstract method, rather than defining Shape as interface in Java. On the other hand if you are going to create classes which can do thinks like, can fly, you can use interface Flyable instead of abstract class.

5) Interface generally define capability

e.g. Runnable can run(), Callable can call(), Displayable can display(). So if you need to define capability, consider using interface. Since a class can have multiple capabilities i.e. a class can be Runnable as well as Displayable at same time. As discussed in first point, Since [java does not allow multiple inheritance at class level](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html), only way to provide multiple capability is via interfaces.

6) Let's see another example of where to use Abstract class and Interface in Java, which is related to earlier point. Suppose you have lot of classes to model which are birds, which can fly, than creating a base abstract class as Bird would be appropriate  but if you have to model other things along with Birds, which can fly e.g. Airplanes, Balloons or Kites than it's better to create interface Flyable to represent flying functionality. In conclusion, if you need to provide a functionality which is used by same type of class than use Abstract class and if functionality can be used by completely unrelated classes than use interface.

7) Another interesting use of Abstract class and interface is defining contract using interface and providing skeletal using abstract class. java.util.List from Java collection framework is a good example of this pattern. List is declared as interface and extends Collection and Iterable interface and AbstractList is an abstract class which implements List. AbstractList provides skeletal implementation of List interface. Benefit of using this approach is that it minimize the effort to implement this interface by concrete class e.g. [ArrayList or LinkedList](http://java67.blogspot.com/2012/12/difference-between-arraylist-vs-LinkedList-java.html). If you don't use skeletal implementation e.g. abstract class and instead decide to implement List interface than not only you need to implement all List methods but also you might be duplicating common code. Abstract class in this case reduce effort to implement interface.

8) Interface also provide more decoupling than abstract class because interface doesn't contain any implementation detail, while abstract class may contain default implementation which may couple them with other class or resource.

9) Using interface also help while implementing [Dependency Injection design pattern](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html) and makes testing easy. Many mock testing framework utilize this behavior.

That's all on **When to use Abstract class and interface in Java**. Though discussion here is centered around Java but given concept of abstract class and interface goes beyond Java and also applicable to other Object oriented language, some of the tips are also applicable to other OOPS languages.

6.Which collection do u use in ur project?

---------------NoAnswer------------------

7.What is the parent class for all java classes?

Object….

8.Difference betweem Arraylist and HashMap?

[**ArrayList**](http://beginnersbook.com/2013/12/java-arraylist/) and [**HashMap**](http://beginnersbook.com/2013/12/hashmap-in-java-with-example/) are two commonly used collection classes in Java. Even though both are the part of collection framework, the way they store and process the data is entirely different. In this post we will see the main differences between these two collections.

## ArrayList vs HashMap in Java

1) **Implementation**: **ArrayList** implements List Interface while **HashMap** is an implementation of Map interface. List and Map are two entirely different collection interfaces.

2) **Memory consumption**: ArrayList stores the element’s value alone and internally maintains the indexes for each element.

ArrayList<String> arraylist =newArrayList<String>();

//String value is stored in array list

arraylist.add("Test String");

HashMap stores key & value pair. For each value there must be a key associated in HashMap. That clearly shows that memory consumption is high in HashMap compared to the ArrayList.

HashMap<Integer,String> hmap=newHashMap<Integer,String>();

//String value stored along with the key value in hash map

hmap.put(123,"Test String");

3) **Order**: ArrayList maintains the insertion order while HashMap doesn’t. Which means ArrayList returns the list items in the same order in which they got inserted into the list. On the other side HashMap doesn’t maintain any order, the returned key-values pairs are not sorted in any kind of order.

4) **Duplicates**: ArrayList allows duplicate elements but HashMap doesn’t allow duplicate keys (It does allow duplicate values).

5) **Nulls**: ArrayList can have any number of null elements. HashMap allows one null key and any number of null values.

6) **get method**: In ArrayList we can **get** the element by specifying the index of it. In HashMap the elements is being fetched by specifying the corresponding key.

#### References:

* [**HashMap javadoc**](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html)
* [**ArrayList Documentation**](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html)

9.HOw to avoid duplicates in String?

Here is my solution for the problem of removing repeated or duplicate characters from given String in Java programming language. If you understand the logic you can write this solution in any programming language e.g. C, C++, C#, Python or JavaScript.  
  
/\*\*

\* Java Program to remove duplicate characters from String.

\*

\* @author Javin Paul

\*/

publicclassRemoveDuplicateCharacters{

publicstaticvoidmain(String args[]) {

System.out.println("Call removeDuplicates(String word) method ....");

String[] testdata = {"aabscs", "abcd", "aaaa", null, "",

"aaabbb", "abababa"};

for (String input : testdata) {

System.out.printf("Input : %s Output: %s %n",

input, removeDuplicates(input));

}

System.out.println("Calling removeDuplicatesFromString(String str).");

for (String input : testdata) {

System.out.printf("Input : %s Output: %s %n",

input, removeDuplicatesFromString(input));

}

}

/\*

\* This algorithm goes through each character of String to check if its

\* a duplicate of already found character. It skip the duplicate

\* character by inserting 0, which is later used to filter those

\* characters and update the non-duplicate character.

\* Time Complexity of this solution is O(n^2), excluded to

\* UniqueString() method, which creates String from character array.

\* This method will work even if String contains more than one duplicate

\* character.

\*/

publicstaticStringremoveDuplicates(String word) {

if (word ==null|| word.length() <2) {

return word;

}

int pos =1; // possible position of duplicate character

char[] characters = word.toCharArray();

for (int i =1; i < word.length(); i++) {

int j;

for (j =0; j < pos; ++j) {

if (characters[i] == characters[j]) {

break;

}

}

if (j == pos) {

characters[pos] = characters[i];

++pos;

} else {

characters[pos] =0;

++pos;

}

}

return toUniqueString(characters);

}

/\*

\* This solution assumes that given input String only contains

\* ASCII characters. You should ask this question to your Interviewer,

\* if he says ASCII then this solution is Ok. This Algorithm

\* uses additional memory of constant size.

\*/

publicstaticStringremoveDuplicatesFromString(String input) {

if (input ==null|| input.length() &lt; 2) {

return input;

}

boolean[] ASCII=newboolean[256];

char[] characters = input.toCharArray();

ASCII[input.charAt(0)] =true;

int dupIndex =1;

for (int i =1; i &lt; input.length(); i++) {

if (!ASCII[input.charAt(i)]) {

characters[dupIndex] = characters[i];

++dupIndex;

ASCII[characters[i]] =true;

} else {

characters[dupIndex] =0;

++dupIndex;

}

}

return toUniqueString(characters);

}

/\*

\* Utility method to convert Character array to String, omitting

\* NUL character, ASCII value 0.

\*/

publicstaticStringtoUniqueString(char[] letters) {

StringBuilder sb =newStringBuilder(letters.length);

for (char c : letters) {

if (c !=0) {

sb.append(c);

}

}

return sb.toString();

}

}

Output

Call removeDuplicates(String word) method ....

Input: aabscs Output: absc

Input: abcd Output: abcd

Input: aaaa Output: a

Input:nullOutput:null

Input:Output:

Input: aaabbb Output: ab

Input: abababa Output: ab

Calling removeDuplicatesFromString(String str) method ....

Input: aabscs Output: absc

Input: abcd Output: abcd

Input: aaaa Output: a

Input:nullOutput:null

Input:Output:

Input: aaabbb Output: ab

Input: abababa Output: ab

10.What are the methods present in Object class?

The **Object class** is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

The Object class is beneficial if you want to refer any object whose type you don't know. Notice that parent class reference variable can refer the child class object, know as upcasting.

Let's take an example, there is getObject() method that returns an object but it can be of any type like Employee,Student etc, we can use Object class reference to refer that object. For example:

1. Object obj=getObject();//we don't know what object will be returned from this method

The Object class provides some common behaviors to all the objects such as object can be compared, object can be cloned, object can be notified etc.



### **Methods of Object class**

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

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

We will have the detailed learning of these methods in next chapters.

11.What is the Hash map?

Java HashMap class implements the map interface by using a hashtable. It inherits AbstractMap class and implements Map interface.

The important points about Java HashMap class are:

* A HashMap contains values based on the key.
* It contains only unique elements.
* It may have one null key and multiple null values.
* It maintains no order.

### **Hierarchy of HashMap class**

As shown in the above figure, HashMap class extends AbstractMap class and implements Map interface.

### **HashMap class declaration**

Let's see the declaration for java.util.HashMap class.

1. **public** **class** HashMap<K,V> **extends** AbstractMap<K,V> **implements** Map<K,V>, Cloneable, Serializable

### **HashMap class Parameters**

Let's see the Parameters for java.util.HashMap class.

* **K**: It is the type of keys maintained by this map.
* **V**: It is the type of mapped values.

### **Constructors of Java HashMap class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| HashMap() | It is used to construct a default HashMap. |
| HashMap(Map m) | It is used to initializes the hash map by using the elements of the given Map object m. |
| HashMap(int capacity) | It is used to initializes the capacity of the hash map to the given integer value, capacity. |
| HashMap(int capacity, float fillRatio) | It is used to initialize both the capacity and fill ratio of the hash map by using its arguments. |

### **Methods of Java HashMap class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void clear() | It is used to remove all of the mappings from this map. |
| boolean containsKey(Object key) | It is used to return true if this map contains a mapping for the specified key. |
| boolean containsValue(Object value) | It is used to return true if this map maps one or more keys to the specified value. |
| boolean isEmpty() | It is used to return true if this map contains no key-value mappings. |
| Object clone() | It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned. |
| Set entrySet() | It is used to return a collection view of the mappings contained in this map. |
| Set keySet() | It is used to return a set view of the keys contained in this map. |
| Object put(Object key, Object value) | It is used to associate the specified value with the specified key in this map. |
| int size() | It is used to return the number of key-value mappings in this map. |
| Collection values() | It is used to return a collection view of the values contained in this map. |

### **Java HashMap Example**

1. **import** java.util.\*;
2. **class** TestCollection13{
3. **public** **static** **void** main(String args[]){
4. HashMap<Integer,String> hm=**new** HashMap<Integer,String>();
5. hm.put(100,"Amit");
6. hm.put(101,"Vijay");
7. hm.put(102,"Rahul");
8. **for**(Map.Entry m:hm.entrySet()){
9. System.out.println(m.getKey()+" "+m.getValue());
10. }
11. }
12. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestCollection13)

Output:102 Rahul

100 Amit

101 Vijay

### **Java HashMap Example: remove()**

1. **import** java.util.\*;
2. **public** **class** HashMapExample {
3. **public** **static** **void** main(String args[]) {
4. // create and populate hash map
5. HashMap<Integer, String> map = **new** HashMap<Integer, String>();
6. map.put(101,"Let us C");
7. map.put(102, "Operating System");
8. map.put(103, "Data Communication and Networking");
9. System.out.println("Values before remove: "+ map);
10. // Remove value for key 102
11. map.remove(102);
12. System.out.println("Values after remove: "+ map);
13. }
14. }

Output:

Values before remove: {102=Operating System, 103=Data Communication and Networking, 101=Let us C}

Values after remove: {103=Data Communication and Networking, 101=Let us C}

### **Difference between HashSet and HashMap**

HashSet contains only values whereas HashMap contains entry(key and value).

### **Java HashMap Example: Book**

1. **import** java.util.\*;
2. **class** Book {
3. **int** id;
4. String name,author,publisher;
5. **int** quantity;
6. **public** Book(**int** id, String name, String author, String publisher, **int** quantity) {
7. **this**.id = id;
8. **this**.name = name;
9. **this**.author = author;
10. **this**.publisher = publisher;
11. **this**.quantity = quantity;
12. }
13. }
14. **public** **class** MapExample {
15. **public** **static** **void** main(String[] args) {
16. //Creating map of Books
17. Map<Integer,Book> map=**new** HashMap<Integer,Book>();
18. //Creating Books
19. Book b1=**new** Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
20. Book b2=**new** Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);
21. Book b3=**new** Book(103,"Operating System","Galvin","Wiley",6);
22. //Adding Books to map
23. map.put(1,b1);
24. map.put(2,b2);
25. map.put(3,b3);
27. //Traversing map
28. **for**(Map.Entry<Integer, Book> entry:map.entrySet()){
29. **int** key=entry.getKey();
30. Book b=entry.getValue();
31. System.out.println(key+" Details:");
32. System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
33. }
34. }
35. }

Output:

1 Details:

101 Let us C Yashwant Kanetkar BPB 8

2 Details:

102 Data Communications & Networking Forouzan Mc Graw Hill 4

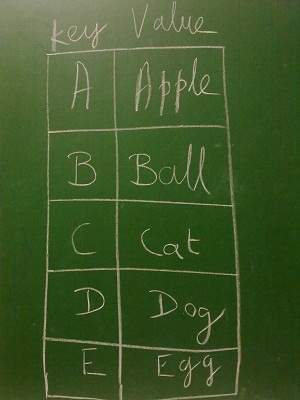
3 Details:

103 Operating System Galvin Wiley 6

12.How hashtable techinique will work?(He asked me no of questions about these topic)

hashtable is an implementation of a key-value pair data structure in java. You can store and retrieve a ‘value’ using a ‘key’ and it is [an identifier](http://javapapers.com/core-java/how-many-types-of-java-variables-are-there/) of the value stored. It is obvious that the ‘key’ should be unique.

java.util.Hashtable extends Dictionary and implements Map. Objects with non-[null value](http://javapapers.com/core-java/nullpointerexception-null-bad-good-and-ugly/) can be used as a key or value. Key of the Hashtable must implement hashcode() and equals() methods. By the end of this article you will find out the reason behind this condition.



Generally a Hashtable in java is created using the empty [constructor](http://javapapers.com/core-java/java-double-brace-initialization/)Hashtable(). Which is a poor decision and an often repeated mistake. Hashtable has two other constructors

Hashtable(int initialCapacity)

and

Hashtable(int initialCapacity,float loadFactor)

. Initial capacity is number of buckets created at the time of Hashtable instantiation. Bucket is a logical space of [storage](http://javapapers.com/core-java/java-jvm-memory-types/)for Hashtable.

## Hashing and Hashtable

Before seeing java’s Hashtable in detail you should understand hashing in general. Assume that, v is a value to be stored and k is the key used for storage / retrieval, then h is a hash function where v is stored at h(k) of table. To retrieve a value compute h(k) so that you can directly get the position of v. So in a key-value pair table, you need not sequentially scan through the keys to identify a value.

h(k) is the hashing function and it is used to find the location to store the corresponding value v. h(k) cannot compute to a indefinite space. Storage allocated for a Hashtable is limited within a program. So, the hasing function h(k) should return a number within that allocated spectrum ([logical address space](http://javapapers.com/core-java/java-jvm-memory-types/)).

## Hashing in Java

Java’s hashing uses uses hashCode() method from the key and value objects to compute. Following is the core code from Hashtable where the hashCode ‘h’ is computed. You can see that both key’s and value’s hashCode() method is called.

h += e.key.hashCode()^ e.value.hashCode();

It is better to have your hashCode() method in your custom objects. [String](http://javapapers.com/core-java/java-string-concatenation/)has its own hashCode methode and it computes the hashcode value as below:

s[0]\*31^(n-1)+ s[1]\*31^(n-2)+...+ s[n-1]

If you don’t have a hashCode() method, then it is derived from Object class. Following is [javadoc comment](http://javapapers.com/core-java/is-javadoc-comment-a-type-of-standard-java-comment/) of hashCode() method from Object class:

*Returns a hash code value for the object. This method is supported for the benefit of hashtables such as those provided by java.util.Hashtable.*

If you are going to write a custom hashCode(), then follow the following contract:

*The general contract of hashCode is: Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified.*

The following is to improve performance of the Hashtable.

*If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.*

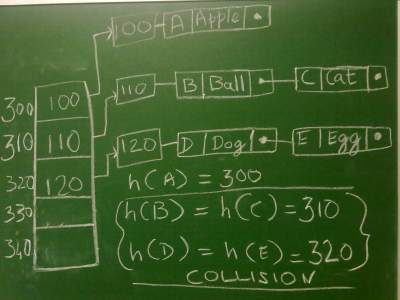
hashCode() guarantees distinct integers by using the internal address of the object.

## Collision in Hashtable

When we try to restrict the hashing function’s output within the allocated address spectrue limit, there is a possibility of a collision. For two different keys k1 and k2, if we have h(k1) = h(k2), then this is called collision in hashtable. What does this mean, our hashing function directs us store two different values (keys are also different) in the same location.

When we have a collision, there are multiple methodologies available to resolve it. To name a few hashtable collision resolution technique, ‘separate chaining’, ‘open addressing’, ‘robin hood hashing’, ‘cuckoo hashing’, etc. Java’s hashtable uses ‘separate chaining’ for collision resolution in Hashtable.

## Collision Resolution in java’s Hashtable

Java uses separate chaining for collision resolution. Recall a point that Hashtable stores elements in buckets. In separate chaining, every bucket will store a reference to a linked list. Now assume that you have stored an element in bucket 1. That means, in bucket 1 you will have a reference to a linked list and in that linked list you will have two cells. In those two cells you will have key and its corresponding value.

Why do you want to store the key? Because when there is a collision i.e., when two keys results in same hashcode and directs to the same bucket (assume bucket 1) you want to store the second element also in the same bucket. You add this second element to the already created linked list as the adjacent element.

Now when you retrieve a value it will compute the hash code and direct you to a bucket which has two elements. You scan those two elements alone sequentially and compare the keys using their equals() method. When the key mathches you get the respective value. Hope you have got the reason behind the condition that your object must have hashCode() and equals() method.

Java has a private static class Entry inside Hashtable. It is an implementation of a list and you can see there, it stores both the key and value.

## Hashtable performance

To get better performance from your java Hashtable, you need to  
1) use the initialCapacity and loadFactor arguments  
2) use them wisely  
while instantiating a Hashtable.

initialCapacitiy is the number of buckets to be created at the time of Hashtable instantiation. The number of buckets and probability of collision is inversly proportional. If you have more number of buckets than needed then you have lesser possibility for a collision.

For example, if you are going to store 10 elements and if you are going to have initialCapacity as 100 then you will have 100 buckets. You are going to calculate hashCoe() only 10 times with a spectrum of 100 buckets. The possibility of a collision is very very less.

But if you are going to supply initialCapacity for the Hashtable as 10, then the possibility of collision is very large. loadFactor decides when to automatically increase the size of the Hashtable. The default size of initialCapacity is 11 and loadFactor is .75 That if the Hashtable is 3/4 th full then the size of the Hashtable is increased.

New capacity in java Hashtable is calculated as follows:

int newCapacity = oldCapacity \*2+1;

If you give a lesser capacity and loadfactor and often it does the rehash() which will cause you performance issues. Therefore for efficient performance for Hashtable in java, give initialCapacity as 25% extra than you need and loadFactor as 0.75 when you instantiate.

13.How hashing mechanisim will work create same hashcode objects?How it will create?(He gave me one scenario also)

----------------------NoAnswer---------------------------------

14.How to write your own customized exception?

If you are creating your own Exception that is known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need.

By the help of custom exception, you can have your own exception and message.

Let's see a simple example of java custom exception.

1. **class** InvalidAgeException **extends** Exception{
2. InvalidAgeException(String s){
3. **super**(s);
4. }
5. }
6. **class** TestCustomException1{
8. **static** **void** validate(**int** age)**throws** InvalidAgeException{
9. **if**(age<18)
10. **throw** **new** InvalidAgeException("not valid");
11. **else**
12. System.out.println("welcome to vote");
13. }
15. **public** **static** **void** main(String args[]){
16. **try**{
17. validate(13);
18. }**catch**(Exception m){System.out.println("Exception occured: "+m);}
20. System.out.println("rest of the code...");
21. }
22. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestCustomException1)

Output:Exception occured: InvalidAgeException:not valid

rest of the code...

15.Tell me your project flow which technologies do u use in your project?

---------------NoAnswer------------------

16.What is spring-hibenate integreation?

We can simply integrate **hibernate application with spring application**.

In hibernate framework, we provide all the database information hibernate.cfg.xml file.

But if we are going to integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file. We can provide all the information in the applicationContext.xml file.

### **Advantage of Spring framework with hibernate**

The Spring framework provides **HibernateTemplate** class, so you don't need to follow so many steps like create Configuration, BuildSessionFactory, Session, beginning and committing transaction etc.

So **it saves a lot of code**.

**Understanding problem without using spring:**

Let's understand it by the code of hibernate given below:

1. //creating configuration
2. Configuration cfg=**new** Configuration();
3. cfg.configure("hibernate.cfg.xml");
5. //creating seession factory object
6. SessionFactory factory=cfg.buildSessionFactory();
8. //creating session object
9. Session session=factory.openSession();
11. //creating transaction object
12. Transaction t=session.beginTransaction();
14. Employee e1=**new** Employee(111,"arun",40000);
15. session.persist(e1);//persisting the object
17. t.commit();//transaction is commited
18. session.close();

As you can see in the code of sole hibernate, you have to follow so many steps.

**Solution by using HibernateTemplate class of Spring Framework:**

Now, you don't need to follow so many steps. You can simply write this:

1. Employee e1=**new** Employee(111,"arun",40000);
2. hibernateTemplate.save(e1);

17.Tell me hibernate spring mapping?(Here ORM mapping)

------------------NoAnswer--------------------

18.Tell me how many interfaces present servlet?

------------------NoAnswer--------------------

19.How to maintain session in web application?

--------------------NoAnswer---------------

20.What is the maven?

-----------Repeated------

21.How to work with mockito?

Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing. This tutorial should help you learn how to create unit tests with Mockito as well as how to use its APIs in a simple and intuitive way.

22.What is the IOC module?

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans, which we will discuss in the next chapter.

The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.



Spring provides the following two distinct types of containers.

|  |  |
| --- | --- |
| **Sr.No.** | **Container & Description** |
| 1 | [**Spring BeanFactory Container**](https://www.tutorialspoint.com/spring/spring_beanfactory_container.htm)  This is the simplest container providing the basic support for DI and is defined by the *org.springframework.beans.factory.BeanFactory* interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purpose of backward compatibility with a large number of third-party frameworks that integrate with Spring. |
| 2 | [**Spring ApplicationContext Container**](https://www.tutorialspoint.com/spring/spring_applicationcontext_container.htm)  This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the *org.springframework.context.ApplicationContext* interface. |

23.What is copyonWriteArray class?

CopyOnWriteArrayList is a concurrent Collection class introduced in Java 5 Concurrency API. CopyOnWriteArrayList implements List interface like, ArrayList, Vector, and LinkedList but it is thread-safe collection and it achieves its thread-safety in a slightly different way than Vector and other thread-safe Collection class. As name suggests CopyOnWriteArrayList creates copy of underlying ArrayList with every mutation operation like add or set. ArrayList class is fail-fast which means that if the ArrayList will be changed while some thread is traversing over it using iterator, the iterator.next() will throw a ConcurrentModificationException. But Iterator of CopyOnWriteArrayList is fail-safe and doesn’t throw ConcurrentModificationException even if underlying CopyOnWriteArrayList is modified once Iteration begins because Iterator is operating on separate copy of ArrayList. Consequently all the updates made on CopyOnWriteArrayList is not available to Iterator. Let have a simple Example on ArrayList ConcurrentModificationException:

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class ArrayListIterator {

public static void main(String[] args) {

List myList = new ArrayList();

myList.add("1");

myList.add("2");

myList.add("3");

myList.add("4");

myList.add("5");

Iterator it = myList.iterator();

while (it.hasNext()) {

String value = it.next();

System.out.println("List Value:" + value);

//if(value.equals("3")) myList.remove(value);

}

}

}

24.Can non static inner class call static method?

--------------NoAnswer---------------

25.Difference between comparable and comparator?

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

But there are many differences between Comparable and Comparator interfaces that are given below.

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by **Collections.sort(List)** method. | We can sort the list elements of Comparator type by **Collections.sort(List,Comparator)** method. |

**1.DelloitteInterview Questions**

Q1.Do u know servlet write login page?How servlet take the request from the login page?(he gave me one scenario)?

Ans: Refer JavaTPoint =====>[https://www.javatpoint.com/example-of-registration-form-in-servlet](#_top)

Q3..What is synchronized statement?

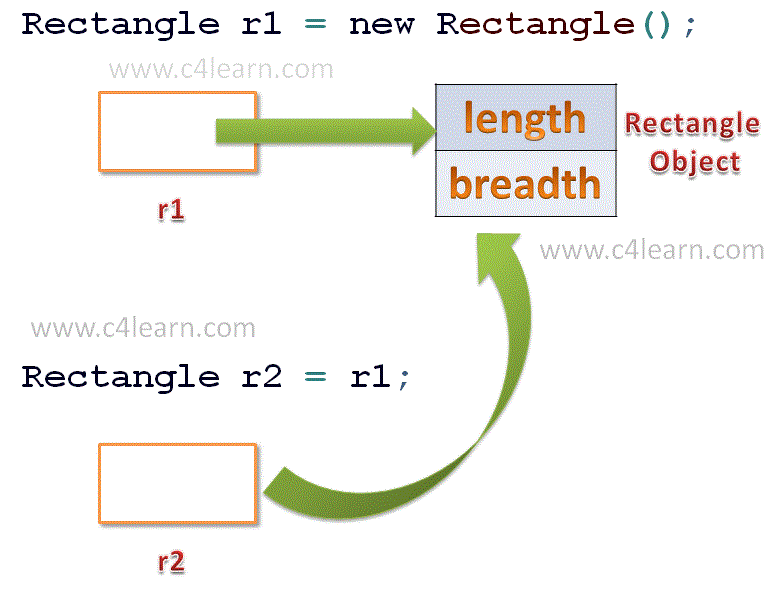
Ans:  Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement

Q4..How many types of veribles is there?

The **eight** primitive data types are: byte, short, int, long, float, double, boolean, and char.

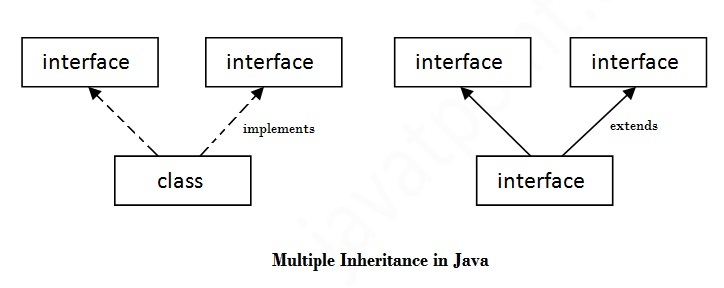
Q5..What is reference variable in java?

A **reference variable** is declared to be of a specific type and that type can never be changed. **Reference variables** can be declared as static **variables**, instance **variables**, method parameters, or local **variables**. A **reference variable** that is declared as final can't never be reassigned to refer to a different object.



6. How to achieve mulitple inheretence in java?

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.



7.Insted of interface Is there any way we can achieve mulitple inheretence in java?

Nooooooooooo

8.What is clusteing & load balancing?

**Clustering**has a formal meaning. A cluster is a group of resources that are trying to achieve a common objective, and are aware of one another. Clustering usually involves setting up the resources (servers usually) to exchange details on a particular channel (port) and keep exchanging their states, so a resource’s state is replicated at other places as well. It usually also includes load balancing, wherein, the request is routed to one of the resources in the cluster as per the load balancing policy.

**Load balancing** can also happen without clustering when we have multiple independent servers that have same setup, but other than that, are unaware of each other. Then, we can use a load balancer to forward requests to either one server or other, but one server does not use the other server’s resources. Also, one resource does not share its state with other resources.

9.What is singleton?Write a program?

Singleton Pattern says that just**"define a class that has only one instance and provides a global point of access to it".**

In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.

There are two forms of singleton design pattern

**Early Instantiation:** creation of instance at load time.

**Lazy Instantiation:** creation of instance when required.

classSingleton

{

    privatestaticSingleton obj;

    // private constructor to force use of

    // getInstance() to create Singleton object

    privateSingleton() {}

    publicstaticSingleton getInstance()

    {

        if(obj==null)

            obj = newSingleton();

        returnobj;

    }

}

10.What is class loader?

ClassLoader in Java is a class which is used to load [class files in Java](http://javarevisited.blogspot.ca/2012/05/10-points-about-class-file-in-java.html). Java code is compiled into class file by [javac](http://javarevisited.blogspot.sg/2012/12/javac-is-not-recognized-as-internal-or-external-command.html)compiler and [JVM](http://javarevisited.blogspot.sg/2011/12/jre-jvm-jdk-jit-in-java-programming.html)executes Java program, by executing byte codes written in class file. ClassLoader is responsible for loading class files from file system, network or any other source. There are three default class loader used in Java, **Bootstrap** , **Extension** and **System or Application class loader**.   
  
12.Where class loader located?

The Java **Classloader** is a part of the Java Runtime Environment that dynamically loads Java ... The system **class loader** loads code **found** on java.class.path , which maps to the CLASSPATH environment variable.

13.What we call "area" in jvm? (Ans:memory)

14.How many types of memories is there?

There are **two** main kinds of semiconductor memory, volatile and non-volatile. Examples of non-volatile memory are flash memory (used as secondary memory) and ROM, PROM, EPROM and EEPROM memory (used for storing firmware such as BIOS).

15.Where static veribles stored in java?

In The Class Area..

16.Difference between Arraylist and vector?

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| 1) ArrayList is **not synchronized**. | Vector is **synchronized**. |
| 2) ArrayList **increments 50%** of current array size if number of element exceeds from its capacity. | Vector **increments 100%** means doubles the array size if total number of element exceeds than its capacity. |
| 3) ArrayList is **not a legacy** class, it is introduced in JDK 1.2. | Vector is a **legacy** class. |
| 4) ArrayList is **fast** because it is non-synchronized. | Vector is **slow** because it is synchronized i.e. in multithreading environment, it will hold the other threads in runnable or non-runnable state until current thread releases the lock of object. |
| 5) ArrayList uses **Iterator** interface to traverse the elements. | Vector uses **Enumeration** interface to traverse the elements. But it can use Iterator also. |

Java Synchronization is better option where we want to allow only one thread to access the shared resource.

18.class Demo{ (Ans:we ll get NullPointer exception)

int i=10;

static int j=20;

public static void main(String args[]){

Demo d=new Demo();

d=null;

Syso(d.i);

Syso(d.j);

Syso(Demo.j);

} }

Q)What is output of these program?It will compile or not?

19.which server do u use in project?

20.What is forward &send redirect?Give me example?(He gava me one scenario tell me where we use

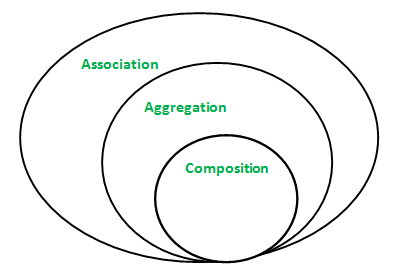
forward &send redirect )

|  |  |
| --- | --- |
| **forward() method** | **sendRedirect() method** |
| The forward() method works at server side. | The sendRedirect() method works at client side. |
| It sends the same request and response objects to another servlet. | It always sends a new request. |
| It can work within the server only. | It can be used within and outside the server. |
| Example: request.getRequestDispacher("servlet2").forward(request,response); | Example: response.sendRedirect("servlet2"); |

21.Which technology we use to get drop down box ?(He gava me one scenario related to irctc application)

22.What is association in java?Where do u use in ur priject?

Association is relation between two separate classes which establishes through their Objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many.  
In Object-Oriented programming, an Object communicates to other Object to use functionality and services provided by that object. **Composition** and **Aggregation** are the two forms of association.

[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/AssociationAggregation-and-Composition.png)

23.What is externilization?

Externalizable is a sub-interface to Serializable but it is not a marker interface because it has two unimplemented methods readExternal() and writeExternal() which should be implemented by the classes which use Externalizable interface.

Manager round(F2F)

----------------------

1.How much rating will u give in core java? (U should tell 8(OR) 9/10 ::(Here they check confidence levels))

2.Where do u work?

3.What is the reason ur changing company?

4.How many years of experience do u hava?

5.Is ur project development or maintainence?

**Accenture(Telephonic-1st round)**

1.What is differnt types of class loaders?

Class loaders are the part of the Java Runtime Environment that dynamically loads Java classes into the Java virtual machine. It is responsible for locating libraries, reading there content and loading the classes contained within the libraries. When JVM is started three class loaders are used

**Bootstrap Class Loader**

Bootstrap class loader loads java’s core classes like java.lang, java.util etc. These are classes that are part of java runtime environment. Bootstrap class loader is native implementation and so they may differ across different JVMs.

**Extensions Class Loader**

JAVA\_HOME/jre/lib/ext contains jar packages that are extensions of standard core java classes. Extensions class loader loads classes from this ext folder. Using the system environment propery java.ext.dirs you can add ‘ext’ folders and ja**r files to be loaded using extensions class loader.**

System Class Loader

Java classes that are available in the java classpath are loaded using System class loader.

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| 1) HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code. | Hashtable is **synchronized**. It is thread-safe and can be shared with many threads. |
| 2) HashMap **allows one null key and multiple null values**. | Hashtable **doesn't allow any null key or value**. |
| 3) HashMap is a **new class introduced in JDK 1.2**. | Hashtable is a **legacy class**. |
| 4) HashMap is **fast**. | Hashtable is **slow**. |
| 5) We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap); | Hashtable is internally synchronized and can't be unsynchronized. |
| 6) HashMap is **traversed by Iterator**. | Hashtable is **traversed by Enumerator and Iterator**. |
| 7) Iterator in HashMap is **fail-fast**. | Enumerator in Hashtable is **not fail-fast**. |
| 8) HashMap inherits **AbstractMap** class. | Hashtable inherits **Dictionary** class. |

2.Difference between hashmap and hashtable?

3.Difference between jdk1.4 and jdk 1.5?

**New features in Jdk 1.4**

* XML Processing
* Java Print Service
* Logging API
* Java Web Start
* JDBC 3.0 API
* Assertions
* Preferences API
* Chained Exception
* IPv6 Support
* Regular Expressions
* Image I/O API
* Linked Hashmap

###### JDK 1.5 Features

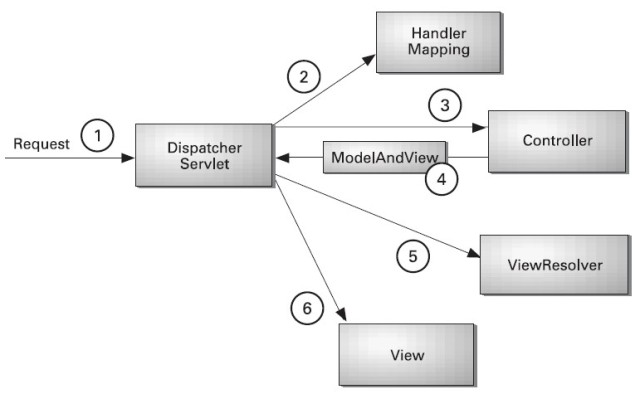
1. Autoboxing
2. Generics
3. Enhanced for loop
4. Varargs
5. Enums
6. Static imports
7. C-lang printf()
8. StringBuilder
9. Metadata

4.What is Queue?

In java Queue interface orders the element in FIFO(First In First Out) manner. In FIFO, first element is removed first and last element is removed at last.

5.Explain flow of spring mvc?

**Spring MVC Flow**



* Based on the Servlet Mappings which we provide in our web.xml, the request will be routed by the Servlet Container to our DispatcherServlet
* Once the request is received, the DispatcherServlet will take the help of HandlerMapping which has been added in the [**Spring**](https://www.javainterviewpoint.com/category/spring/) Configuration file and get to know the Controller class to be called for the request received.
* Now the request will get transfered to the Controller, the Controller then executes the appropriate methods and returns the corresponding ModelAndView object to the DispatcherServlet.
* The DispatcherServlet will send the Model received to the ViewResolver to get the view page.
* Finally the DispatcherServlet will pass the Model to the View page and the page will be rendered to the user

6)Where do u configure internelResource view resolver?

Spring Configuration file(spring.xml)

7.How to cofigure suffix and prefix?

<bean id="viewResolver"class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix"><value>/WEB-INF/pages/</value></property>

<property name="suffix"><value>.jsp</value></property>

</bean>

8.What is transaction management?

A transaction is a unit of work in which either all operations must execute or none of them. To understand the importance of the transaction, think of an example which applies to all of us. “Transferring Amount from one account to another “ – this operation includes below at least below two steps

1. Deduct the balance from the sender’s account
2. Add the amount to the receiver’s account.

Now think of the situation where the amount is deducted from the sender’s account, but not gets delivered to receiver account due to some errors. Such issues are managed by transaction management in which both the steps are performed in a single unit of work where either both steps are performed successfully or in case anyone gets failed, it should be roll backed.

9.What is @autowired annatation?

Spring @Autowired annotation is used for automatic dependency injection. [**Spring framework**](http://www.journaldev.com/2888/spring-tutorial-spring-core-tutorial) is built on [dependency injection](http://www.journaldev.com/2410/spring-dependency-injection) and we inject the class dependencies through spring bean configuration file.

10.What is cache levels ?

There are mainly two types of caching: first level cache and second level cache.

#### **First Level Cache**

Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to entire application. An application can use many session object.

#### **Second Level Cache**

SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.

DBS(Telephonic-1st round)

-----------------------------

1.How much rating will give in core java,adv java& frameworks ?

4 out of 5 or 8 out of 10 …..like

2.What is volatile?

The Java volatile keyword cannot be used with method or class and it can only be used with a variable. Java volatile keyword also guarantees visibility and ordering, after Java 5 write to any volatile variable happens before any read into the volatile variable. By the way use of volatile keyword also prevents compiler or JVM from the reordering of code or moving away them from synchronization barrier.

3.What is native?

The **native** keyword is used to declare a method which is implemented in platform-dependent code such as C or C++. When a method is marked as native, it cannot have a body and must ends with a semicolon instead. The [Java Native Interface (JNI)](http://docs.oracle.com/javase/7/docs/technotes/guides/jni/)specification governs rules and guidelines for implementing native methods, such as data type conversion between Java and the native application.

The following example shows a class with a method declared as native:

|  |  |
| --- | --- |
|  | publicclassNativeExample {        publicnativevoidfastCopyFile(String sourceFile, String destFile);    } |

4.Difference between comparable & comparator?

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by **Collections.sort(List)** method. | We can sort the list elements of Comparator type by **Collections.sort(List,Comparator)** method. |

5.What is linked hash set?

Java LinkedHashSet class is a Hash table and Linked list implementation of the set interface. It inherits HashSet class and implements Set interface.

The important points about Java LinkedHashSet class are:

* Contains unique elements only like HashSet.
* Provides all optional set operations, and permits null elements.
* Maintains insertion order.

6.What is weak hash map?

WeakHashMap is an implementation of the Map interface that stores only weak references to its keys. Storing only weak references allows a key-value pair to be garbage-collected when its key is no longer referenced outside of the WeakHashMap.

7.What is implicit objects in java?

There are **9 jsp implicit objects**. These objects are *created by the web container* that are available to all the jsp pages.

|  |  |
| --- | --- |
| **Object** | **Type** |
| Out | JspWriter |
| Request | HttpServletRequest |
| Response | HttpServletResponse |
| Config | ServletConfig |
| Application | ServletContext |
| Session | HttpSession |
| pageContext | PageContext |
| Page | Object |
| Exception | Throwable |

8.What is dialet?

To connect to any database with hibernate, we need to specify the SQL dialect class in hibernate.cfg.xml

**Ex:**To connect to oracle database we need to specify oracle dialect class in configuration xml as below.

<property name="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</property>

* **Dialect**class is java class, which contains code to map between java language data type to database data type.
* All Dialect classes extend the Dialect abstract class.
* Dialect is used to convert HQL statements to data base specific statements.

9.What is autowiring?

# **Autowiring in Spring**

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

Autowiring can't be used to inject primitive and string values. It works with reference only.

## Advantage of Autowiring

It requires the **less code** because we don't need to write the code to inject the dependency explicitly.

## Disadvantage of Autowiring

No control of programmer.

It can't be used for primitive and string values.

10.Which type of inheritence does hibernate supports,What is different types of inheritence?

There are three types of inheritance mapping in hibernate

1. Table per concrete class with unions

2. Table per class hierarchy

3. Table per subclass

different types of inheritance

1.override

2.overload

11.What is bean factory?

A BeanFactory is like a factory class that contains a collection of beans. The BeanFactory holds Bean Definitions of multiple beans within itself and then instantiates the bean whenever asked for by clients.

The BeanFactory is the actual container which instantiates, configures, and manages a number of beans. These beans typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the BeanFactory

BeanFactory also takes part in the life cycle of a bean, making calls to custom initialization and destruction methods.

12.What is DAO and DTO?Tell me main difference?

Data Access Object (DAO)

Data Transfer Object (DTO).

DAO is a class that usually has operations like save, update, delete. Whereas the DTO is just an object that holds data.

Infosys(Technical F2F 1st-round)

1.How much rating will give in core java,adv java& frameworks ?

It’s depends on your preparation.

2.What is different types of memories present in jvm?

· Heap

· Stack

· Program Counter Register

· Native Method Stack

3.Where Arrays will store?Is Arrays premitive(or)non-Primitive?

Arrays in Java store one of two things: either primitive values (int, char, ...) or references (a.k.a pointers).

When an object is creating by using "new", memory is allocated on the heap and a reference is returned. This is also true for arrays, since arrays are objects.

Arrays premitive(or)non-Primitive

No,because primitive data type is a data type which is direclty accessble by the operating system or which is predfined.like int,char,boolean.  
but Array derived from the primitve data type.so it is Non primitve data types.

4.What is String constant pool?

1. In our program if any String object is required to use repeatedly then it is not recommended to create multiple object with same content it reduces performance of the system and effects memory utilization.
2. We can create only one copy and we can reuse the same object for every requirement. This approach improves performance and memory utilization we can achieve this by using "scp".
3. In SCP several references pointing to same object the main disadvantage in this approach is by using one reference if we are performing any change the remaining references will be impacted. To overcome this problem sun people implemented immutability concept for String objects.
4. According to this once we creates a String object we can't perform any changes in the existing object if we are trying to perform any changes with those changes a new String object will be created hence immutability is the main disadvantage of scp.

5.String s1="abc";

String s2=new String("abc");

syso(s1.equals(s2));

syso(s1==s2); What is the output? y?

A)

syso(s1.equals(s2)) = true

syso(s1==s2) =false

s1 reference is stored in string constant pool and s2 stored in heap area. So when we are comparing content the output is True. And comparing addresses the output is False.

6.class A{

m1()throws Exception

{

} what are the exceptions get in child class?

A.Exception

7.What are the runtime exceptions?

A)Exception happends at runtime without showing any compile time error.

8.I have a list of objects how to iterate it?

* We can iterate list in 6  different ways in java.

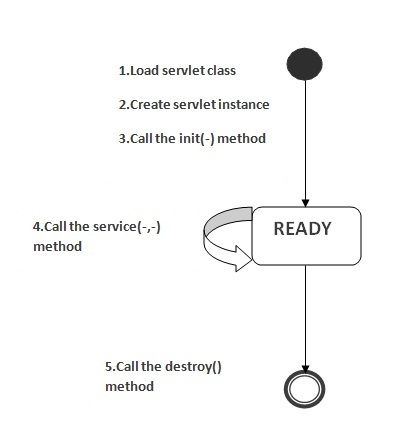
1. For Loop
2. Enhanced For Loop
3. While Loop
4. Iterator
5. Collections stream() util (Java8 feature)
6. List<String> instanceofjavaList = new ArrayList<String>();
8. //Add elements to Arraylist
10. // add 5 different values to arraylist
11. instanceofjavaList.add("Interview Questions");
12. instanceofjavaList.add("Interview Programs");
13. instanceofjavaList.add("Concept and example program");
14. instanceofjavaList.add("Concept and interview questions");
15. instanceofjavaList.add("Java Quiz");

18. instanceofjavaList.forEach((name) -> {
19. System.out.println(name);
20. });
21. ListIterator

9.What are the java 1.5 features?

* For-each loop (Java 5)
* Varargs (Java 5)
* Static Import (Java 5)
* Autoboxing and Unboxing (Java 5)
* Enum (Java 5)
* Covariant Return Type (Java 5)
* Annotation (Java 5)
* Generics (Java 5)

10.What is life cycle of servlet?



11.I have 10 request how many objects servlet container will create?

A)at the time of first request

12.I have 1000 requests how servlet will give responce to 1000 requests at a time?

By default web-server creates only one instance per servlet, if multiple request is going to a servlet then each request will processed in a separate thread, so container creates a thread per request for the single servlet instance, so your servlet should be thread-safe.

But web-server doesn’t create thread per request, there is a pool of threads to process the request.

For example , If there are 1000 request , and the maximum threads that can be generated by servlet is 100, then there will be performance degradation.

In order to avoid this problem , we can use **load balancer** by putting multiple servers behind a load balancer.

13.how to handle exceptions by using jsp pages?

The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer. In JSP, there are two ways to perform exception handling:

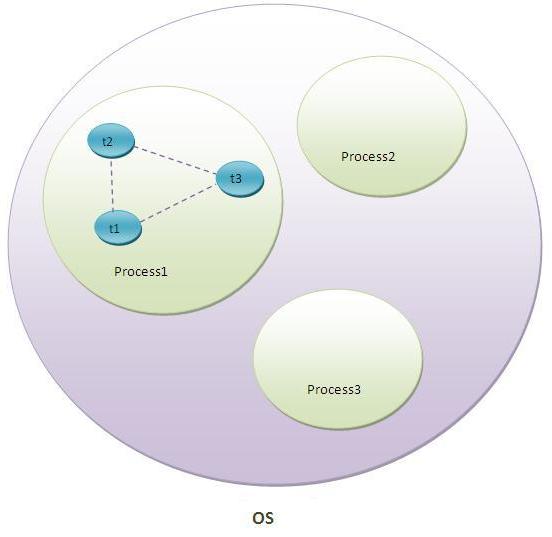
By **errorPage** and **isErrorPage** attributes of page directive

**<error-page>** element in web.xml file

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*veerababu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**16.What is multithreading?If two threads access same resource wt happen(synchronization)?**

**Multithreading in java** is a process of executing **multiple threads** simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and **multithreading**, both are used to achieve multitasking.



The synchronized statement (§14.19) computes a reference to an object; it then attempts to perform a lock action on that object's monitor and **does not proceed further until the lock action has successfully completed**. After the lock action has been performed, the body of the synchronized statement is executed. If execution of the body is ever completed, either normally or abruptly, an unlock action is automatically performed on that same monitor.

**17.What is procedures and functions?**

**Functions** can have only input parameters for it whereas **Procedures** can have input/output parameters . **Function** takes one input parameter it is mandatory but Stored **Procedure** may take o to n input parameters.. **Functions** can be called from**Procedure** whereas **Procedures** cannot be called from **Function**.

A function returns a value and a procedure just executes commands.

A procedure is a set of command which can be executed in order.

**18.What is deep cloning and shallow cloning?**

**Deep Cloning:**

Cloned Object and original object are 100% disjoint. Any changes made to cloned object will not be reflected in original object or vice versa. To create the deep copy of an object, you have to override clone method. Deep copy is preferred if an object has references to other objects as fields. Deep copy is slow and very expensive.

**Shallow Cloning:**

Cloned Object and original object are not 100% disjoint. Any changes made to cloned object will be reflected in original object or vice versa. Default version of clone method creates the shallow copy of an object. Shallow copy is preferred if an object has only primitive fields. Shallow copy is fast and also less expensive.

**19.What synchronization and seriolization?**

**Synchronization:**

Synchronization in java is the capability to control the access of multiple threads to any shared resource.

There are two types of synchronization

1. Process Synchronization
2. Thread Synchronization

**Serialization:**

**Serialization in java** is a mechanism of writing the state of an object into a byte stream.

**C:\Users\Admin\Desktop\java-serialization (1).png**

**20.How to implement seriolization in urproject?What is the use?**

* [**Implement Serialization using Java Serializable Interface**.](http://www.techbeamers.com/java-serialization-tutorial/#serializable-interface)
* [**Implement Serialization with Inheritance**.](http://www.techbeamers.com/java-serialization-tutorial/#serialization-Inheritance)

**21.What is diff b/w oracle and mysql?**

1. Oracle runs on many platforms, SQL on Windows only

2. Oracle includes IFS (Internet File System), Java integration, SQL is more of a pure database   
3. Oracle requires client install and setup (Not difficult, but very UNIX-like for Windows users)

**22.If u want change oracle db to postgracedb?How to change?**

**Not found**

**23.What is diff b/w arraylist and hashmap?**

**ArrayList** vs **HashMap in Java**. 1) Implementation: **ArrayList** implements List Interface while **HashMap** is an implementation of Map interface. List and Map are two entirely different collection interfaces.

2) Memory consumption: **ArrayList** stores the element's value alone and internally maintains the indexes for each element

**Order**: ArrayList maintains the insertion order while HashMap doesn’t.  Which means ArrayList returns the list items in the same order in which they got inserted into the list. On the other side HashMap doesn’t maintain any order, the returned key-values pairs are not sorted in any kind of order.

4) **Duplicates**: ArrayList allows duplicate elements but HashMap doesn’t allow duplicate keys (It does allow duplicate values).

5) **Nulls**: ArrayList can have any number of null elements. HashMap allows one null key and any number of null values.

**24.What is try,catch and finally blocks?**

 try block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.

Java try block must be followed by either catch or finally block.

**try**{

//code that may throw exception

}**catch**(Exception\_class\_Name ref){}

**try**{

//code that may throw exception

}**finally**{}

## catch block

Java catch block is used to handle the Exception. It must be used after the try block only.

You can use multiple catch block with a single try.

# finally block

Java finally block is always executed whether exception is handled or not.

Java finally block follows try or catch block

## C:\Users\Admin\Desktop\finally.JPG

**25.Which design patterns do u use in urproject?Which one do u think most useful?**

**Answer not found**

**26.What is singleton?What is the use?Where do u use?**

This property is useful to create **singleton** class in **java**. **Singleton** pattern helps us to keep only one instance of a class at any time. The purpose of **singleton** is to control object creation by keeping private constructor.

**Where do u use**

**27.What is diff b/w update and merge?**

|  |  |  |
| --- | --- | --- |
| **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. |

**28.What is diff b/w HQL and Criteria?**

* HQL is to perform both select and non-select operations on the data,  but Criteria is only for selecting the data, we cannot perform non-select operations using criteria
* HQL is suitable for executing Static Queries, where as Criteria is suitable for executing Dynamic Queries
* HQL doesn’t support pagination concept, but we can achieve pagination with Criteria
* Criteria used to take more time to execute then HQL

***29.Write program on calender?I want o/p like 10 days before date?What is the logic?***

*Answer not found*

**30.What are the technologies do u use in front end ?**

**HTML,CSS,JAVASCRIPT,BOOTSTRAP**

**31.What is lazy init attribute?**

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.

**32.What is restful api?How to implement ?What is the use?**

**Answer not found**

**33.** **If u get conflicts in urproject?How do u solve?**

**Answer not found**

**34.** **What is named parameters?**

**Answer not found**

**35.If u add the same element in treeset ?What will happen?**

**Answer not found**

**36.What is secondlevelcache?**

**Hibernate second level cache** uses *a common cache for all the session object of a session factory*. It is useful if you have multiple session objects from a session factory.

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

Different vendors have provided the implementation of Second Level Cache.

1. EH Cache
2. OS Cache
3. Swarm Cache
4. JBoss Cache

Each implementation provides different cache usage functionality. There are four ways to use second level cache.

1. **read-only:** caching will work for read only operation.
2. **nonstrict-read-write:** caching will work for read and write but one at a time.
3. **read-write:** caching will work for read and write, can be used simultaneously.
4. **transactional:** caching will work for transaction.

**37.How many types of injections are there in spring?**

Spring supports 2 types of dependency injection, they are:

1) Constructor-based dependency injection: It is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.

2) Setter-based dependency injection: It is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

**1.Which version of java do u use?(Ans:java 1.6)**

**2.Can u tell me the feature of java 1.5?**

* For-each loop (Java 5)
* Varargs (Java 5)
* Static Import (Java 5)
* Autoboxing and Unboxing (Java 5)
* Enum (Java 5)
* Covariant Return Type (Java 5)
* Annotation (Java 5)
* Generics (Java 5)

**3.What is Generics?**

A type parameter, also known as a type variable, is an identifier that specifies a**generic** type name. The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the **generic** method, which are known as actual type arguments.

**4.Which version of spring do u use?**

**Answer not found**

**5.What is the scopes in spring?**

In most cases, you may only deal with the Spring's core scope – singleton and prototype, and the default scope is singleton. Singleton - (Default) Scopes a single**bean definition** to a single **object instance** per Spring IoC container.

**6.What is the singleton?**

This property is useful to create **singleton** class in **java**. **Singleton** pattern helps us to keep only one instance of a class at any time. The purpose of **singleton** is to control object creation by keeping private constructor.

**7.Write a program on comparable?**

**Answer not found**

**8.How to sort the objects?**

1. Sort an Array. To sort an Array, use the Arrays.sort(). ...
2. Sort an ArrayList. To sort an ArrayList, use the Collections.sort(). ...
3. Sort an Object with Comparable. How about a Java Object? ...
4. Sort an Object with Comparator. How about sorting with Fruit's “fruitName” or “Quantity”?

**9.Write a program on comparable?I want to comapare only name ?Write a internal logic?**

**Answer not found**

**10.I want to implement both id and name comaparsion is it possible in comparable?**

**Answer not found**

**11.How do u done validation?**

**12.How do u done authentication and autherization in ur project?**

**Answer not found**

**13.how do u come to knw is he is particular autherized person?**

**Answer not found**

**14.How many types of containers is there in spring?**

**1. BeanFactory Container:**

This is the simplest container providing basic support for Dependency Injection and defined by the *org.springframework.beans.factory.BeanFactory* interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purposes of backward compatibility with the large number of third-party frameworks that integrate with Spring.

**2. ApplicationContext Container:**

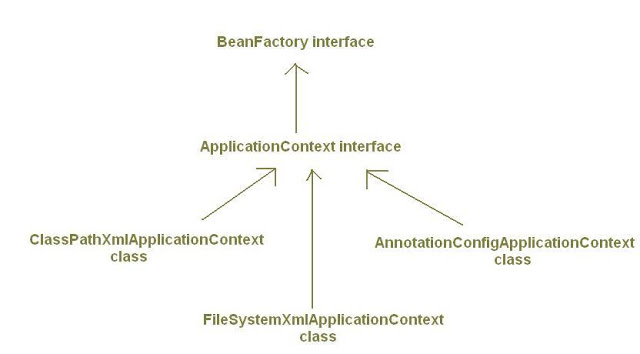
This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the org.springframework.context.ApplicationContext interface.

The ApplicationContext container includes all functionality of the BeanFactory container, so it is generally recommended over the BeanFactory. BeanFactory can still be used for light weight applications like mobile devices or applet based applications where data volume and speed is significant.

**15.What is diff b/w bean factory and application context?**

BeanFactory is the **basic container** whereas ApplicationContext is the **advanced container**. ApplicationContext extends the BeanFactory interface. ApplicationContext provides more facilities than BeanFactory such as integration with spring AOP, message resource handling for i18n etc.

**16.What is the features present in application context?**

****

The ClassPathXmlApplicationContext class is the implementation class of ApplicationContext interface. We need to instantiate the ClassPathXmlApplicationContext class to use the ApplicationContext as given below:

1. ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

**17.Inhashmap if u add the duplicates it will override existing value?my requirement i want**

**allowduplicates?How is it possible in hashmap?**

**18.** **Two beans configured in spring.xml first bean some data is present in second bean also some data is there**

**but we are modifying that data in second bean,in both beans x()method is there if u call that x() what data we will get**

**ie original data (or) modifying data?**

**Answer not found.**

**19.How to implement triggers and joins?**

**Triggers:**

Triggers are similar to stored procedures. A trigger stored in the database can include SQL and PL/SQL or Java statements to run as a unit and can invoke stored procedures. However, procedures and triggers differ in the way that they are invoked. A procedure is explicitly run by a user, application, or trigger. Triggers are implicitly fired by Oracle when a triggering event occurs, no matter which user is connected or which application is being used.

Joins:

**Oracle JOINS** are used to retrieve data from multiple tables. An **Oracle JOIN** is performed whenever two or more tables are joined in a SQL statement. There are 4 different types of **Oracle joins**: **Oracle** INNER **JOIN** (or sometimes called simple **join**) **Oracle** LEFT OUTER **JOIN**

**20.What is junit&junitparams?**

**JUnit** is a unit testing framework for the **Java** programming language. **JUnit** has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

**21.How to u write test case ?Tell me the scenario?**

**Answer not found**

**22.What is mockito?**

**Mockito** is a mocking framework, **JAVA**-based library that is used for effective unit testing of **JAVA**applications. **Mockito** is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

--------------------------------------

**1.Explain about exceptions?Different types?**

Exception Handling is a mechanism to handle runtime errors.It is mainly used to handle checked exceptions.

### 1)Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException,SQLException etc. Checked exceptions are checked at compile-time.

### 2)Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException,NullPointerException etc. Unchecked exceptions are not checked at compile-time.

**2.When will u get null pointer exception?**

In **Java**, a special **null** value **can** be assigned to an object reference.**NullPointerException** is thrown when an application attempts to use an object reference that has the **null** value. These include: Calling an instance method on the object referred by a **null** reference.

**3.Explain about collectonframeworks?scenarios?**

**Answer not found**

**4.Which one is better for searching operation?**

**A.Hibernate criteria**

**5.If u add duplicates in employee object what will happen?How will u remove duplicates?**

**6.Write program on employee based upon id and based upon name?How will u iterate?**

7.Write program on "MALAYALAM" ?Is it palemdram?

8.Wtite a program String s="abc10ef1";

o/p=11.How will u do it?

9.Write a singleternprogram?What is it?

**10.What is string constant pool?**

As the name suggests, **String Pool in java** is a **pool** of **Strings** stored in **Java** Heap 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

11.String s="abc"+"def"+efg";How many objects will create in scp?

12.String s="abc";

String s1=new String ("abc");

syso(s==s1); What is the output? y?

**13.How will u configure servlert in web.xml?**

<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"

version="2.4">

<display-name>HelloWorld Application</display-name>

<description>

This is a simple web application with a source code organization

based on the recommendations of the Application Developer's Guide.

</description>

<servlet>

<servlet-name>HelloServlet</servlet-name>

<servlet-class>examples.Hello</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>HelloServlet</servlet-name>

<url-pattern>/hello</url-pattern>

</servlet-mapping>

</web-app>

**14.Write a query on update?**

**15.What is joins?**

**Oracle JOINS** are used to retrieve data from multiple tables. An **Oracle JOIN** is performed whenever two or more tables are joined in a SQL statement. There are 4 different types of **Oracle joins**: **Oracle** INNER **JOIN** (or sometimes called simple **join**) **Oracle** LEFT OUTER **JOIN**

**16.Write a test case on addition?**

**Answer not found.**

**Manager round(F2F)**

-----------------------------

**1.Tell me which technologies do u use?**

**2.can u explain your project flow?How will u get data one controller to another controller?**

**3.how will u done validations?**

**4.How to write test cases?**

**5.Which design pattern do u use in your project?**

**6.Is your project maintainence or scratch?**

**7.Why do u want change another comapany?**

**8.Are u interested to do work on development project?**

**9.Do u know jquery?**

**10.which layers do work on?what do u implement?**

**11.Tell me about your current project?**

**12.Tell me about your previous project?What do u implement?**

**13.Who is your client?Can u tell me the client name?**

**14.Which database do u use?Do u knw SQL?**

**15.He explain about company branches and company details?**

**HR-round**

**--------------------**

**1.Tell me about yourself?Where do u live?**

**2.Tell me about your family background?**

**3.Why do u want change another company?**

**4.Which technologies do u work on?**

**5.Which domain do u work on?**

**6.He explain about company details?What they will offer for employees?**

**7.Do u have any offers on hand?**

**8.Why do want relocate?**

**9.What is your official notice period?When will u join ?**

**10.Do u want ask any questions?**

**=======Integra Micro Software Services =========**

**1.tell me about yourself**

**2.expalin about your project**

**3.what is spring orm why we use it**

**4.what is hibernate one which way you configure by annotaion or configuration**

**5.what is trancation who will take of it**

**6.if there are three table i am inserting data into three table .exception occures while insterting data into second tabe who will take of it**

**7.what is mockito**

**8.where will configure exceptions**

**9.if exception occured i want to through one message how will you display it**

**10.what is spring MVC and explain flow**

**11.how will you validated the data**

**13.what are the annotations used for validations**

**14.what is multithreading**

**15.there is one file if there are five members are accessing i want to dispaly the values parllely how by using multitheading**

**16.write a program for add to list when two integers are added give sum as 60 and when therdiffrence give 14**

**17.write a program for the dsplaying the string in the s-1.t-2,r-3,i-4,n-5,g-6**

**18.write a program for method ovrriding**

............... Monarch info tech services pvt ltd (interview questions)................

1.EAR,WAR,JAR Difference

* [Jar files](http://searchdatamanagement.techtarget.com/definition/JAR-file)(files with a .jar extension) are intended to hold generic [libraries](http://searchsqlserver.techtarget.com/definition/library) of Java classes, resources, auxiliary files, etc.
* War files (files with a .war extension) are intended to contain complete Web applications. In this context, a Web application is defined as a single group of files, classes, resources, .jar files that can be packaged and accessed as one servlet context.
* Ear files (files with a .ear extension) are intended to contain complete enterprise applications. In this context, an enterprise application is defined as a collection of .jar files, resources, classes, and multiple Web applications.

2.it is possible rename spring-servlet.xml ? How to rename?

3.two systems is there one system have Server and another system have database how to communicate?

A.Through network

4.if create one jsp page and run it successfully..after display output and it is possible to modify the jsp page?

A. we can modify

5.if i write ERROR code in INFO level then it is display or not?

A. yes it displays.

6.without using XML,Properties how to configure Log4j file?

7.log4j level order?

|  |  |
| --- | --- |
| **Level** | **Description** |
| ALL | All levels including custom levels. |
| DEBUG | Designates fine-grained informational events that are most useful to debug an application. |
| ERROR | Designates error events that might still allow the application to continue running. |
| FATAL | Designates very sever error events that will presumably lead the application to abort. |
| INFO | Designates informational messages that highlight the progress of the application at coarse-grained level. |
| OFF | The highest possible rank and is intended to turn off logging. |
| TRACE | Designates finer-grained informational events than the DEBUG. |
| WARN | Designates potentially harmful situations. |

8.Difference between Table & View?

A **view** is a virtual **table**. A **view** consists of rows and columns just like a **table**. The**difference between** a **view** and a **table** is that **views** are definitions built on top of other **tables** (or **views**), and do not hold data themselves. If data is changing in the underlying **table**, the same change is reflected in the **view**.

9.what are the hibernate annotations?

@Entity Annotation:

The EJB 3 standard annotations are contained in the **javax.persistence** package, so we import this package as the first step. Second we used the **@Entity** annotation to the Employee class which marks this class as an entity bean, so it must have a no-argument constructor that is visible with at least protected scope.

@Table Annotation:

The @Table annotation allows you to specify the details of the table that will be used to persist the entity in the database.

The @Table annotation provides four attributes, allowing you to override the name of the table, its catalogue, and its schema, and enforce unique constraints on columns in the table. For now we are using just table name which is EMPLOYEE.

@Id and @GeneratedValue Annotations:

Each entity bean will have a primary key, which you annotate on the class with the **@Id** annotation. The primary key can be a single field or a combination of multiple fields depending on your table structure.

By default, the @Id annotation will automatically determine the most appropriate primary key generation strategy to be used but you can override this by applying the **@GeneratedValue** annotation which takes two parameters **strategy** and **generator** which I'm not going to discuss here, so let us use only default the default key generation strategy. Letting Hibernate determine which generator type to use makes your code portable between different databases.

@Column Annotation:

The @Column annotation is used to specify the details of the column to which a field or property will be mapped. You can use column annotation with the following most commonly used attributes:

* **name** attribute permits the name of the column to be explicitly specified.
* **length** attribute permits the size of the column used to map a value particularly for a String value.
* **nullable** attribute permits the column to be marked NOT NULL when the schema is generated.
* **unique** attribute permits the column to be marked as containing only unique values.

10.using jstl tags how to get database connection?

The **<sql:setDataSource>** tag sets the data source configuration variable or saves the data-source information in a scoped variable that can be used as input to the other JSTL database actions.

11.how to call procedures in hibernate?

1. Native SQL – createSQLQuery

You can use **createSQLQuery()** to call a store procedure directly.

2. NamedNativeQuery in annotation

Declare your store procedure inside the **@NamedNativeQueries** annotation.

Call it with **getNamedQuery()**.

3. sql-query in XML mapping file

Declare your store procedure inside the "**sql-query**" tag.

Call it with **getNamedQuery()**.

12.can constructor throw exception java?

**Constructors are nothing** more than special methods, and **can throw exceptions**like any other method. A **constructor CAN throw** any **exception**. But if any subclass **constructor** calls a super class **constructor** which **throws** an **exception**, then the subclass **constructor** must either catch the **exception** or **throw**it.

13.difference between soap and restful services?

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 6) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |

14.what is catalog database in oracle?

Use the CREATE CATALOG command to create a **recovery catalog**. The **recovery catalog** can be a **base recovery catalog**, which is a database schema that contains**RMAN** metadata for a set of target databases. A virtual private catalog is a set of synonyms and views that enable user access to a subset of a **base recovery catalog**.

15.What is front controller design pattern in Spring MVC?

In Spring Web MVC, **DispatcherServlet** class works as the front controller. It is responsible to manage the flow of the spring mvc application.

The **@Controller** annotation is used to mark the class as the controller in Spring 3.

The **@RequestMapping** annotation is used to map the request url. It is applied on the method.

16.What is DAO design pattern in Spring MVC?

A.answer not found.

17.What happens if a finally block throws an exception?

**A.Finally** clause is executed even when **exception** is thrown from anywhere in try/catch **block**. Because it's the last to be executed in the main and it **throws an exception**, that's the **exception** that the callers see. ... A method can't **throw** two**exceptions** at the same time

18.how to download eclipse plugins manually?

1. Download your plugin
2. Open Eclipse
3. From the menu choose: Help / Install New Software...
4. Click the Add button
5. In the Add Repository dialog that appears, click the Archive button next to the Location field
6. Select your plugin file, click OK

19.spring bean life cycle?

20.How to write junit tests for interfaces?

public interface MyInterface {

/\*\* Return the given value. \*/

public boolean myMethod(boolean retVal);

}

public class MyClass1 implements MyInterface {

public boolean myMethod(boolean retVal) {

return retVal;

}

}

public class MyClass2 implements MyInterface {

public boolean myMethod(boolean retVal) {

return retVal;

}

}

Possibility 1:

public abstract class MyInterfaceTest {

public abstract MyInterface createInstance();

@Test

public final void testMyMethod\_True() {

MyInterface instance = createInstance();

assertTrue(instance.myMethod(true));

}

@Test

public final void testMyMethod\_False() {

MyInterface instance = createInstance();

assertFalse(instance.myMethod(false));

}

}

public class MyClass1Test extends MyInterfaceTest {

public MyInterface createInstance() {

return new MyClass1();

}

}

public class MyClass2Test extends MyInterfaceTest {

public MyInterface createInstance() {

return new MyClass2();

}

}

Possibility 2:

public abstract class MyInterfaceTest

public void testMyMethod\_True(MyInterface instance) {

assertTrue(instance.myMethod(true));

}

public void testMyMethod\_False(MyInterface instance) {

assertFalse(instance.myMethod(false));

}

}

public class MyClass1Test extends MyInterfaceTest {

@Test

public void testMyMethod\_True() {

MyClass1 instance = new MyClass1();

super.testMyMethod\_True(instance);

}

@Test

public void testMyMethod\_False() {

MyClass1 instance = new MyClass1();

super.testMyMethod\_False(instance);

}

}

public class MyClass2Test extends MyInterfaceTest {

@Test

public void testMyMethod\_True() {

MyClass1 instance = new MyClass2();

super.testMyMethod\_True(instance);

}

@Test

public void testMyMethod\_False() {

MyClass1 instance = new MyClass2();

super.testMyMethod\_False(instance);

}

}

21.what are the Differen types of spring bean scopes

The Spring Framework supports the following five scopes, three of which are available only if you use a web-aware ApplicationContext.

|  |  |
| --- | --- |
| **Sr.No.** | **Scope & Description** |
| 1 | **singleton**  This scopes the bean definition to a single instance per Spring IoC container (default). |
| 2 | **prototype**  This scopes a single bean definition to have any number of object instances. |
| 3 | **request**  This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext. |
| 4 | **session**  This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |
| 5 | **global-session**  This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |

22.What are the features of hibernate?

* Since hibernate generates lots of SQL statements at runtime so it is **slower than pure JDBC**
* Hibernate is not much flexible in case of **composite mapping**. This is not disadvantage since understanding of composite mapping is complex
* Hibernate does not support **some type of queries** which are supported by JDBC
* **Boilerplate code issue**, actually we need to write same code in several files in the same application, but **spring** eliminated this

23.What is hibernate mapping file?

In this file hibernate application developer specify the mapping from **entity** class name to **table** name and entity **properties** names to table **column** names. i.e. mapping object oriented data to relational data is done in this file

Standard name for this file is **<domain-object-name.hbm.xml>**

In general, for each domain object we create one mapping file

**Number of Entity classes = that many number of mapping xmls**

Mapping can be done using **annotations** also. If we use annotations for mapping then we no need to write mapping file.

From hibernate 3.x version on wards it provides support for annotation, So mapping can be done in two ways

**XML**

**Annotations**

**Syntax Of Mapping xml:**

**<hibernate-mapping>**

**<class name="Entity class name" table="table name in database">**

**<id name="id variable name" column="primary column name in database" />**

**<property name="variable1 name" column="column name in database" />**

**<property name="variable2 name" column="column name in database" />**

**</class>**

**</hibernate-mapping>**

**Syntax Of Mapping Annotations:**

**@Entity**

**@Table(name = "table name in database")**

**public class EntityName {**

**@Id**

**@Column(name = "primary column name in database")**

**private int idVariableName;**

**@Column(name = "column name in database ")**

**private String variableName1;**

**@Column(name = "column name in database ")**

**private String variableName2;**

**// setters & getters**

**}**

24.What do you know about dialect in Hibernate?

Hibernate.dialect property tells Hibernate to generate the appropriate **SQL**statements for the chosen database.

25.Explain about annotations which are used in our application to map entity to table?

26.What are the states of object in hibernate?

27.What is difference between sorted and ordered collection in hibernate?

28.Hibernate Caching Mechanism ?

29.What is IOC? What is role of it in spring?

30.What is DI? What are different types of DI? which is better?

**dependency injection** is a technique whereby one object supplies the**dependencies** of another object. A **dependency** is an object that can be used (a service). An **injection** is the passing of a **dependency** to a dependent object (a client) that would use it.

31.what is JNDI?

The Java Naming and Directory Interface (**JNDI**) is a Java API for a directory service that allows Java software clients to discover and look up data and objects via a name. Like all Java APIs that interface with host systems, **JNDI** is independent of the underlying implementation.

32.spring MVC architecture?

28.what are the JSP tags?

There are three types of directive tag

|  |  |
| --- | --- |
| **S.No.** | **Directive & Description** |
| 1 | **<%@ page ... %>**  Defines page-dependent attributes, such as scripting language, error page, and buffering requirements. |
| 2 | **<%@ include ... %>**  Includes a file during the translation phase. |
| 3 | **<%@ taglib ... %>**  Declares a tag library, containing custom actions, used in the page |

=======MITS===========

1.Tell me about your self

2.expalin your project

3.what is svn why it is used

SVN is a centralized version control system, mainly used for managing source code, but also used for other text-based data that can be changed over time by different people.

People who work with source code on a daily basis rely on such systems and are familiar with their workings, but they can be bewildering for people who encounter them for the first time.

4.what is log4j why it is used

log4j is a reliable, fast and flexible **logging framework** (APIs) written in Java, which is distributed under the Apache Software License. log4j is a popular logging package written in Java. log4j has been ported to the C, C++, C#, Perl, Python, Ruby, and Eiffel languages.

* Logging is the act of recording the events occurred in execution of application.

5.what is junit

**JUnit** is a unit testing framework for the Java programming language. **JUnit** has been important in the development of test-driven development.

6.what is mockito and how can we create project

**Mockito** is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. **Mockito** is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

Basically a class which provides an API for the application to access and modify the data in the data repository. Unit test the DAL(Data Access Layer) without actually the need to connect to the data repository. The data repository can be a local database or remote database or a file system or any place where we can store and retrieve the data. The use of a DAL class helps us in keeping the data mappers separate from the application code.

We all write unit tests but the challenge we face at times is that the unit under test might be dependent on other components. And configuring other components for unit testing is definitely an overkill. Instead we can make use of Mocks in place of the other components and continue with the unit testing.

7.what is singleton

This property is useful to create **singleton** class in **java**. **Singleton** pattern helps us to keep only one instance of a class at any time. The purpose of **singleton** is to control object creation by keeping private constructor.

8.what is DAO

**data access object** (**DAO**) is an [object](https://en.wikipedia.org/wiki/Object_(computer_science)) that provides an abstract [interface](https://en.wikipedia.org/wiki/Interface_(computer_science)) to some type of [database](https://en.wikipedia.org/wiki/Database) or other persistence mechanism. By mapping application calls to the persistence layer, the DAO provides some specific data operations without exposing details of the database. ... **DAO** is an abbreviation for**Data Access Object**, so it should encapsulate the logic for retrieving, saving and updating data in your data storage (a database, a file-system, whatever).

9.What is DTO

**Data transfer object** (**DTO**), formerly known as value objects or VO, is a design pattern used to transfer data between software application subsystems. **DTOs** are often used in conjunction with data access objects to retrieve data from a database.

**DTO** is an abbreviation for **Data Transfer Object**, so it is used to transfer the data between classes and modules of your application.

10.what are factory method

**Factory Method Pattern**. A **Factory Pattern** or**Factory Method Pattern** says that just define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate. ... The **Factory Method Pattern** is also known as Virtual Constructor.

(Factory methods are static methods that return an instance of the native class. Examples in the JDK:

* LogManager.[getLogManager](http://docs.oracle.com/javase/7/docs/api/java/util/logging/LogManager.html#getLogManager())
* Pattern.[compile](http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#compile(java.lang.String))
* Collections.[unmodifiableCollection](http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#unmodifiableCollection(java.util.Collection)), Collections.[synchronizeCollection](http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#synchronizedCollection(java.util.Collection)) , and so on
* Calendar.[getInstance](http://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html#getInstance())

Factory methods:

* have names, unlike constructors, which can clarify code.
* do not need to create a new object upon each invocation - objects can be cached and reused, if necessary.
* can return a subtype of their return type - in particular,*can return an object whose implementation class is unknown to the caller.*This is a very valuable and widely used feature in many frameworks which use interfaces as the return type of static factory methods.

Common names for factory methods include getInstance and valueOf. These names are not mandatory - choose whatever makes sense for each case.)

11.what is hibrnate and why we are going for hibernate

Hibernate framework simplifies the development of java application to interact with the database. Hibernate is an open source, lightweight, [ORM (Object Relational Mapping)](http://en.wikipedia.org/wiki/Object-relational_mapping) tool.

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.

* Hibernate **persists java objects** into database (Instead of primitives)
* It provides Database services in **Database vendor independent** Manner, so that java applications become portable across the multiple databases
* Hibernate generates **efficient queries** for java application to communicate with Database
* It provides **fine-grained exception handling** mechanism. In hibernate we only have**Un-checked exceptions,** so no need to write try, catch, or no need to write throws (In hibernate we have the translator which converts checked to Un-checked)
* It supports **synchronization** between in-memory java objects and relational records
* Hibernate provides implicit **connection pooling** mechanism
* Hibernate supports **Inheritance**, **Associations**, **Collections**
* Hibernate supports a special query language(**HQL**) which is **Database** **vendor** **independent**
* Hibernate has **capability to generate primary keys** automatically while we are storing the records into database
* Hibernate **addresses** the **mismatches** between java and database
* Hibernate provides **automatic change detection**
* Hibernate often reduces the **amount of code** needed to be written, so it Improves the **productivity**
* Database objects (tables, views, procedures, cursors, functions …etc) name changes will not affect hibernate code
* Supports over 30 **dialects**
* Hibernate provides **caching mechanism** for efficient data retrieval
* **Lazy loading** concept is also included in hibernate so you can easily load objects on start up time
* Getting **pagination** in hibernate is quite simple.
* Hibernate Supports automatic **versioning of rows**
* Hibernate provides **transactional capabilities** that can work with both stand-alone or java Transaction API (JTA) implementations …etc
* Hibernate supports **annotations**, apart from **XML**

12.what is HQL and write **code for the HQlrewritng purpose**

**Hibernate Query Language** (**HQL**) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, **HQL** works with persistent objects and their properties. **HQL** queries are translated by Hibernate into conventional SQL queries which in turns perform action on database.

13.what is transaction how we are using this

org.**hibernate**. Interface **Transaction**. ... Allows the application to define units of work, while maintaining abstraction from the underlying **transaction** implementation (eg. JTA, JDBC). A **transaction** is associated with a Session and is usually instantiated by a call to Session.beginTransaction().

14.what is purpose of transcation who will takecare of it

In hibernate framework, we have **Transaction** interface that defines the unit of work. It maintains abstraction from the transaction implementation (JTA,JDBC).

A transaction is associated with Session and instantiated by calling **session.beginTransaction()**.

The methods of Transaction interface are as follows:

1. **void begin()** starts a new transaction.
2. **void commit()** ends the unit of work unless we are in FlushMode.NEVER.
3. **void rollback()** forces this transaction to rollback.
4. **void setTimeout(int seconds)** it sets a transaction timeout for any transaction started by a subsequent call to begin on this instance.
5. **boolean isAlive()** checks if the transaction is still alive.
6. **void registerSynchronization(Synchronization s)** registers a user synchronization callback for this transaction.
7. **boolean wasCommited()** checks if the transaction is commited successfully.
8. **boolean wasRolledBack()** checks if the transaction is rolledback successfully.

15.where we will configure all the peroperties of hibernate.what will be configured in it

Hibernate can be configured using **hibernate.cfg.xml**file.

**hibernate.cfg.xml**

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://192.168.1.102:3306/javapapers</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<!-- SQL dialect -->

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- Specify session context -->

<property name="hibernate.current\_session\_context\_class">org.hibernate.context.internal.ThreadLocalSessionContext</property>

</session-factory>

</hibernate-configuration>

16.what is mappingfile. what will be configured init and write configuration of mapping file

The **mapping** document is an XML document having <**hibernate**-**mapping**> as the root element which contains all the <class> elements. The <class> elements are used to define specific **mappings** from a Java classes to the database tables.

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

<meta attribute="class-description">

This class contains the employee detail.

</meta>

<id name="id" type="int" column="id">

<generator class="native"/>

</id>

<property name="firstName" column="first\_name" type="string"/>

<property name="lastName" column="last\_name" type="string"/>

<property name="salary" column="salary" type="int"/>

</class>

</hibernate-mapping>

17.what is the difference between list and vector

|  |  |
| --- | --- |
| **List** | **Vector** |
| List is not synchronized | Vector is synchronized |
| It doesn’t have default size | It has a default size of 10 |
| It is not thread safe | It is thread safe |
| Lists are faster | Vectors take more CPU |
| It decreased to half i.e. 50% | It grows by its size twice |
| It do not permits accessing random addresses | It enables random address and making changes there |

18.what is hashtable

**hash table** (hash map) is a data structure which implements an associative array abstract data type, a structure that can map keys to values. A **hash table** uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found.

19.what is hashmap ,hashset

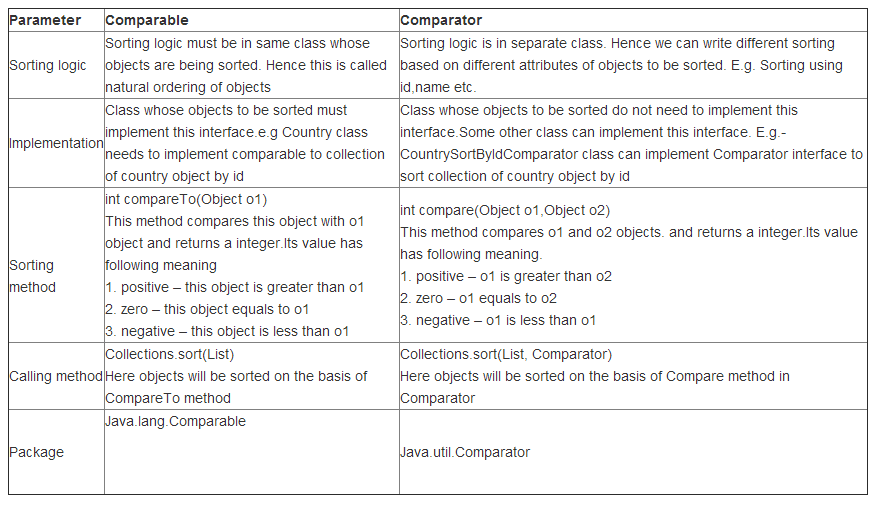
**HashMap** maintains key and value pairs and often denoted as **HashMap**<Key, Value> or **HashMap**<K, V>. **HashMap** implements Map interface. **HashMap** is similar to Hashtable with two exceptions – **HashMap**methods are unsynchornized and it allows null key and null values unlike Hashtable.

**HashSet** extends AbstractSet and implements the Set interface. It creates a collection that uses a hash table for storage. A hash table stores information by using a mechanism called hashing. In hashing, the informational content of a key is used to determine a unique value, called its hash code.

20.what is difference between list and set

A **Set** cannot contain duplicate elements while a **List** can. A **List** (in Java) also implies order. Conceptually we usually refer to an unordered grouping that allows duplicates as a Bag and doesn't allow duplicates is a **Set**. **List** is used to collection of elements with duplicates.

21.what is difference between comparator and comparable



22.what is servlet and life cycle

A Java **servlet** is a Java program that extends the capabilities of a server. Although **servlets** can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web **servlets** are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

# Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.

23.what is difference between forward and include

The main **difference** is that when you use **forward** the control is transferred to the next servlet/jsp you are calling, while **include** retains the control with the current servlet, it just includes the processing done by the calling servlet/jsp(like doing any out.println or other processing)

24.what is loadonstartup

If two or more servlets have the same <**load-on-startup**> positive integer value then they will be loaded in an order on which they are declared inside web.xml file. When the servlet is loaded by container, its init() method is called.

25.what is generaic servlet what are the unimplememted methods init

**GenericServlet** class implements **Servlet**, **ServletConfig** and **Serializable** interfaces. It provides the implementation of all the methods of these interfaces except the service method.

GenericServlet class can handle any type of request so it is protocol-independent.

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

**public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.

26.what is jsp what is the use of it

JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

A JavaServer Pages component is a type of Java **servlet** that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

**Uses:**

JavaServer Pages often serve the same purpose as programs implemented using the **Common Gateway Interface (CGI)**. But JSP offers several advantages in comparison with the CGI.

* Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
* JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
* JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including **JDBC, JNDI, EJB, JAXP,** etc.
* JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

27.what is scriplet,delcerative

In JavaServer Pages (JSP) technology, a **scriptlet** is a piece of Java-code embedded in the HTML-like JSP code. The **scriptlet** is everything inside the <% %> tags. Between these the user can add any valid **Scriptlet** i.e. any valid Java Code. In AppleScript, a **scriptlet** is a small script.

The **JSP declaration tag** is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

#### Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

**<**%!  field or method declaration %**>**

|  |  |
| --- | --- |
| **Jsp Scriptlet Tag** | **Jsp Declaration Tag** |
| The jsp scriptlet tag can only declare variables not methods. | The jsp declaration tag can declare variables as well as methods. |
| The declaration of scriptlet tag is placed inside the \_jspService() method. | The declaration of jsp declaration tag is placed outside the \_jspService() method. |

28.what are jstl tags

## JSTL Tags

There JSTL mainly provides 5 types of tags:

|  |  |
| --- | --- |
| **Tag Name** | **Description** |
| [Core tags](https://www.javatpoint.com/jstl-core-tags) | The JSTL core tag provide variable support, URL management, flow control etc. The url for the core tag is **http://java.sun.com/jsp/jstl/core** . The prefix of core tag is **c**. |
| [Function tags](https://www.javatpoint.com/jstl-function-tags) | The functions tags provide support for string manipulation and string length. The url for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**. |
| [Formatting tags](https://www.javatpoint.com/jstl-formatting-tags) | The Formatting tags provide support for message formatting, number and date formatting etc. The url for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**. |
| [XML tags](https://www.javatpoint.com/jstl-xml-tags) | The xml sql tags provide flow control, transformation etc. The url for the xml tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**. |
| [SQL tags](https://www.javatpoint.com/jstl-sql-tags) | The JSTL sql tags provide SQL support. The url for the sql tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**. |

29.what are implect method

30.what are the jstl tags to communicate with the database

The **<sql:setDataSource>** tag sets the data source configuration variable or saves the data-source information in a scoped variable that can be used as input to the other JSTL database actions.

## Attribute

The **<sql:setDataSource>** tag has the following attributes −

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Description** | **Required** | **Default** |
| driver | Name of the JDBC driver class to be registered | No | None |
| url | JDBC URL for the database connection | No | None |
| user | Database username | No | None |
| password | Database password | No | None |
| password | Database password | No | None |
| dataSource | Database prepared in advance | No | None |
| var | Name of the variable to represent the database | No | Set default |
| scope | Scope of the variable to represent the database | No | Page |

31.what is spring ioc

**Inversion of Control** (**IOC**) and Dependency Injection (DI) are used interchangeably. ... By DI the responsibility of creating objects is shifted from our application code to **Spring** container hence the phenomenon is called **IOC**. Dependency Injection can be done by setter injection, constructor injection.

32.what are the spring ioc annotations used and expalin them

| **ANNOTATION** | **USE** | **DESCRIPTION** |
| --- | --- | --- |
| @Autowired | Constructor, Field, Method | Declares a constructor, field, setter method, or configuration method to be autowired by type. Items annotated with @Autowired do not have to be public. |
| @Configurable | Type | Used with <context:springconfigured> to declare types whose properties should be injected, even if they are not instantiated by Spring. Typically used to inject the properties of domain objects. |
| @Order | Type, Method, Field | Defines ordering, as an alternative to implementing the org. springframework.core.Ordered interface. |
| @Qualifier | Field, Parameter, Type, Annotation Type | Guides autowiring to be performed by means other than by type. |
| @Required | Method (setters) | Specifies that a particular property must be injected or else the configuration will fail. |
| @Scope | Type | Specifies the scope of a bean, either singleton, prototype, request, session, or some custom scope. |

### Stereotyping Annotations

| **ANNOTATION** | **USE** | **DESCRIPTION** |
| --- | --- | --- |
| @Component | Type | Generic stereotype annotation for any Spring-managed component. |
| @Controller | Type | Stereotypes a component as a Spring MVC controller. |
| @Repository | Type | Stereotypes a component as a repository. Also indicates that SQLExceptions thrown from the component's methods should be translated into Spring DataAccessExceptions. |
| @Service | Type | Stereotypes a component as a service. |

33.flow of spring MVC

34.how u done validations in your project

Front end validation with Javascript

Backend validation with spring validator

35.tell some annotation used while writing validation

36.what is the configuratio file in spring

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Deloitte\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Tell me about yourself?

2.In which technologies you have experience?

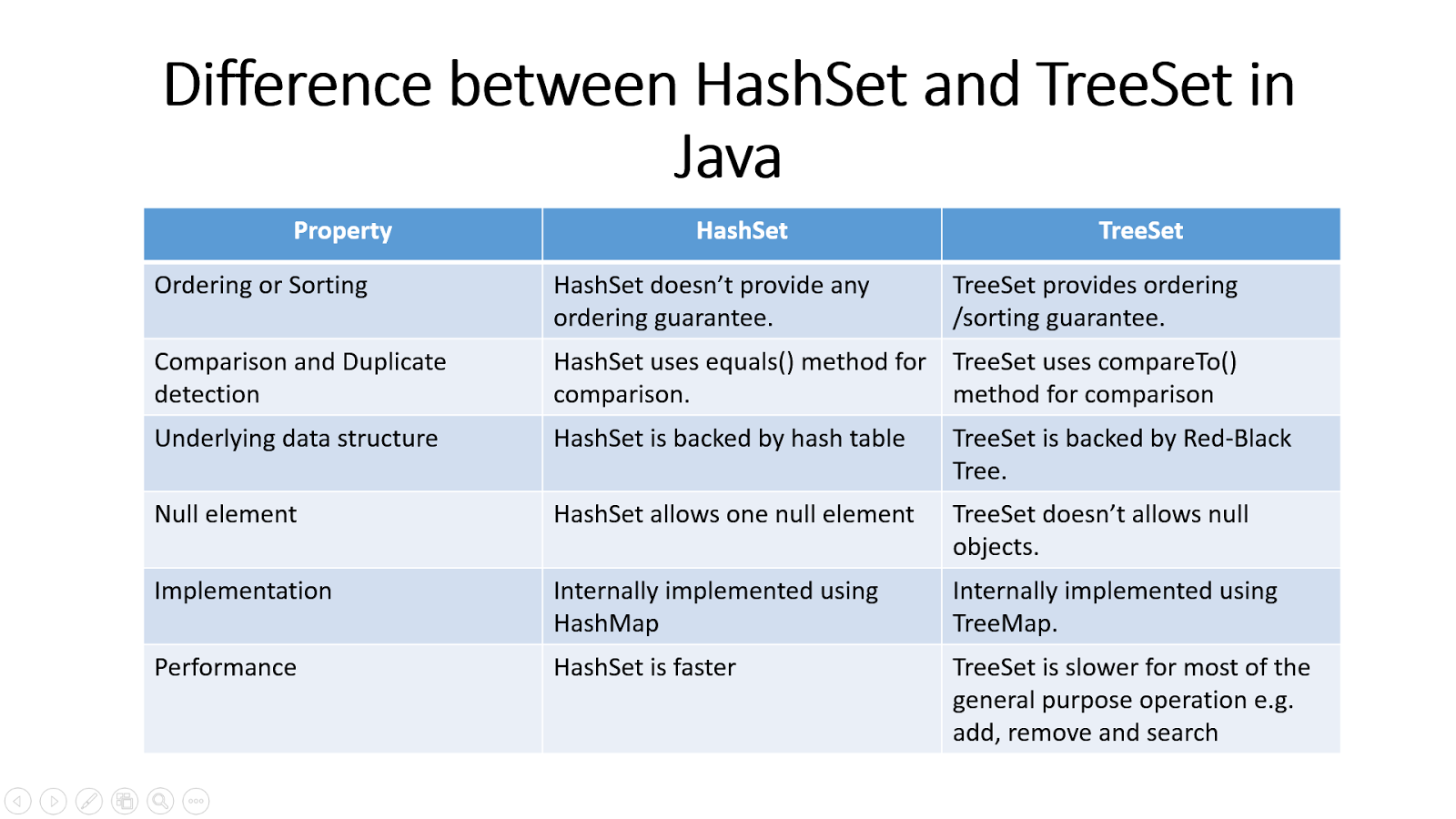
3.what is method overriding? write an example?

4.If you take private cmethod in parent, is this override in child?

5.If you take parent method throws Exception,whathappend in child?

6.what is the difference between set and Treeset?

7.what is the difference between Hashset and Treeset?



8.what is the diff. b/w Arraylist and Linkedlist?

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1) ArrayList internally uses **dynamic array** to store the elements. | LinkedList internally uses **doubly linked list** to store the elements. |
| 2) Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory. |
| 3) ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. |
| 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data. |

9.what is the diff. b/w Hashmap and concrete hashmap?

10.If, Emp e1=new Emp(1);

Emp e2=new Emp(1);

How jvm knows it is duplicate in Hashset?

11.what is singleton?what it’s purpose?write a program?

12.what is marker interface?what is it’s purpose?

**Marker Interface** in java is an **interface** with no fields or methods within it. It is used to convey to the JVM that the class implementing an **interface** of this category will have some special behavior. Hence, an empty **interface** in java is called a **marker interface**.

13.Inhashcode() and equals(),who write the logic?

14.what is the purpose of callable interface?

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called call. The **Callable interface** is similar to Runnable , in that both are designed for classes whose instances are potentially executed by another thread.

15.how to configure one-to-many relationship in hibernate?

16.what is the use of cascade attribute?

Cascade attribute is mandatory, when ever we apply relationship between objects, cascade attribute transfers operations done on one object onto its related child objects

If we write cascade = “all” then changes at parent class object will be effected to child class object too,  if we write cascade = “all” then all operations like insert, delete, update at parent object will be effected to child object also

Example: if we apply insert(or update or delete) operation on parent class object, then child class objects will also be stored into the database.

default value of cascade =”none” means no operations will be transfers to the child class

Example: if we apply insert(or update or delete) operation on parent class object, then child class objects will not be effected, if cascade = “none”

Cascade having the values…….

* none (default)
* save
* update
* save-update
* delete
* all
* all-delete-orphan

17.diff. b/w HQL and Criteria?

* HQL is to perform both select and non-select operations on the data,  but Criteria is only for selecting the data, we cannot perform non-select operations using criteria
* HQL is suitable for executing Static Queries, where as Criteria is suitable for executing Dynamic Queries
* HQL doesn’t support pagination concept, but we can achieve pagination with Criteria
* Criteria used to take more time to execute then HQL
* With Criteria we are safe with SQL Injection because of its dynamic query generation but in HQL as your queries are either fixed or parametrized, there is no safe from SQL Injection.

18.what is springIOC?

**Inversion of Control** (IoC) is the mechanism to achieve loose-coupling between Objects dependencies. To achieve loose coupling and dynamic binding of the objects at runtime, the objects define their dependencies that are being injected by other assembler objects. Spring IoC container is the program that injects dependencies into an object and make it ready for our use.

Spring Framework IoC container classes are part of org.springframework.beans and org.springframework.context packages and provides us different ways to decouple the object dependencies.

19.You have a bean.you want more objects for that bean.How you configure that in spring.xml?

Ans: bean scope=”prototype”

20.what is the default scope of a bean in spring?

Ans: singleton

21.suppose, <bean id=”s1” class=”Student” scope=”singleton”/>

<bean id=”s2” class=”Student” scope=”singleton”/>

For this how many objects are created?

Ans: Two

22.what is the diff. b/w restful and soap webservices?

There are many differences between SOAP and REST web services. The important 10 differences between SOAP and REST are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services. |
| 6) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 7) | SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. |
| 8) | SOAP **defines its own security**. | RESTful web services **inherits security measures** from the underlying transport. |
| 9) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| 10) | SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |

................. SavenTechnical(Technical Round) ..................

How much rating will you give for core java

1)what is Your roles & res in project?

2)What are the collection interfaces used in your project?

3)Where you have used the collection interfaces?

4)What is map and it's struture? Explainevery thingbrefiely?

5)Difference between list and set?

A **Set** cannot contain duplicate elements while a **List** can. A **List** (in Java) also implies order. Conceptually we usually refer to an unordered grouping that allows duplicates as a Bag and doesn't allow duplicates is a **Set**. **List** is used to collection of elements with duplicates.

6)What is marker interface?

**Marker Interface** in java is an **interface** with no fields or methods within it. It is used to convey to the JVM that the class implementing an **interface** of this category will have some special behavior. Hence, an empty **interface** in java is called a **marker interface**.

7)What is marker interface tells to the jvm?

. It is used to convey to the JVM that the class implementing an **interface** of this category will have some special behavior.

8)Tell me some marker interfaces?

**Marker interface** in **Java**. It is an empty **interface** (no field or methods). **Examples**of **marker interface** are Serializable, Clonnable and Remote **interface**. ... By convention, classes that implement this **interface** should override Object.clone() method.

9)Did you used serilization in your project? And In which situation you have used?

10)Which version of java you have used?

JDK 1.8

11) Tell me what are the changes are added in before 1.7(features of 1.6) and after 1.7(features of 1.8) and features of 1.7?

1.8 features

1. [forEach() method in Iterable interface](http://www.journaldev.com/2389/java-8-features-with-examples#iterable-forEach)
2. [default and static methods in Interfaces](http://www.journaldev.com/2389/java-8-features-with-examples#interface-default-static-method)
3. [Functional Interfaces and Lambda Expressions](http://www.journaldev.com/2389/java-8-features-with-examples#functional-interface-lambdas)
4. [Java Stream API for Bulk Data Operations on Collections](http://www.journaldev.com/2389/java-8-features-with-examples#java-stream-api)
5. [Java Time API](http://www.journaldev.com/2389/java-8-features-with-examples#java8-time)
6. [Collection API improvements](http://www.journaldev.com/2389/java-8-features-with-examples#java8-collection)
7. [Concurrency API improvements](http://www.journaldev.com/2389/java-8-features-with-examples#java8-concurrency)
8. [Java IO improvements](http://www.journaldev.com/2389/java-8-features-with-examples#java8-io)
9. [Miscellaneous Core API improvements](http://www.journaldev.com/2389/java-8-features-with-examples#java8-core)

1.7 features

**1.** String in Switch Expression  
**2.** Underscores Between Digits in Numeric Literals  
**3.** Integral Types as Binary Literals  
**4.** Handling multiple exceptions in a single catch block  
**5.** Try-with-resources Statement  
**6.** Automatic Type Inference in Generic object instantiation

12)What is mockito and junit?

**Mockito** is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. **Mockito** is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

**JUnit** is a unit testing framework for the Java programming language. **JUnit** has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

13) Find highest sal of person details? (o/p:101,xyz,25000)

|  |  |  |
| --- | --- | --- |
| Empno | Ename | Esal |
| 101 | Xyz | 25000 |
| 102 | Lmn | 5000 |
| 103 | Abc | 30000 |

14) I have map with out using system.out.println() swap the key as a value and value as key?

Map<String, Character> myNewHashMap = new HashMap<>();

for(Map.Entry<Character, String> entry : myHashMap.entrySet()){

myNewHashMap.put(entry.getValue(), entry.getKey());

}

Alternatively, you can use a Bi-Directional map like [Guava](http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/collect/BiMap.html) provides and use the [inverse()](http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/collect/BiMap.html#inverse%28%29)method :

BiMap<Character, String> myBiMap = HashBiMap.create();

myBiMap.put('a', "test one");

myBiMap.put('b', "test two");

BiMap<String, Character> myBiMapInversed = myBiMap.inverse();

As [java-8](https://stackoverflow.com/questions/tagged/java-8) is out, you can also do it this way :

Map<String, Integer> map = new HashMap<>();

map.put("a",1);

map.put("b",2);

Map<Integer, String> mapInversed =

map.entrySet()

.stream()

.collect(Collectors.toMap(Map.Entry::getValue, Map.Entry::getKey))

15)What is immutable class?How to write immutable class?

**Immutable class** is a **class**which once created, it's contents can not be changed. **Immutable** objects are the objects whose state can not be changed once constructed. e.g. String**class**.

public final class Immutable

{

private final String name;

public Immutable(String name)

{

this.name = name;

}

public String getName() { return this.name; }

// No setter;

}

16)What is IOC,dependency injection?

**Inversion of Control** (**IOC**) and Dependency Injection (DI) are used interchangeably. ... By DI the responsibility of creating objects is shifted from our application code to **Spring** container hence the phenomenon is called **IOC**. Dependency Injection can be done by setter injection, constructor injection. Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

* Resolving the dependencies among the beans is nothing but Dependency Injection.

17)Flow of Spring mvc?

18)What is svn?Do have any idea about git hub?

19)In your project if bug will come how to fix it?

\*\*\*\*\*Tripod Solutions\*\*\*\*\*\*\*\*

1.Tell me about yourself?

**\*\*\*\*\*Somasekhar\*\*\*\*\*\*\*\***

----- @@@@ SAGGEZZA @@@@ ------ (may 10,2016)

>1.Tell me about ur project

>2.Take a Sample string find out the duplicate word and how many times on comes..in which place

" html on html on"

ANS-->import java.util.Arrays;

import java.util.Collections;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class DuplicateWordSearcher {

@SuppressWarnings("unchecked")

public static void main(String[] args) {

String text = "a r b k c d se f g a d f s s f d s ft gh f ws w f v x s g h d h j j k f sd j e wed a d f";

List<String> list = Arrays.asList(text.split(" "));

Set<String> uniqueWords = new HashSet<String>(list);

for (String word : uniqueWords) {

System.out.println(word + ": " + Collections.frequency(list, word));

}

}

}

>3.How to convert string to array

ANS--->public class JavaStringToStringArrayExample {

public static void main(String args[]){

String str = "Java String to String Array Example";

/\*

\* To convert String object to String array, first thing

\* we need to consider is to how we want to create array.

\*

\* In this example, array will be created by words contained

\* in the original String object. So, first element of array

\* will contain "java", second will contain "String" and so on.

\*

\* To convert String to String array, use

\* String[] split(String delimiter) method of Java String

\* class as given below.

\*/

String strArray[] = str.split(" ");

System.out.println("String converted to String array");

//print elements of String array

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

System.out.println(strArray[i]);

}

}

}

4. **What is the use of SVN?**

A. SVN is Apache Subversion which is often abbreviated as SVN, it is a software versioning and revision control system distributed under an open source license.

5. **What is Version Control System?**

A. Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.

Following are the goals of a Version Control System.

• Allow developers to work simultaneously.

• Do not overwrite each other’s changes.

• Maintain history of every version of everything.

6. **What is load balancing?**

A. Load balancing is a method for distributing tasks onto multiple computers. For instance, distributing incoming HTTP requests (tasks) for a web application onto multiple web servers. There are a few different ways to implement load balancing.

7**. Why object is not required for calling static methods?**

If u use Static keyword at variable level, method level it will be loaded at the time of Class loading, so it will be available in Class area. So no need to create an instance to call Static Methods.

8. **How to create your own exception?**

A. You just need to extend the Exception class to create your own Exception class. These are considered to be checked exceptions. If you want to write a unchecked or runtime exception, you need to extend the RuntimeException class.

Exp: Java defines several built-in classes for exception handling. All these classes are a part of the java.lang package which is automatically imported. That is why, while throwing an exception you did not specifically include any package for exception classes.

Apart from the built-in exception classes you can define your own exception classes. This is very useful when you want to define exception types which are having a behavior different from the standard exception type, particularly when you want to do validations in your application

All exceptions must be a child of Throwable.

9. **Which spring modules have you used in your project?**

Spring IOC, DAO, ORM, MVC.

10. **Spring dependency types.**

A. Constructor Injection and Setter Injection.

Exp: You can mix both, Constructor-based and Setter-based DI but it is a good rule of thumb to use constructor arguments for mandatory dependencies and setters for optional dependencies.

11. **Are u using xml based or annotation based dependencies?**

A. Annotation based @AutoWired.

12. **Explain collection framework?**

A. Collections in java is a framework that provides an architecture to store and manipulate the group of objects.

All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

13. **What are the legacy classes present in collection?**

A.

**Legacy Classes**

Early version of java did not include the Collection framework. It only defined several classes and interface that provide method for storing objects. When Collection framework were added in J2SE 1.2, the original classes were reengineered to support the collection interface. These classes are also known as Legacy classes. All legacy claases and interface were redesign by JDK 5 to support Generics.

The following are the legacy classes defined by java.util package

Dictionary

HashTable

Properties

Stack

Vector

There is only one legacy interface called Enumeration

NOTE: All the legacy classes are synchronized

**14. What are the cursors in collection? On which interfaces these cursors are used?**

A. Cursors in Collection framework

To retrieve objects one by one from Collection we have to use cursors.

There are 3 types of cursors available

1. Enumeration

2. Iterator

3. ListIterator

Enumeration

This interface was introduced in 1.0 version it contains the following 2 methods.

public boolean hasMoreElements()

public Object nextElement()

Limitations of Enumeration

It is applicable only for legacy classes and it is not a universal cursor.

While iterating the objects by using enumeration we can get only read access and we can’t perform removal or replacement operations.

To overcome these problems we should go for Iterator interface.

Iterator

It is Introduced in 1.2 version.

We can get Iterator Object for any collection implemented class hence it is universal cursor.

while iterating objects, we are allowed perform removal operation in addition to read operation.

This interface contains the following 3 methods.

public boolean hasNext()

public Object next()

public void remove()

Enumeration and Iterator are single directional cursors. They can always move towords forward direction only.

By using Iterator we can perform read and remove operations. And we can’t perform any replace or addition of new objects

To over come these limitations we should go for ListIterator interface.

**ListIterator**

It has introduced in 1.2 version and it is child interface of Iterator.

It is a bi-directional cursor,i.e we can move either to the forward or backward direction.

While Iterating we can perform read,remove,replace and addition of new objects.

This interface defines the following 9 methods.

public boolean hasNext();

public boolean hasPrevious();

public Object next();

public Object previous();

public int nextIndex();

If there is no next element it returns size of the list.

public int previousIndex();

If there is no previous element it returns -1 .

public void remove();

public void set(Object new)

public void set(Object new)

public void add(Object new)

15. **What are the default methods of Object class?**

A. Object class in Java

Object class is present in java.lang package. Every class in Java is directly or indirectly derived from the Object class. If a Class does not extend any other class then it is direct child class of Object and if extends other class then it is an indirectly derived. Therefore the Object class methods are available to all Java classes. Hence Object class acts as a root of inheritance hierarchy in any Java Program.

Using Object class methods

There are 12 methods in Object class:

toString() : toString() provides String representation of an Object and used to convert an object to String. The default toString() method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@’, and the unsigned hexadecimal representation of the hash code of the object. It is always recommended to override toString() method to get our own String representation of Object

Note : Whenever we try to print any Object reference, then internally toString() method is called.

hashCode() : For every object, JVM generates a unique number which is hashcode. It returns distinct integers for distinct objects. A common misconception about this method is that hashCode() method returns the address of object, which is not correct. It convert the internal address of object to an integer by using an algorithm. The hashCode() method is native because in Java it is impossible to find address of an object, so it uses native languages like C/C++ to find address of the object.

Use of hashCode() method : Returns a hash value that is used to search object in a collection. JVM(Java Virtual Machine) uses hashcode method while saving objects into hashing related data structures like HashSet, HashMap, Hashtable etc. The main advantage of saving objects based on hash code is that searching becomes easy.

Note : Override of hashCode() method needs to be done such that for every object we generate a unique number. For example,for a Student class we can return roll no. of student from hashCode() method as it is unique.

equals(Object obj) : Compares the given object to “this” object (the object on which the method is called). It gives a generic way to compare objects for equality. It is recommended to override equals(Object obj) method to get our own equality condition on Objects. For more on override of equals(Object obj) method refer – Overriding equals method in Java

Note : It is generally necessary to override the hashCode() method whenever this method is overridden, so as to maintain the general contract for the hashCode method, which states that equal objects must have equal hash codes.

getClass() : Returns the class object of “this” object and used to get actual runtime class of the object. It can also be used to get metadata of this class. The returned Class object is the object that is locked by static synchronized methods of the represented class. As it is final so we don’t override it.

Note :After loading a .class file, JVM will create an object of the type java.lang.Class in the Heap area. We can use this class object to get Class level information. It is widely used in Reflection

finalize() method : This method is called just before an object is garbage collected. It is called by the Garbage Collector on an object when garbage collector determines that there are no more references to the object. We should override finalize() method to dispose system resources, perform clean-up activities and minimize memory leaks. For example before destroying Servlet objects web container, always called finalize method to perform clean-up activities of the session.

Note :finalize method is called just once on an object even though that object is eligible for garbage collection multiple times.

clone() : It returns a new object that is exactly the same as this object. For clone() method refer Clone()

The remaining three methods wait(), notify() notifyAll() are related to Concurrency.

wait()-It tells the calling thread to give up the lock and go to sleep until some other thread enters the same monitor and calls notify().

notify()-It wakes up one single thread that called wait() on the same object. It should be noted that calling notify() does not actually give up a lock on a resource.

notifyAll()-It wakes up all the threads that called wait() on the same object.

16. **What is the purpose of hashcode() method?**

Answer:

The hashcode of a Java Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure. We know that hash code is an unique id number allocated to an object by JVM. But actually speaking, Hash code is not an unique number for an object. If two objects are equals then these two objects should return same hash code. So we have to implement hashcode() method of a class in such way that if two objects are equals, ie compared by equal() method of that class, then those two objects must return same hash code. If you are overriding hashCode you need to override equals method also.

17. **Explain about spring IOC? What is the use of IOC?**

A. The core of the Spring Framework is its Inversion of Control (Ioc) container. The IoC container manages java objects – from instantiation to destruction – through its BeanFactory. Java components that are instantiated by the IoC container are called beans, and the IoC container manages a bean's scope, lifecycle events, and any AOP features for which it has been configured and coded.

The IoC container enforces the dependency injection pattern for your components, leaving them loosely coupled and allowing you to code to abstractions.

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

to instantiate the application class

to configure the object

to assemble the dependencies between the objects

There are two types of IoC containers. They are:

BeanFactory

ApplicationContext

18. **What are the properties configured in configure file and write the configuration file using all the properties?**

A. Spring allows us to externalize String literals in its context configuration files into external properties files in order to separate application-specific settings from framework-specific configuration.

Spring will read all the properties files declared by PropertyPlaceholderConfigurer bean to resolve the placeholders at application’s start up time.

Declare a PropertyPlaceholderConfigurer bean in Spring’s application context file as follows:

<bean id="properties"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value=" properties file path" />

</bean>

That tells Spring to load the properties file named “properties” in the classpath to resolve any placeholders ${…} found. An exception will be thrown if Spring could not find the specified properties file.

And the following bean declaration uses some placeholders which will be resolved by Spring:

<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">

<property name="host" value="${smtp.host}" />

<property name="port" value="${smtp.port}" />

<property name="username" value="${smtp.user}" />

<property name="password" value="${smtp.pass}" />

</bean>

Spring will replace these placeholders by actual values of the corresponding entries in the properties file. An exception will be thrown if a placeholder could not be resolved, e.g there is no entry with the specified key.

2. Location of the properties files

By default, Spring looks for the properties files in the application’s directory. So if we specify:

<property name="location" value="WEB-INF/mail.properties" />

Then it will find the mail.properties file under WEB-INF directory of the application (in case of a Spring MVC application).

We can use the prefix classpath: to tell Spring loads a properties file in the application’s classpath. For example:

<property name="location" value="classpath:mail.properties" />

In case of a Spring MVC application, the mail.properties file should be present in the WEB-INF/classes directory (or in the source directory (src) in Eclipse IDE).

Use the prefix file:/// or file: to load a properties file from an absolute path. For example:

<property name="location" value="file:///D:/Config/mail.properties" />

NOTE: There is no white space between the prefixes classpath: and file: with the path of properties file.

3. Loading multiple properties files

Spring allows us to specify multiple properties files for the PropertyPlaceholderConfigurer bean declaration as follows:

<bean id="appProperties"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath:mail.properties</value>

<value>classpath:database.properties</value>

</list>

</property>

</bean>

That tells Spring to resolve the placeholders with entries loaded from both mail.properties and database.properties files in the application’s classpath.

19. **OOPS concepts? Explain briefly? Where you used these concepts in your project?**

A. OOPs (Object Oriented Programming System)

Object means a real word entity such as pen, chair, table etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

Object

Class

Inheritance

Polymorphism

Abstraction

Encapsulation

**Object**

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

**Class**

Collection of objects is called class. It is a logical entity.

**Inheritance**

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**polymorphism in java oops concepts**

**Polymorphism**

When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

**Abstraction**

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.

encapsulation in java oops concepts

**Encapsulation**

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

20. **How to configure two servers if i send 4 request's?**

A.

21. **Maven configuration?**

A.

22. **i/o streams concept?**

A.

23. **How to deploy ur project in server?**

A.

24. **String s[]={“a.pdf”, “b.txt”, ”c.pdf”} from this string[] how can you retrieve the .txt files?**

A.

25. **How to add new data in a .txt file which is already exist?**

A.

26. **What is form backing object? How to use?**

A.

27. Explain spring MVC architecture?

A.

|  |  |
| --- | --- |
| Step **1**: First request will be received by DispatcherServlet Step **2**: DispatcherServlet will take the help of HandlerMapping and get to know the Controller class name associated with the given request Step **3**: So request transfer to the Controller, and then controller will process the request by executing appropriate methods and returns ModelAndView object (contains Model data and View name) back to the DispatcherServlet | http://www.java4s.com/wp-content/uploads/2013/07/Spring-MVC-execution-flow.png |

Step 4: Now DispatcherServlet send the model object to the ViewResolver to get the actual view page  
Step **5**: Finally DispatcherServlet will pass the Model object to the View page to display the result

That’s it :-)

28. **What is the purpose of hibernate and spring framework? Why these frameworks are mostly used?**

A. Hibernate framework simplifies the development of java application to interact with the database. Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool.

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.

The ORM tool internally uses the JDBC API to interact with the database.

Advantages of Hibernate Framework

There are many advantages of Hibernate Framework. They are as follows:

1) Opensource and Lightweight: Hibernate framework is opensource under the LGPL license and lightweight.

2) Fast performance: The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled bydefault.

3) Database Independent query: HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, If database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Automatic table creation: Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

5) Simplifies complex join: To fetch data form multiple tables is easy in hibernate framework.

6) Provides query statistics and database status: Hibernate supports Query cache and provide statistics about query and database status.

**Spring Framework**

Spring is a lightweight framework. It can be thought of as a framework of frameworks because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

next →

**Spring Tutorial**

**spring framework tutorial**

This spring tutorial provides in-depth concepts of Spring Framework with simplified examples. It was developed by Rod Johnson in 2003. Spring framework makes the easy development of JavaEE application.

It is helpful for beginners and experienced persons.

**Spring Framework**

Spring is a lightweight framework. It can be thought of as a framework of frameworks because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

**Inversion Of Control (IOC) and Dependency Injection**

These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain.

In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

Advantages of Spring Framework

There are many advantages of Spring Framework. They are as follows:

1) Predefined Templates

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

2) Loose Coupling

The Spring applications are loosely coupled because of dependency injection.

3) Easy to test

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

4) Lightweight

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

5) Fast Development

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

6) Powerful abstraction

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

7) Declarative support

It provides declarative support for caching, validation, transactions and formatting.

29. **To which tables you applied joins in your project?**

A.

30. **Which mapping relationships are used for your tables?**

A. Many to one and One to many.

31. **What is purpose of <bag> tag?**

A. If your table does not have an index column, and you still wish to use List as the property type, you can map the property as a Hibernate < bag>. A bag does not retain its order when it is retrieved from the database, but it can be optionally sorted or ordered.

A bag is an unordered, unkeyed collection that can contain the same element multiple times. A bag permits duplicates, so it has no primary key. The Java collections API, curiously, has no Bag.

You don't need to use the <index> tag when mapping an unsorted List and you also don't need an additional index column in database.

Unlike a List a Bag does not persist the order of its elements, but you can specify an order-by parameter to retrieve its elements in a specific order.

32. **Tell me about yourself?**

A.

33. **What is your project , roles and responsibilities?**

A.

34. **How do you write the testcase? What is the class name that is used to run your test class?**

A.

35. **Why obj is not required for calling static methods?**

A. If we use static keyword at variable level and method level, it will be loaded at the time of class loading. So it will be available in class area. So no need to create an instance to call static methods.

UHG

1. Tell me about yourself
2. What is your project , roles and responsibilities.
3. What is the use of SVN?

Ans: *Subversion is a free/open source*version control system*(VCS). That is, Subversion manages files and directories, and the changes made to them, over time. This allows you to recover older versions of your data or examine the history of how your data changed.*

1. How do you write the testcase? What is the class name that is used to run your test class?

Ans: *A*test case*is a set of conditions or variables under which a tester will determine whether a system under*test*satisfies requirements or works correctly. @Test* is the class name that is used to run your test class.

1. How you connected to the database?

Ans:

*//creating configuration*

*Configuration cfg=****new****Configuration();*

*cfg.configure("hibernate.cfg.xml");*

*//creating seession factory object*

*SessionFactory factory=cfg.buildSessionFactory();*

*//creating session object*

*Session session=factory.openSession();*

*//creating transaction object*

*Transaction t=session.beginTransaction();*

*Employee e1=****new****Employee(111,"arun",40000);*

*session.persist(e1);//persisting the object*

*t.commit();//transaction is commited*

*session.close();*

*(or) Using basic datasource in spring.*

1. Write the xml file for getting hibernate template?

Ans: Spring.xml

*<?xml version="1.0" encoding="UTF-8"?>*

*<beans*

*xmlns="http://www.springframework.org/schema/beans"*

*xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xmlns:p="http://www.springframework.org/schema/p"*

*xsi:schemaLocation="http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">*

*<bean id="dataSource"****class****="org.apache.commons.dbcp.BasicDataSource">*

*<property name="driverClassName"  value="oracle.jdbc.driver.OracleDriver"></property>*

*<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"></property>*

*<property name="username" value="system"></property>*

*<property name="password" value="oracle"></property>*

*</bean>*

*<bean id="mysessionFactory"****class****="org.springframework.orm.hibernate3.LocalSessionFactoryBean">*

*<property name="dataSource" ref="dataSource"></property>*

*<property name="mappingResources">*

*<list>*

*<value>employee.hbm.xml</value>*

*</list>*

*</property>*

*<property name="hibernateProperties">*

*<props>*

*<prop key="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</prop>*

*<prop key="hibernate.hbm2ddl.auto">update</prop>*

*<prop key="hibernate.show\_sql">****true****</prop>*

*</props>*

*</property>*

*</bean>*

*<bean id="template"****class****="org.springframework.orm.hibernate3.HibernateTemplate">*

*<property name="sessionFactory" ref="mysessionFactory"></property>*

*</bean>*

*<bean id="d"****class****="com.javatpoint.EmployeeDao">*

*<property name="template" ref="template"></property>*

*</bean>*

*</beans>*

1. What is the use of log4J and what are the appenders you used mostly?

Ans:

1. With Log4j we can store the flow details of  our Java/J2EE in a file or databases.

2. we use log statements rather SOPL statements in the code to know the status of a project while it is executing.

3. File appender and Console Appender are appenders used.

1. What is slf4J?

Ans: Simple Logging Facade for Java (**SLF4J**) provides a Java logging API by means of a simple facade pattern. The underlying logging backend is determined at runtime by adding the desired binding to the classpath and may be the standard Sun Java logging package java.util.logging, log4j, logback or tinylog.

1. What is the difference between forward and sendRedirect?|

Ans:

**Forward()**

**SendRediret()**

When we use forward method request is transfer to other resource within the same server for further processing.

In case of sendRedirect request is transfer to another resource to different domain or different server for futher processing.

In case of forward Web container handle all process internally and client or browser is not involved.

When you use SendRedirect container transfers the request to client or browser so url given inside the **sendRedirect**method is visible as a new request to the client.

When forward is called on **requestdispather** object we pass request and response object so our old request object is present on new resource which is going to process our request

In case of SendRedirect call old request and response object is lost because it’s treated as new request by the browser.

Visually we are not able to see the forwarded address, its is transparent

In address bar we are able to see the new redirected address it’s not transparent.

Using forward () method is faster then send redirect.

SendRedirect is slower because one extra round trip is required beasue completely new request is created and old request object is lost.Two browser request requird.

When we redirect using forward and we want to use same data in new resource we can use request.setAttribute () as we have request object available.

But in sendRedirect if we want to use we have to store the data in session or pass along with the URL.

1. What are the different session tracking mechanisms and what is the best among them?

There are four techniques used in Session tracking:

1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession
5. How to handle the exception in jsp pages?

To handle exceptions thrown by the JSP page, all we need is an error page and define the error page in JSP using [jsp page directive](http://www.journaldev.com/2044/jsp-directives-page-include-taglib-example).

To create a JSP error page, we need to set page directive attribute **isErrorPage** value to true, then we can access exception [jsp implicit object](http://www.journaldev.com/2038/jsp-implicit-objects) in the JSP and use it to send customized error message to the client.

1. What are the implicit objects?

Ans:

JSP implicit objects are created during the translation phase of JSP to the servlet.

These objects can be directly used in scriplets that goes in the service method.

They are created by the container automatically, and they can be accessed using objects.

There are 9 types of implicit objects available in the container:

1. out
2. request
3. response
4. config
5. application
6. session
7. pageContext
8. page
9. exception
10. What is the main class of all java classes?

Ans: java.lang.**Object** class is the super base class of all Java classes.

1. What are the methods of object class? What is the use of hashcode()?

Ans: Methods of Object Class are

1. public final Class **getClass()**
2. public int **hashCode()**
3. public boolean **equals(Object obj)**
4. protected Object **clone()** throws CloneNotSupportedException
5. public String **toString()**
6. public final void **notify()**
7. public final void **notifyAll()**
8. public final void **wait(long timeout)**throws InterruptedException
9. public final void **wait(long timeout,int nanos)**throws InterruptedException
10. public final void **wait()**throws InterruptedException
11. protected void **finalize()**throws Throwable
    * If the objects are stored according to hashCode searching will become very efficient (The most powerful search algorithm is hashing which will work based on hashCode).
12. What is the difference between hashcode() and tostring()?
13. if we are giving opportunity to Object class toString() method it internally calls hashCode() method. But if we are overriding toString() method it may not call hashCode() method.
14. We can use toString() method while printing object references and we can use hashCode() method while saving objects into HashSet or Hashtable or HashMap.
15. Where you configure database properties if you use spring?

Spring.xml

1. What are the advantages of spring over struts?
2. Write the code for getting a object using Ajax?
3. By using JQuery how can you get list of employee?
4. Explain spring mvc architecture?



1. What is the use of view resolver and handler mapping?

**InternalResourceViewResolver** is used to resolve “internal resource view” (in simple, it’s final output, jsp or htmp page) based on a predefined URL pattern. In additional, it allow you to add some predefined prefix or suffix to the view name (prefix + view name + suffix), and generate the final view page URL.

HandlerMapping is an interface that is implemented by all Objects that map the request to the corresponding Handler Object. A HandlerExecutionChain always wraps the Handler and may also contain one or more HandlerInterceptors. The default Implementations used by the DispatcherServlet are BeanNameUrlHandlerMapping and DefaultAnnotationHandlerMapping.

1. Write the code of singleton?

publicclassASingleton{

privatestaticASingleton instance =null;

privateASingleton(){

}

publicstaticASingleton getInstance(){

if(instance ==null){

instance =newASingleton();

}

return instance;

}

1. Can a try block can exist with out catch?

No.

1. What is the IDE you used?

Eclipse

1. How you deploy your application from your local system?
2. Have you used criteria in your project?

Yes, The **Criteria** API provides the org.**hibernate**.**criterion**.Projections class which can be used to get average, maximum or minimum of the property values. The Projections class is similar to the Restrictions class in that it provides several static factory methods for obtaining Projection instances.

1. Which collection interfaces you have used?

List, Set, Map

1. What is best for retrieving purpose and what is the best for adding and deleting?
2. What is one to many relationship write the code?

An example of Student and Standard (grade) entities where one Standard can include many Students. So the relation between Student and Standard entities would be one-to-many.

Eg: publicclassStudent

{

public Student() { }

publicint StudentId { get; set; }

publicstring StudentName { get; set; }

publicvirtualStandard Standard { get; set; }

}

publicclassStandard

{

public Standard()

{

Students = newList<Student>();

}

publicint StandardId { get; set; }

publicstring Description { get; set; }

publicvirtualICollection<Student> Students { get; set; }

}

1. What is exception?

An unwanted, unexpected event that disturbs normal flow of execution is called exception. Every exception need to be handled.

1. What is the super class of exception? How can we handle it?

Throwable is the super class of exception. We can handle with try, catch or throws.

1. What is the server you used?

Tomcat.

1. Where can we use JSON object?

JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type. The keys and values are separated by colon. Each key/value pair is separated by comma.

The curly brace **{** represents JSON object.

JSON.stringify turns a Javascript object into JSON text and stores that JSON text in a string.

JSON.parse turns a string of JSON text into a Javascript object.

1. What is Mockito?

**Mockito** is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. **Mockito** is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

1. There is a employee class and employee class has three attributes and assume that data is present in file or database. How can you get the employee object with only two attributes?
2. What is the difference between final and static?

## The static keyword can be used in 4 scenarios

* static variables
* static methods
* static blocks of code.
* static nested class

Lets look at static variables and static methods first.

## static variable

* It is a variable which belongs to the class and not to object(instance)
* Static variables are initialized only once , at the start of the execution . These variables will be initialized first, before the initialization of any instance variables
* A single copy to be shared by all instances of the class
* A static variable can be accessed directly by the class name and doesn’t need any object
* Syntax : Class.variable

## static method

* It is a method which belongs to the class and not to the object(instance)
* A static method can access only static data. It can not access non-static data (instance variables) unless it has/creates an instance of the class.
* A static method can call only other static methods and can not call a non-static method from it unless it has/creates an instance of the class.
* A static method can be accessed directly by the class name and doesn’t need any object
* Syntax : Class.methodName()
* A static method cannot refer to this or super keywords in anyway

## static class

Java also has "static nested classes",A static nested class is just one which doesn't implicitly have a reference to an instance of the outer class.

Static nested classes can have instance methods and static methods.

There's no such thing as a top-level static class in Java.

### **Side Note:**

main method is static since it must be be accessible for an application to run before any instantiation takes place.

## final keyword is used in several different contexts to define an entity which cannot later be changed.

* A final class cannot be subclassed. This is done for reasons of security and efficiency. Accordingly, many of the Java standard library classes are final, for example java.lang.System and java.lang.String. All methods in a final class are implicitly final.
* A final method can't be overridden by subclasses. This is used to prevent unexpected behavior from a subclass altering a method that may be crucial to the function or consistency of the class.
* A final variable can only be initialized once, either via an initializer or an assignment statement. It does not need to be initialized at the point of declaration: this is called a blank final variable. A blank final instance variable of a class must be definitely assigned at the end of every constructor of the class in which it is declared; similarly, a blank final static variable must be definitely assigned in a static initializer of the class in which it is declared; otherwise, a compile-time error occurs in both cases.

### Note: If the variable is a reference, this means that the variable cannot be re-bound to reference another object. But the object that it references is still mutable, if it was originally mutable.

When an anonymous inner class is defined within the body of a method, all variables declared finalin the scope of that method are accessible from within the inner class. Once it has been assigned, the value of the final variable cannot change.

1. What happens if a class is final?

Ans:

Java class with final modifier is called final [class in Java](http://javarevisited.blogspot.com/2011/10/class-in-java-programming-general.html). Final class is complete in nature and can not be sub-classed or inherited. Several classes in Java are final e.g. String, Integer and other wrapper classes.

1. Final keyword improves performance. Not just JVM can cache **final variable** but also application can cache frequently use final variables.

2. Final variables are safe to share in [multi-threading](http://javarevisited.blogspot.com/2011/02/how-to-implement-thread-in-java.html) environment without additional synchronization overhead.

3. **Final keyword**allows [JVM](http://javarevisited.blogspot.com/2011/12/jre-jvm-jdk-jit-in-java-programming.html) to optimized method, variable or class.

1. Can we have a private class?

Ans: **Java** doesnt allow a top level **class** to be **private**. Only 'public' or 'package'. ...**private classes** are allowed but only as inner or nested **classes**.If you **have a private** inner or nested **class**, then access is restricted to the scope of that outer**class**.

1. What is Webservices?

Ans: A **web service** is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a **web service**. For example, a client invokes a **web service** by sending an XML message, then waits for a corresponding XML response.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

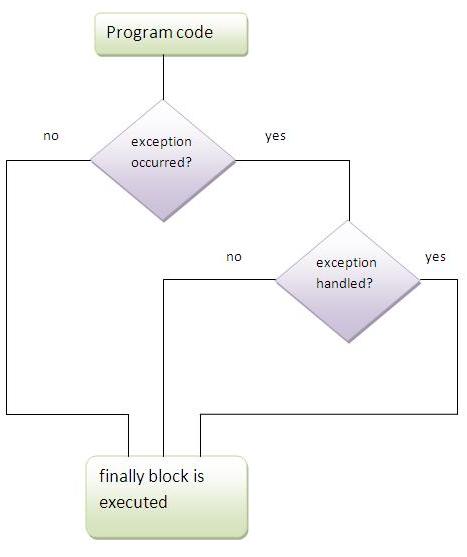
1. Why you want change your company?
2. What is the finally block?

# **Java finally block**

**Java finally block** is a block that is used to execute important code such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not.

Java finally block follows try or catch block.



## Why use java finally

* Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc.

1. What are the classes present in collection framework?

## The Collection Classes

Java provides a set of standard collection classes that implement Collection interfaces. Some of the classes provide full implementations that can be used as-is and others are abstract class, providing skeletal implementations that are used as starting points for creating concrete collections.

The standard collection classes are summarized in the following table −

|  |  |
| --- | --- |
| **Sr.No.** | **Class & Description** |
| 1 | **AbstractCollection**  Implements most of the Collection interface. |
| 2 | **AbstractList**  Extends AbstractCollection and implements most of the List interface. |
| 3 | **AbstractSequentialList**  Extends AbstractList for use by a collection that uses sequential rather than random access of its elements. |
| 4 | [**LinkedList**](https://www.tutorialspoint.com/java/java_linkedlist_class.htm)  Implements a linked list by extending AbstractSequentialList. |
| 5 | [**ArrayList**](https://www.tutorialspoint.com/java/java_arraylist_class.htm)  Implements a dynamic array by extending AbstractList. |
| 6 | **AbstractSet**  Extends AbstractCollection and implements most of the Set interface. |
| 7 | [**HashSet**](https://www.tutorialspoint.com/java/java_hashset_class.htm)  Extends AbstractSet for use with a hash table. |
| 8 | [**LinkedHashSet**](https://www.tutorialspoint.com/java/java_linkedhashset_class.htm)  Extends HashSet to allow insertion-order iterations. |
| 9 | [**TreeSet**](https://www.tutorialspoint.com/java/java_treeset_class.htm)  Implements a set stored in a tree. Extends AbstractSet. |
| 10 | **AbstractMap**  Implements most of the Map interface. |
| 11 | [**HashMap**](https://www.tutorialspoint.com/java/java_hashmap_class.htm)  Extends AbstractMap to use a hash table. |
| 12 | [**TreeMap**](https://www.tutorialspoint.com/java/java_treemap_class.htm)  Extends AbstractMap to use a tree. |
| 13 | [**WeakHashMap**](https://www.tutorialspoint.com/java/java_weakhashmap_class.htm)  Extends AbstractMap to use a hash table with weak keys. |
| 14 | [**LinkedHashMap**](https://www.tutorialspoint.com/java/java_linkedhashmap_class.htm)  Extends HashMap to allow insertion-order iterations. |
| 15 | [**IdentityHashMap**](https://www.tutorialspoint.com/java/java_identityhashmap_class.htm)  Extends AbstractMap and uses reference equality when comparing documents. |

1. What are the possible ways to debug the code?
2. What is the methodology?

A software development **methodology** or system development **methodology** in software engineering is a framework that is used to structure, plan, and control the process of developing an information system. ... Agile Software Development. Crystal Methods. Dynamic Systems Development Model (DSDM) Extreme Programming (XP).

# Agile Software Development Methodology

Agile software development is a conceptual framework for undertaking software engineering projects. There are a number of agilesoftware development methodologies e.g. Crystal Methods, Dynamic Systems Development Model (DSDM), and Scrum.

Most agile methods attempt to minimize risk by developing software in short timeboxes, called iterations, which typically last one to four weeks. Each iteration is like a miniature software project of its own, and includes all the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding, testing, and documentation. While iteration may not add enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration. At the end of each iteration, the team reevaluates project priorities.

Agile methods emphasize realtime communication, preferably face-to-face, over written documents. Most agile teams are located in a bullpen and include all the people necessary to finish the software. At a minimum, this includes programmers and the people who define the product such as product managers, business analysts, or actual customers. The bullpen may also include testers, interface designers, technical writers, and management .

Agile methods also emphasize working software as the primary measure of progress. Combined with the preference for face-to-face communication, agile methods produce very little written documentation relative to other methods.

Birlasoft

1. Where is your company located?
2. Tell me about your project?
3. What is the difference between stringbuffer and stringbuilder?

Ans:

|  |  |  |
| --- | --- | --- |
| **No.** | **StringBuffer** | **StringBuilder** |
| 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |

1. Write a list and add elements to the list and iterate the list.
2. **class** TestCollection1{
3. **public** **static** **void** main(String args[]){
4. ArrayList<String> list=**new** ArrayList<String>();//Creating arraylist
5. list.add("Ravi");//Adding object in arraylist
6. list.add("Vijay");
7. list.add("Ravi");
8. list.add("Ajay");
9. //Traversing list through Iterator
10. Iterator itr=list.iterator();
11. **while**(itr.hasNext()){
12. System.out.println(itr.next());
13. }
14. }
15. }
16. Explain the spring mvc architecture?

all the incoming request is intercepted by the DispatcherServlet that works as the front controller. The DispatcherServlet gets entry of handler mapping from the xml file and forwards the request to the controller. The controller returns an object of ModelAndView. The DispatcherServlet checks the entry of view resolver in the xml file and invokes the specified view component.

1. What are the method of Object class?
2. Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | [**protected Object clone()**](https://www.tutorialspoint.com/java/lang/object_clone.htm)  This method creates and returns a copy of this object. |
| 2 | [**boolean equals(Object obj)**](https://www.tutorialspoint.com/java/lang/object_equals.htm)  This method indicates whether some other object is "equal to" this one. |
| 3 | [**protected void finalize()**](https://www.tutorialspoint.com/java/lang/object_finalize.htm)  This method is called by the garbage collector on an object when garbage collection determines that there are no more references to the object. |
| 4 | [**Class<?> getClass()**](https://www.tutorialspoint.com/java/lang/object_getclass.htm)  This method returns the runtime class of this Object. |
| 5 | [**int hashCode()**](https://www.tutorialspoint.com/java/lang/object_hashcode.htm)  This method returns a hash code value for the object. |
| 6 | [**void notify()**](https://www.tutorialspoint.com/java/lang/object_notify.htm)  This method wakes up a single thread that is waiting on this object's monitor. |
| 7 | [**void notifyAll()**](https://www.tutorialspoint.com/java/lang/object_notifyall.htm)  This method wakes up all threads that are waiting on this object's monitor. |
| 8 | [**String toString()**](https://www.tutorialspoint.com/java/lang/object_tostring.htm)  This method returns a string representation of the object. |
| 9 | [**void wait()**](https://www.tutorialspoint.com/java/lang/object_wait.htm)  This method causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object. |
| 10 | [**void wait(long timeout)**](https://www.tutorialspoint.com/java/lang/object_wait_timeout.htm)  This method causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed. |
| 11 | [**void wait(long timeout, int nanos)**](https://www.tutorialspoint.com/java/lang/object_wait_nanos.htm)  This method causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed. |

1. Explain about generics?

The **Java Generics** programming is introduced in J2SE 5 to deal with type-safe objects.

Before generics, we can store any type of objects in collection i.e. non-generic. Now generics, forces the java programmer to store specific type of objects.

#### **Advantage of Java Generics**

There are mainly 3 advantages of generics. They are as follows:

**1) Type-safety :** We can hold only a single type of objects in generics. It doesn’t allow to store other objects.

**2) Type casting is not required:** There is no need to typecast the object.

Before Generics, we need to type cast.

1. List list = **new** ArrayList();
2. list.add("hello");
3. String s = (String) list.get(0);//typecasting

After Generics, we don't need to typecast the object.

1. List<String> list = **new** ArrayList<String>();
2. list.add("hello");
3. String s = list.get(0);
4. How many issues you handle daily? How you will get the issues?
5. What you will do if you get a serious issue?
6. What is svn?

Apache Subversion (often abbreviated **SVN**, after its command name **svn**) is a software versioning and revision control system distributed as open source under the Apache License. Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.

1. Write a query for selecting sal column from emp table and the sal should be in ascending order?

Ans: SELECT \* FROM EMPLOYEE ORDERBY SALARY;

SELECT \* from Emp **order by** salary;

1. What is the difference between drop, truncate and delete?

The DROP command removes a table from the database. All the tables' rows, indexes and privileges will also be removed.

TRUNCATE removes **all rows** from a table

The DELETE command is used to remove rows from a table.

1. What are different forms of joins?

A SQL **join** clause combines columns from one or more tables in a relational database. It creates a set that can be saved as a table or used as it is. ... ANSI-standard SQL specifies five **types** of **JOIN** : INNER , LEFT OUTER , RIGHT OUTER , FULL OUTER and CROSS .

1. Write the code for reversing a string with out using reverse method?

Public class ReverseString{

Public static void main(String args[]){

String name =”uday”;

String reverse=””;

For(int i=name.length()-1;i>=0;i--){

reverse+=name.charAt(I);

}

System.out.println(reverse);

}

}

1. Write the code for palindrome?
2. **class** PalindromeExample{
3. **public** **static** **void** main(String args[]){
4. **int** r,sum=0,temp;
5. **int** n=454;//It is the number variable to be checked for palindrome
7. temp=n;
8. **while**(n>0){
9. r=n%10;  //getting remainder
10. sum=(sum\*10)+r;
11. n=n/10;
12. }
13. **if**(temp==sum)
14. System.out.println("palindrome number ");
15. **else**
16. System.out.println("not palindrome");
17. }
18. }

Output:

palindrome number

1. Where you configure the spring.xml? How you read xml file manually?
2. Where you configure the database properties?
3. How you handle the exceptions in jsp page?

 In JSP, there are two ways to perform exception handling:

1. By **errorPage** and **isErrorPage** attributes of page directive
2. By **<error-page>** element in web.xml file
3. Generally how you handle the exceptions in java?

By using try catch

1. Can we write only catch and finally with out try? If No, why?

No it is not possible to have try without catch block or finally block.  
  
You can write in 3 ways.  
  
Either try and catch(single or multiple)  
  
Either try and finally  
  
Either try , catch and finally in the same sequence.

1. What is session?

**Session** Tracking is a way to maintain state (data) of an user. It is also known as **session** management in servlet. Http protocol is a stateless so we need to maintain state using **session** tracking techniques. Each time user requests to the server, server treats the request as the new request.

1. Why string is immutable class? Can we extend string class? Why?
2. The string is Immutable in Java because String objects are cached in String pool. Since cached String literals are shared between multiple clients there is always a risk, where one client's action would affect all another client. For example, if one client changes the value of String "Test" to "TEST", all other clients will also see that value as explained in the first example. Since caching of String objects was important from performance reason this risk was avoided by making String class Immutable. At the same time, [*String was made final*](http://java67.blogspot.com/2014/01/why-string-class-has-made-immutable-or-final-java.html) so that no one can compromise invariant of String class e.g. Immutability, Caching, hashcode calculation etc by extending and overriding behaviors. Another reason of *why String class is immutable* could die due to HashMap.

Since Strings are very popular as HashMap key, it's important for them to be immutable so that they can retrieve the value object which was stored in HashMap. Since [HashMap works in the principle of hashing](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html), which requires same has value to function properly. Mutable String would produce two different hashcodes at the time of insertion and retrieval if contents of String was modified after insertion, potentially losing the value object in the map.

We cannot extend String class because it is final class

BirlaSoft Interview Questions

**All rounds(1,2,3)**

1. What is abstract class?where you have used in your project?what is the use?Can you give me some of the examples?

An **abstract class** is a **class** that is declared **abstract** —it may or may not include**abstract** methods. **Abstract classes** cannot be instantiated, but they can be subclassed. ... When an **abstract class** is subclassed, the subclass usually provides implementations for all of the **abstract** methods in its parent **class**.

1. How can you call parent class constructor?

By using Super keyword

1. **class** Animal{
2. Animal(){System.out.println("animal is created");}
3. }
4. **class** Dog **extends** Animal{
5. Dog(){
6. **super**();
7. System.out.println("dog is created");
8. }
9. }
10. **class** TestSuper3{
11. **public** **static** **void** main(String args[]){
12. Dog d=**new** Dog();
13. }}

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestSuper3)

Output:

animal is created

dog is created

#### Note: super() is added in each class constructor automatically by compiler if there is no super() or this().

1. Can we create object for abstract class?

No we can’t create object for abstract class

1. Can we declare a constructor in abstract class?if so what is the use?

Yes, an abstract class can have a constructor

To initialize the variables in that class.

1. Can you write the code for arraylist& iterate the list using iterator?

Question Repeated

1. Write the code for hashmap and iterate it
2. public class HashMapIteratorExample {
4. public static void main(String[] args) {
6. // Creating a HashMap of int keys and String values
7. HashMap<Integer, String> hashmap = new HashMap<Integer, String>();
9. // Adding Key and Value pairs to HashMap
10. hashmap.put(11,"Value1");
11. hashmap.put(22,"Value2");
12. hashmap.put(33,"Value3");
13. hashmap.put(44,"Value4");
14. hashmap.put(55,"Value5");
16. // Getting a Set of Key-value pairs
17. Set entrySet = hashmap.entrySet();
19. // Obtaining an iterator for the entry set
20. Iterator it = entrySet.iterator();
22. // Iterate through HashMap entries(Key-Value pairs)
23. System.out.println("HashMap Key-Value Pairs : ");
24. while(it.hasNext()){
25. Map.Entry me = (Map.Entry)it.next();
26. System.out.println("Key is: "+me.getKey() +
27. " & " +
28. " value is: "+me.getValue());
29. }
30. }
31. }
32. Output:
33. HashMap Key-Value Pairs :
34. Key is: 33 & value is: Value3
35. Key is: 55 & value is: Value5
36. Key is: 22 & value is: Value2
37. Key is: 11 & value is: Value1
38. Key is: 44 & value is: Value4
39. Write the logic for factorial
40. **class** FactorialExample{
41. **public** **static** **void** main(String args[]){
42. **int** i,fact=1;
43. **int** number=5;//It is the number to calculate factorial
44. **for**(i=1;i<=number;i++){
45. fact=fact\*i;
46. }
47. System.out.println("Factorial of "+number+" is: "+fact);
48. }
49. }

Output:

Factorial of 5 is: 120

1. If we are having a string like ‘baba” we need to count how many times each letter is repeated with out using forloop

|  |  |
| --- | --- |
| 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  36  37  38  39  40  41  42  43  44  45 | class EachCharCountInString  {      static void characterCount(String inputString)      {          //Creating a HashMap containing char as a key and occurrences as  a value            HashMap<Character, Integer> charCountMap = new HashMap<Character, Integer>();            //Converting given string to char array            char[] strArray = inputString.toCharArray();            //checking each char of strArray            for (char c : strArray)          {              if(charCountMap.containsKey(c))              {                  //If char is present in charCountMap, incrementing it's count by 1                    charCountMap.put(c, charCountMap.get(c)+1);              }              else              {                  //If char is not present in charCountMap,                  //putting this char to charCountMap with 1 as it's value                    charCountMap.put(c, 1);              }          }            //Printing the charCountMap            System.out.println(charCountMap);      }        public static void main(String[] args)      {         characterCount("Java J2EE Java JSP J2EE");           characterCount("All Is Well");           characterCount("Done And Gone");      }  } |

1. Write the logic for prime number
2. **class** PrimeExample{
3. **public** **static** **void** main(String args[]){
4. **int** i,m=0,flag=0;
5. **int** n=17;//it is the number to be checked
6. m=n/2;
7. **for**(i=2;i<=m;i++){
8. **if**(n%i==0){
9. System.out.println("Number is not prime");
10. flag=1;
11. **break**;
12. }
13. }
14. **if**(flag==0)
15. System.out.println("Number is prime");
16. }
17. }

Output:

Number is prime

1. What are stereotype annotations in hibernate?

No answer

1. Why we need to use @repository on top of DAO?if we replace it with other annotation what happens?what is the functionality @repository is having related to DAO?

Ans: we need to use @repository on top of DAO to inform that class is DAO Layer.

if we replace it with other annotation like @Component readability problem occurs.

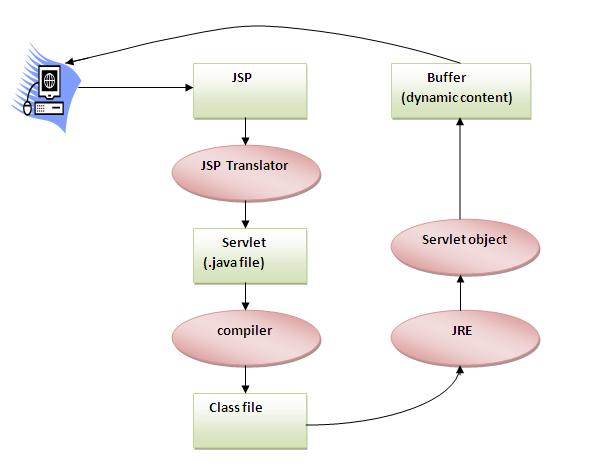
1. Jsp lifecycle?

### **Life cycle of a JSP Page**

The JSP pages follows these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (class file is loaded by the classloader)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( jspInit() method is invoked by the container).
* Reqeust processing ( \_jspService() method is invoked by the container).
* Destroy ( jspDestroy() method is invoked by the container).

#### Note: jspInit(), \_jspService() and jspDestroy() are the life cycle methods of JSP.



As depicted in the above diagram, JSP page is translated into servlet by the help of JSP translator. The JSP translator is a part of webserver that is responsible to translate the JSP page into servlet. Afterthat Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happens in servlet is performed on JSP later like initialization, committing response to the browser and destroy.

1. Why jspservice is having “\_”before the service method?

the \_ ( underscore) is used to specify that this method cannot be overriden. so the \_jspService() method cannot be overriden.

1. If we make a second call to the jsp what happens?

Service method will be executed twice.

1. Jsp implicit objects?
2. A list of the 9 implicit objects is given below:

|  |  |
| --- | --- |
| **Object** | **Type** |
| Out | JspWriter |
| Request | HttpServletRequest |
| Response | HttpServletResponse |
| Config | ServletConfig |
| Application | ServletContext |
| Session | HttpSession |
| pageContext | PageContext |
| Page | Object |
| Exception | Throwable |

1. Diff bw saveorupdate and merge methods?
2. update(), merge() & saveOrUpdate() all three methods are used to modify the DB table records. While working with these methods these all look quite similar but there are some differences between them.
3. Difference between all those three methods is as follows:

|  |  |  |
| --- | --- | --- |
| **public void update(Object obj)** | **public Object merge(Object obj)** | **public void saveOrUpdate(Object obj)** |
| Just updates the record & throws Hibernate Exception if record is not available to update. | Updates the record if record is available otherwise inserts the record in DB table. | Updates the record if record is available otherwise inserts the record in DB table. |
| This method can not update primary key column value. | Through this method we can update primary key column value also. | Through this method we can update primary key column value also. |
| It just involves only one object of POJO class while performing operation. | It involves two objects of POJO class while performing operation. | It involves only one POJO class object while performing operation. |

1. Is everytime the object will be creted by the spring container?

NO because every bean scope is singleton

1. What is singleton designpattern?where you will write this in applications?

In **Java** the **Singleton pattern** will ensure that there is only one instance of a class is created in the **Java**Virtual Machine. It is used to provide global point of access to the object. In terms of practical use**Singleton patterns** are used in logging, caches, thread pools, configuration settings, device driver objects.

1. What is Static factory method?

Ans:

**Factory Method**: "Define an interface for creating an object, but let the classes which implement the interface decide which class to instantiate. ... "**Static factory method** is simply a **static method** that returns an instance of a class." (c) Effective**Java**, Joshua Bloch. Usually this **method** is inside a particular class. A method which returns same class object is known as Static factory method

1. What is mockito?

**Mockito** is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. **Mockito** is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

1. How will you run junittestcases using maven?

BY using install command

1. Tell me about your project?
2. What kind of serious bug you have faced in your project and how did you handled it?
3. What are your roles in your project?
4. Have you involved in documentation of your project?
5. Where is your company located?
6. What is your current ctc?
7. Expected ctc?
8. What is your initial salary?how many times you got hikes?In which month?
9. Since how long you are on bench?
10. Why they didn’t give you permanent position?
11. Where is your payroll company located?
12. From when your current ctc is effecting?
13. Your team size?
14. Who is your team lead?
15. Mail id of your tl?
16. What are jsp implicit objects?why we need them?

|  |  |
| --- | --- |
| **Object** | **Type** |
| out | JspWriter |
| request | HttpServletRequest |
| response | HttpServletResponse |
| config | ServletConfig |
| application | ServletContext |
| session | HttpSession |
| pageContext | PageContext |
| Page | Object |
| exception | Throwable |

1. how can you maintain a session?

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**
5. Have you ever seen constructor chaining in your project?
6. What is the version control tool you are using?

Ans: SVN

1. Why we need svn?

Ans: To store our source code in certain repository.

1. **What are the servers you are using?**

We are using **Tomcat** server

1. **What is log4j?why we need log4j?how did you implemented in your project?where you will set the levels?what are the levels?**

**log4j:-**

* Log4j is aopen source logging framework / debugging tool.
* Log4j is from ASF(Apache Software Foundation)
* Log4j is written in java.

**log4j need:-**

* No project can exist without LOG4J.
* **Logging** is the act of recording the events occurred in execution of application.
* Logging occurred in the console. So that cannot be preserved for future reference.
* This logging won’t give where exactly event occurs, when occurs,under which thread occurs…etc.

But if we want all these info we should implement the logic on our own.

* If we don’t want the logging then we need to physically delete the code (System.out.println() statements),then we have to recompile the application and redeploy application is required.
* Performance compromise.
* **log4j**overcomes all the problems (above specified) of traditional logging.
* It also added some more advantages over the traditional logging.
* **log4j**is light weight framework.

**Log4j implemention in our project:- (explain anyone approach)**

**Properties file Approach**

* Write Properties file and specify appender, layout and level information.
* Name the properties file name as ‘log4j.properties’ and keep it in classes folder, Otherwise we need to read on our own.
* Create Logger
* Perform logging activites

**XML file approach**

* Write xml file and specify appender, layout and level information.
* Name the xml file name as ‘log4j.xml and keep it in classes folder(web project) Otherwise we need to read it on our own.
* Create Logger
* Perform logging activites.

**log4j.xml versus log4j.properties (optional)**

* Properties can be defined by a properties file or by an XML file. Log4j looks for a file namedlog4j.xml and then for a file named log4j.properties. Both must be placed in the src folder.The property file is more verbose than an XML file. The XML requires the log4j.dtd to be placed inthe source folder as well. The XML requires a dom4j.jar which might not be included in older Javaversions.
* The properties file does not support some advanced configuration options like Filters, customErrorHandlers and a special type of appenders, i.e. AsyncAppender.

**Log4j Major components:- (optional)**

The use of log4j revolves around 3 main components:

* **Logger** is responsible for handling the majority of logging operations.**(What to log)**
* **Appender** is responsible for controlling the output of logging operations. **(Where to log)**
* **Layout** is responsible for formatting the output.**(How to log)**

**Where you will set the levels?**

We set the levels either in property file or xml fileat root level.

**Log4j levels:-**

Level specifies priority of the logging message.

* **All**🡺All levels including custom levels
* **trace**🡺 (since log4j 1.2.12) developing only, can be used to follow the program execution.

(flow of execution)

* **Debug**🡺 developing only, for debugging purpose

Fine-grained informational events.(variable values)

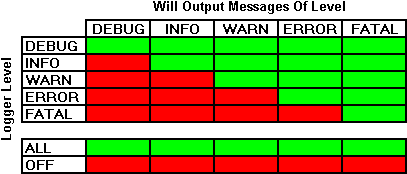
* **info**🡺 Production optionally, Coarse-grainedinformational events.

(configuration success, business method output, …etc.)

* **Warn**🡺 Production, simple application errors or unexpected behavior. Application cancontinue. For example in case of bad login attempts.
* **Error**🡺 Production, application error/exceptionbut application can continue. Part of the application is probably may not work.

For example file reading, file downloading…etc.

* **fatal**🡺 Production,
* application cannot continue, for example database is down, configuration file readingfailed(irrecoverable)
* **OFF**🡺 Do not log at all.



1. **Is your project is based on agile?**

Yes, my project is based on agile

1. **If so what is your sprint time?if you didn’t complete the task in assigned sprint what happens?**

My Sprint time is2 weeks, if I didn’t complete the task in assigned Sprint then we plan for a Sprint and we make it as a short Sprint.

**For more reference:**

[http://www.agileadvice.com/2014/06/12/howtoapplyagile/21-tips-on-choosing-a-sprint-length/](Resume%20Preparation/Malik_Basha_Resume.doc)

1. **If you will join in birlasoft what are the changes and extra things you want to see after 1 year in your resume?**

After 1 year in my resume, I will add new technologies and new skills what I learn from **birlasoft** which are increase my efficiency as a developer.

1. **Tell me about your family background?**

About my family; my father is a contractor, my mother is a housewife and I have one more brother, he is in B.Sc 2nd year.

1. **How much percentage you got in your 10th,inter,btech?**

In 10th I got 90%, in Inter I got 80%, in btech I got 70% (in mtech I got 75%)

1. **Have you taken any training outside before joining your first company?**

Yes, I have taken C, C++ and Java training outside before joining my first company

1. **Why didn’t u asked your company to make you as a permanent employee?**

I am already apermanent employee, there is no need to ask (we don’t have project so present I am on bench)

1. **Scopes in spring?**

**Singleton:**

This scopes the bean definition to a single instance per Spring IoC container (default).

**Prototype:**

This scopes a single bean definition to have any number of object instances.

**Request:**

This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.

**Session:**

This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

**global-session:**

This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

1. **What is springIOC?what is the use of it?**

|  |  |
| --- | --- |
| **Inversion Of Control** is a design principle, which explains that an object creation, dependency injection, object lifecycle management, object destruction… etc. has to managed by an external entity.  In Spring, Spring Container is the external entity, which implements principles of IOC.  **Use of IOC:**  The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. | Image result |

1. **Do you have any experience with webservices?**

Yes, I have experience with webservices.

**UST GLOBAL SERVICES**

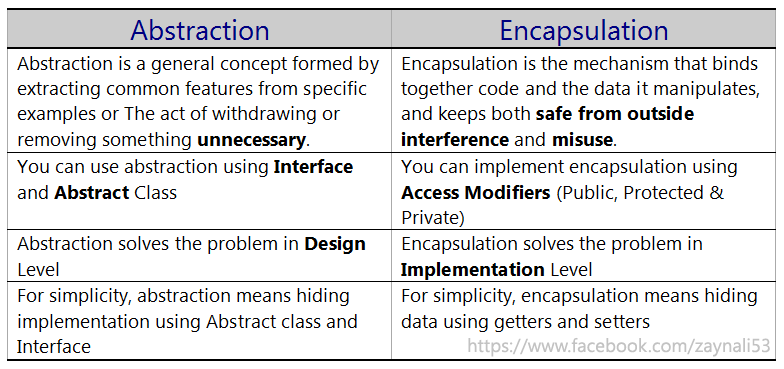
**CORE JAVA**

1. **What are the OOPS principles and explain them briefly?**

**OOPS Principles:**

|  |  |
| --- | --- |
| **Abstraction:**   * **Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing. * In java, we use abstract class and interface to achieve abstraction.   **Inheritance:**   * **When one object acquires all the properties and behaviours of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism. | Image result |
| **Polymorphism:**   * When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc. * In java, we use method overloading and method overriding to achieve polymorphism. * Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc. | **Encapsulation:**   * Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines. * A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here. |

1. **What is the difference between abstraction and encapsulation, where you used these concepts in your project?**



|  |  |
| --- | --- |
| **Abstraction:**  In project, we use abstract class and interface to achieve abstraction. | **Encapsulation:**  In project, Java bean is the fully encapsulated class because all the data members are private here. |

1. **How you applied encapsulation in your project? Example**

|  |  |
| --- | --- |
| **Encapsulation:**  I applied encapsulation in my project using getter and setter methods. To hide the implementation details from users. If a data member is **private** it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class. However if we setup public **getter and setter** methods to update and read the private data fields then the outside class can access those private data fields via public methods. | https://qph.ec.quoracdn.net/main-qimg-3d4963eaf4de2261b6007ef7ef5dad98 |

This way data can only be accessed by public methods thus making the private fields and their implementation hidden for outside classes.

**Example:**In our project, Java bean (refer our pojo, model classes) is the fully encapsulated class because all the data members are private here.

1. **What is static and final keyword? Can we override?**

|  |  |
| --- | --- |
| **Static Keyword:**  Static keyword is used to do better memory management.   * You can use static keyword with variable, method, class. * If you declare method using static keyword then you don't need to create an object to call that method. * If you declare variable as a static then only once memory is created for that variable in class area and not every time when object is created. e.g. in student database you can declare "collage name" as because it common for all students. * We can not **override** the static method because static metod is a class method and the scope of this method within the same class itself. ... Hence, it does not makes sense to "override" it (or for that matter any static method). The concept of "overriding" is only for instance methods | **Final Keyword:**  Final Keyword is used to restrict the user. **It means:**   * If you declare variable as static then you cannot change its value. * If you declare method as static then you cannot override that method. * If you make any class as final, you cannot extend it. * Final cannot be**overridden** because that is the purpose of the keyword, something that cannot be changed or overridden. So there is no purpose tooverridingthe Static methods. And you can have a static method in subclass by the same name but that won't be anoverriddenmethod. |

1. **What is static polymorphism and dynamic polymorphism?**

|  |  |
| --- | --- |
| **Static (compile time) polymorphism:**  It is the polymorphism existed at run-time. Here, Java compiler does not understand which method is called at compilation time. Only JVM decides which method is called at run-time. Method overloading and method overriding using instance methods are the examples for dynamic polymorphism. | **Dynamic (run time) polymorphism:**  It is the polymorphism exhibited at compile time. Here, Java compiler knows which method is called. Method overloading and method overriding using static methods; method overriding using private or final methods are examples for static polymorphism |

1. **Explain about collection framework, which collection interface you used in your project and where?**

Collection framework represents a unified architecture for storing and manipulating group of objects. It has:

* + - Interfaces and its implementations i.e. classes
    - Algorithm

Let us see the hierarchy of collection framework.The **java.util** package contains all the classes and interfaces for Collection framework.

hierarchy of collection framework

List, Set and Map are the collection interfaces, we used in our project.

List🡪 To show the table we are using list interface

Set🡪 To maintain the table in sorting order according to the requirement

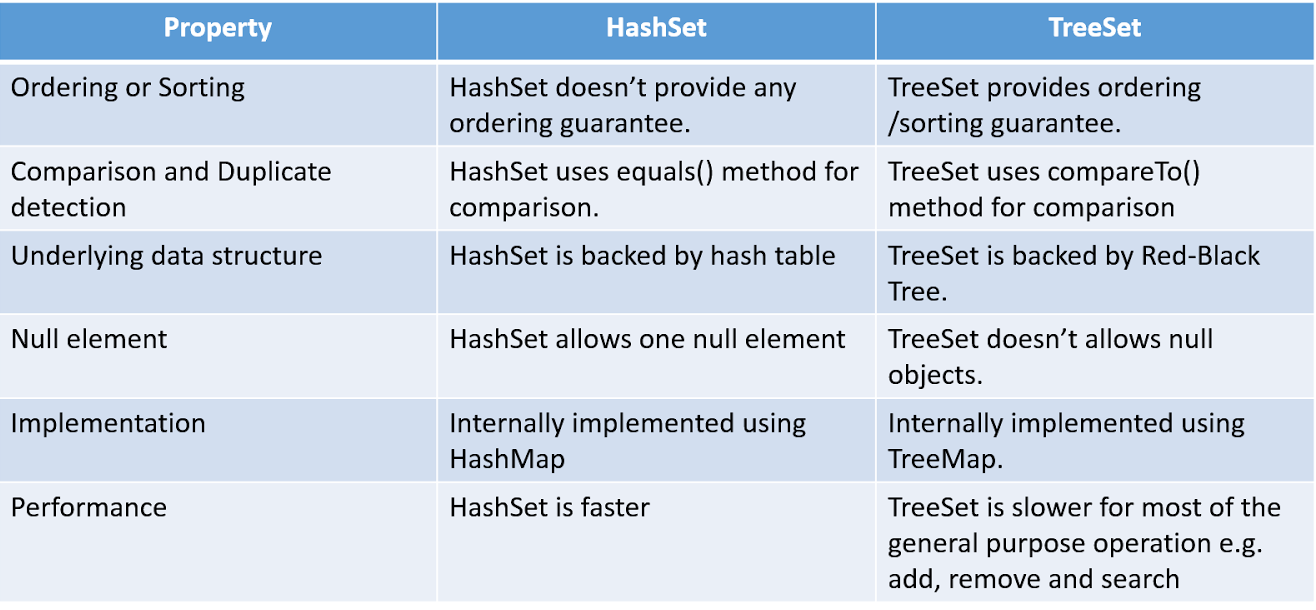
Map🡪 To show the dropdown list (**note**: It is not a collection framework)

1. **What is comparable and comparator, how to use this and where you used in your project?**

|  |  |
| --- | --- |
| Comparator interface:  * Java Comparator interface is used to order the objects of user-defined class. * This interface is found in java.util package and contains 2 methods compare(Object obj1,Object obj2) and equals(Object element). * It provides multiple sorting sequence i.e. you can sort the elements on the basis of any data member, for example rollno, name, age or anything else. | Comparable interface:  * Java Comparable interface is used to order the objects of user-defined class. * This interface is found in java.lang package and contains only one method named compareTo(Object). * It provide single sorting sequence only i.e. you can sort the elements on based on single data member only. For example it may be rollno, name, age or anything else. |

In our project, to maintain the table in sorting order we used the comparator and comparable

1. **What is the difference between HashSet and TreeSet?**



1. **Will you pass objects or datatypes to the TreeSet in your project?**

Yes, I passed the same datatype objects to the TreeSet in my project (note: TreeSet is a homogeneous It means, It don’t allow different datatypes)

1. **What is serialization?**

|  |  |
| --- | --- |
| **Serialization** is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called **deserialization**. | Image result |

**SPRING:**

1. **What is the use of spring framework?**

**1) Predefined Templates**

* + Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.
  + Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

**2) Loose Coupling**

* + The Spring applications are loosely coupled because of dependency injection.

**3) Easy to test**

* + The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

**4) Lightweight**

* + Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

**5) Fast Development**

* + The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

**6) Powerful abstraction**

* + It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

**7) Declarative support**

* + It provides declarative support for caching, validation, transactions and formatting.

1. **What is the dependency injection and types of dependency injections, which dependency injections used in your projects?**

**Dependency Injection:** The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing

Spring framework provides two ways to inject dependency

* + By Constructor
  + By Setter method

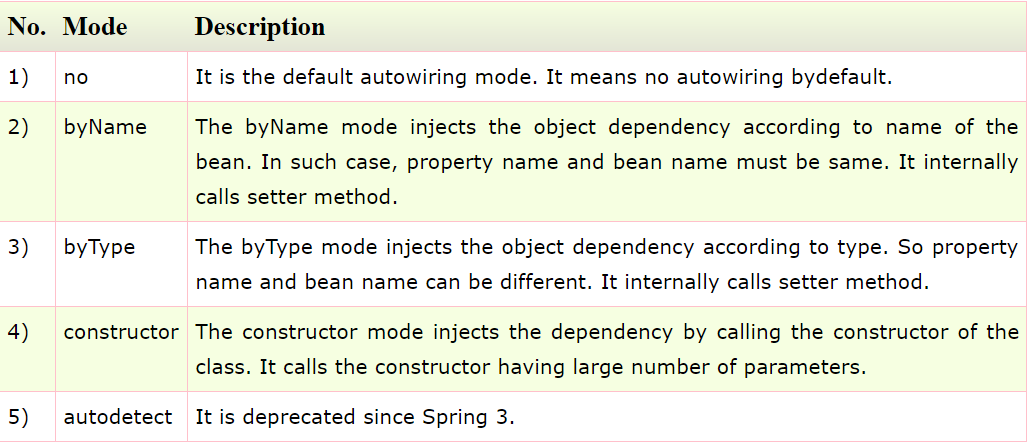
In our project, we are using annotations@Autowired.It internally calls setter method

1. **What is autowiring and types of autowiring? Explain briefly the use of autowiring.**

**Autowiring**:

* The Spring container can **autowire** relationships between collaborating beans **without** using **<constructor-arg>** and **<property>** elements which helps cut down on the amount of XML configuration you write for a big Spring based application.
* **Auto wiring** means automatically injecting the dependencies.
* Instead of manually configuring the injection we done it automatically by using auto wiring
* To implement autowiring we use **autowire** attribute of **<bean>** tag.

**Types of autowiring:**



**Use of autowiring**: It requires the **less code** because we don't need to write the code to inject the dependency explicitly.

1. **What is the default autowiring?**

**no Mode,**It is the default autowiring mode. It means no autowiring bydefault.

(**no**: It won’t allow autowiring.)

1. **What are the bean scopes? Explain all scopes.**

When defining a <bean> you have the option of declaring a scope for that bean. For example, to force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be **prototype**. Similarly, if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be **singleton**.

The Spring Framework supports the following five scopes, three of which are available only if you use a web-aware ApplicationContext

|  |  |
| --- | --- |
| 1. **singleton:**   Specifies to container to create the configured spring bean only once in the container.  https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/images/singleton.png | 1. **prototype:**   Specifies the container to create the configured spring bean for each time when it is requested (getBean(“springref”)).  https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/images/prototype.png |

1. **request**

Specifies the container to create the configured spring bean once per web request. It is applicable only in web module.

1. **session**

Specifies the container to create the configured spring bean once per HttpSession. It is applicable only web module.

1. **globalsession**

Specifies the container to crate the configured spring bean once in a global session. It is applicable in portals.

1. **thread**

Specifies the container to create the configured spring bean once per each thread. It is not implicitly registered. If we want to specify thread as a scope for bean, we have to register explicitly.

1. **What is the default bean scope in spring?**

**Singleton** is the **default** bean scope in spring

1. **Difference between session and prototype scope?**

|  |  |
| --- | --- |
| **Session scope:**  Returns a single instance for every HTTP session.It isonly available on Web aware application Context like WebApplicationContext. | **Prototype scope:**  Returns a new bean instance each time when requested.which is available on all Spring Application Context. |

1. **Bean lifecycle? Explain briefly?**

Spring container controls the life cycle of spring bean i.e. from instantiation to destruction.

**Spring bean has the following life cycle states**:

* **Instantiation** 
  + Creation of the object
  + In spring objects are created by spring container
* **Injecting dependencies** 
  + This is Dependency Injection
  + If any dependencies are configured to spring bean (either constructor or setter approach) those will be injected by the spring container.
* **Initialization**
  + When a bean is instantiated, it may be required to perform some initialization to get it into a usable state. If any initialization methods are configured (either with interfaces or with the configuration or with the annotations) those will be called by the spring container
* **Method Ready state**
  + In this stated Object ready to use, means we can business methods. Because in the previous states object was created, happensdependencies are injected, initialization, so now the bean ready serve.
* **Destruction**
  + When the bean is no longer from the required and is removed container, some cleanup may be required. If any destruction methods are configured (either with interfaces or with the configuration or with the annotations) those will be called by the spring container, when the container is shutdown.

|  |  |
| --- | --- |
| Initialization callbacks: | <bean id = "exampleBean" class = "examples.ExampleBean" init-method = "init"/> |
| Destruction callbacks: | <bean id = "exampleBean" class = "examples.ExampleBean" destroy-method = "destroy"/> |

1. **What is constructor injection and setter injections?**

|  |  |
| --- | --- |
| **Setter Injection** | **Constructor Injection** |
| **1.** In Setter Injection, partial injection of dependencies can possible, means if we have 3 dependencies like int, string, long, then its not necessary to inject all values if we use setter injection. If you are not inject it will takes default values for those primitives | **1.** In constructor injection, partial injection of dependencies cannot possible, because for calling constructor we must pass all the arguments right, if not so we may get error |
| **2.** Setter Injection will overrides the constructor injection value, provided if we write setter and constructor injection for the same property [i already told regarding this, hope you remember ] | **2.** But, constructor injection cannot overrides the setter injected values |
| **3.** If we have more dependencies for example 15 to 20 are there in our bean class then, in this case setter injection is not recommended as we need to write almost 20 setters right, bean length will increase. | **3.** In this case, Constructor injection is highly recommended, as we can inject all the dependencies within 3 to 4 lines [i mean, by calling one constructor] |
| **4.** Setter injection makes bean class object as mutable [We can change ] | **4.** Constructor injection makes bean class object as immutable [We cannot change ] |

1. **In which situation you use singleton and prototype?**
   * **Singleton scope:**use the singleton scope for stateless beans.
   * **Prototype scope:**use the prototype scope for all state-full beans.
2. **Difference between prototype and request scopes?**

|  |  |
| --- | --- |
| **Requestscope:**  Returns a single instance for every HTTP request.It isonly available on Web aware application Context like WebApplicationContext. | **Prototype scope:**  Returns a new bean instance each time when requested.which is available on all Spring Application Context. |

1. **What are the helper classes in spring mvc?**

**Service Class :** Contain Business logic.   
**Helper Class :** this class is one type of reusable component.

The helper classes in spring mvc are, **WebMvcConfigurationSupport**, **ApplicationContextProvider, etc.,**

**HIBERNATE:**

1. **What are the different levels of caching? Explain?**

**Hibernatecaching** improves the performance of the application by pooling the object in the cache.

There are mainly two types of caching: first level cache and second level cache.

**FirstLevel Cache:**

Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to entire application. An application can use many session object.

**Second Level Cache**

SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.

1. **What is the default caching mechanism?**

**Hibernate** uses first-level **cache** by **default** and you have nothing to do to use first-level cache. Let's go straight to the optional second-level **cache.**

1. **What are the relationship mappings present in hibernate?**

The relationship mappings present in hibernate are:

* + One-To-One
  + Many-To-One**(f.k—>p.k)**
  + Many-To-Many
  + One-To-Many**(p.k-->f.k)**

1. **How you used one-to-many relationship in your project?**

|  |  |
| --- | --- |
| A **one-to-many(p.k-->f.k)** relationship occurs when one entity is related to many occurrences in another entity.  (**E**x:In our project, to represent EMP to DEPT relationship,we used **one-to-many** relationship.) |  |

1. **How you implemented the one-to-many relationship in your project?**

One-to-Many relationship can be developed in four ways.

* Set
* List
* Bag
* Array

I implemented the one-to-many relationship in your project, Using annotations:

Ex:-

**@OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)**

**@JoinColumn(name = "E\_DNO", referencedColumnName = "DNO")**

**private Set<Employee> employeeSet;**

**WEBSERVICES:**

1. **What is Webservices and why we use Webservices?**

|  |  |
| --- | --- |
| **Webservices:**   * Supports Enterprise services * Interoperable (Language independent & Plat from independent)   So, Web services is interoperable distributed technology  **(OR)**  **Web services** (sometimes called application **services**) are **services** (usually including some combination of programming and data, but possibly including human resources as well) that are made available from a business's **Web** server for **Web**users or other **Web**-connected programs. | web services |

**Use of Web services:**

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents. A web service is a collection of **open** protocols and standards used for exchanging data between applications or systems.

1. What are the types of classifications?

|  |  |
| --- | --- |
| There are mainly two types of web services.   1. SOAP web services. 2. RESTful web services. | types of web services |

**CAPGEMINI**

**COREJAVA:**

|  |  |
| --- | --- |
| Class Parent  {  Public void m1(int i)  {  Syso(“parent”);  }  } | Class Child extands Parent  {  Public void m1(int i)  {  Syso(“child”);  }  **main()**  **{**  **Parent p=new Child();**  **p.m1(5);**  **}**  } |

1. **Which method will be called parent method or child method?**

Parent m1 method will be called

1. **Which tag is this <%= ?**

<%=  statement %> , this is [**JSPExpression** Tag](http://www.javatpoint.com/jsp-expression-tag)

1. **If a servlet class implements singlethreadmodel interface how many threads will be created?**

Multiple threads will be created.

1. **Can we write doGet() method in doPost() method?**

Yes, we can write doGet() method in doPost() method

1. **Who provided implementation of AOP annotations?**

Implementation of AOP annotations provided by **Spring AspectJ**

1. **Which of the below annotation are wrong with respect to AOP?**
2. **@Before b) @After c) @Throw d) all the above.**

Option, c) @Throw is wrong with respect to AOP

1. **What is the use of @Pointcut?**

|  |  |
| --- | --- |
| **Advice**: This is an object which includes API invocations to the system wide concerns representing the action to perform at a joinpoint specified by a point.  **Pointcut**: A **pointcut** defines at what joinpoints, the associated Advice should be applied. Advice can be applied at any joinpoint supported by the AOP framework. | Image result |

1. **What is Serialization?**

|  |  |
| --- | --- |
| **Serialization** is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called **deserialization**. | Image result |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **What it is?** | **Who creates it?** | **When it will create?** | **How many times it created?** | **What it contains?** | **When it is destroyed?** |
| Servlet | Interface | Servlet  Container | At the time of first  Request(or)At application deploying using Load on Startup method | One | Developers logic  init()  service()  destroy() | 1.Application undeployed  2.Servlet undeployed  3.server shutdown |
| HttpServletRequest | Interface | Servlet  Container | Before calling service method | One per  request | User input  Request header  cookies | After generating response |
| HttpServletResponse | Interface | Servlet  Container | Before calling service method | One per  Response | Response header  cookies | After generating response |
| ServletConfig | Interface | Servlet  Container | Before calling init method | One per servlet | Servlet configuration information | Until servlet is alive |
| ServletContext | Interface | Servlet  Container | At the time of deploying the application | One per application | Entire application information | When application undeployed,server shutdown |
| HttpSession | Interface | Servlet  Container | After calling request.getSession() method | One per browser | Session information like cookies headers,attributesetc | When browser close,when user logout |
| RequestDispatcher | Interface | Servlet  Container | After calling request.getRequestDispatcher() method | Depends on requirement | Request,responseobjects,path of other resources | After executing service method |
| Filters | Interface | Servlet  Container | Before sending request to server and sending response to client | Depends on developer’s requirement(one or more per servlet) | Developer’s logic doFilter() | After the completion of doFilter() method |