**Core Java**

**Q #1) What is JAVA?**

**Answer:** Java is a high-level programming language and is platform-independent.

Java is a collection of objects. It was developed by Sun Microsystems. There are a lot of applications, websites, and games that are developed using Java.

**Q #2) What are the features of JAVA?**

**Answer: Features of Java are as follows:**

* **OOP concepts**
  + Object-oriented
  + Inheritance
  + Encapsulation
  + Polymorphism
  + Abstraction
* **Platform independent:** A single program works on different platforms without any modification.
* **High Performance:** JIT (Just In Time compiler) enables high performance in Java. JIT converts the bytecode into machine language and then JVM starts the execution.
* **Multi-threaded:** A flow of execution is known as a Thread. JVM creates a thread which is called the main thread. The user can create multiple threads by extending the thread class or by implementing the Runnable interface.

**Q #3) How does Java enable high performance?**

**Answer:** Java uses Just In Time compiler to enable high performance. It is used to convert the instructions into bytecodes.

**Q #4) Name the Java IDE’s?**

**Answer:** Eclipse and NetBeans are the IDE’s of JAVA.

**Q #5) What do you mean by Constructor?**

**Answer: Constructor can be explained in detail with enlisted points:**

* When a new object is created in a program a constructor gets invoked corresponding to the class.
* The constructor is a method which has the same name as the class name.
* If a user doesn’t create a constructor implicitly a default constructor will be created.
* The constructor can be overloaded.
* If the user created a constructor with a parameter then he should create another constructor explicitly without a parameter.c

**Q #6) What is meant by the Local variable and the Instance variable?**

**Answer:**

**Local variables** are defined in the method and scope of the variables that exist inside the method itself.

**Instance variable** is defined inside the class and outside the method and the scope of the variables exists throughout the class.

**Q #7) What is a Class?**

**Answer:** All Java codes are defined in a Class. It has variables and methods.

**Variables**are attributes which define the state of a class.

**Methods** are the place where the exact business logic has to be done. It contains a set of statements (or) instructions to satisfy the particular requirement.

**Example:**

|  |
| --- |
| public class Addition{ //Class name declaration  int a = 5; //Variable declaration  int b= 5;  public void add(){ //Method declaration  int c = a+b;  }  } |

**Q #8) What is an Object?**

**Answer:** An instance of a class is called an object. The object has state and behavior.

Whenever the JVM reads the “new()” keyword then it will create an instance of that class.

**Example:**

|  |
| --- |
| public class Addition{  public static void main(String[] args){  Addion add = new Addition();//Object creation  }  } |

The above code creates the object for the Addition class.

**Q #9)What are the OOPs concepts?**

**Answer: OOPs concepts include:**

* Inheritance
* Encapsulation
* Polymorphism
* Abstraction
* Interface

**Q #10) What is Inheritance?**

**Answer:** Inheritance means one class can extend to another class. So that the codes can be reused from one class to another class. The existing class is known as the Super class whereas the derived class is known as a sub class.

**Example:**

|  |
| --- |
| Super class:  public class Manupulation(){  }  Sub class:  public class Addition extends Manipulation(){  } |

Inheritance is only applicable to the public and protected members only. Private members can’t be inherited.

**Q #11) What is Encapsulation?**

**Answer: Purpose of Encapsulation:**

* Protects the code from others.
* Code maintainability.

**Example:**

We are declaring ‘a’ as an integer variable and it should not be negative.

|  |
| --- |
| public class Addition(){  int a=5;  } |

If someone changes the exact variable as “***a = -5”***then it is bad.

**In order to overcome the problem we need to follow the steps below:**

* We can make the variable private or protected.
* Use public accessor methods such as set<property> and get<property>.

**So that the above code can be modified as:**

|  |
| --- |
| public class Addition(){  private int a = 5; //Here the variable is marked as private  } |

**The code below shows the getter and setter.**

Conditions can be provided while setting the variable.

|  |
| --- |
| get A(){  }  set A(int a){  if(a&gt;0){// Here condition is applied  .........  }  } |

For encapsulation, we need to make all the instance variables private and create setter and getter for those variables. Which in turn will force others to call the setters rather than access the data directly.

**Q #12) What is Polymorphism?**

**Answer:** Polymorphism means many forms.

A single object can refer to the super-class or sub-class depending on the reference type which is called polymorphism.

**Example:**

|  |
| --- |
| Public class Manipulation(){ //Super class  public void add(){  }  }  public class Addition extends Manipulation(){ // Sub class  public void add(){  }  public static void main(String args[]){  Manipulation addition = new Addition();//Manipulation is reference type and Addition is reference type  addition.add();  }  } |

Using the Manipulation reference type we can call the Addition class “add()” method. This ability is known as Polymorphism. Polymorphism is applicable for **overriding**and not for **overloading**.

**Q #13) What is meant by Method Overriding?**

**Answer: Method overriding happens if the sub-class method satisfies the below conditions with the Super-class method:**

* Method name should be the same
* The argument should be the same
* Return type should also be the same

The key benefit of overriding is that the Sub-class can provide some specific information about that sub-class type than the super-class.

**Example:**

|  |
| --- |
| public class Manipulation{ //Super class  public void add(){  ………………  }  }    Public class Addition extends Manipulation(){  Public void add(){  ………..  }  Public static void main(String args[]){  Manipulation addition = new Addition(); //Polimorphism is applied  addition.add(); // It calls the Sub class add() method  }  } |

**addition.add()**method calls the add() method in the Sub-class and not the parent class. So it overrides the Super-class method and is known as Method Overriding.

**Q #14) What is meant by Overloading?**

**Answer:** Method overloading happens for different classes or within the same class.

**For method overloading, sub-class method should satisfy the below conditions with the Super-class method (or) methods in the same class itself:**

* Same method name
* Different argument types
* There may be different return types

**Example:**

|  |
| --- |
| public class Manipulation{ //Super class  public void add(String name){ //String parameter  ………………  }  }    Public class Addition extends Manipulation(){  Public void add(){//No Parameter  ………..  }  Public void add(int a){ //integer parameter    }  Public static void main(String args[]){  Addition addition = new Addition();  addition.add();  }  } |

Here the add() method has different parameters in the Addition class is overloaded in the same class as with the super-class.

**Note:** Polymorphism is not applicable for method overloading.

**Q #15) What is meant by Interface?**

**Answer:** Multiple inheritances cannot be achieved in java. To overcome this problem the Interface concept is introduced.

An interface is a template which has only method declarations and not the method implementation.

**Example:**

|  |
| --- |
| Public abstract interface IManupulation{ //Interface declaration  Public abstract void add();//method declaration  public abstract void subtract();  } |

* All the methods in the interface are internally **public abstract void**.
* All the variables in the interface are internally **public static final** that is constants.
* Classes can implement the interface and not extends.
* The class which implements the interface should provide an implementation for all the methods declared in the interface.

|  |
| --- |
| public class Manupulation implements IManupulation{ //Manupulation class uses the interface  Public void add(){  ……………  }  Public void subtract(){  …………….  }  } |

**Q #16) What is meant by Abstract class?**

**Answer:** We can create the Abstract class by using the “Abstract” keyword before the class name. An abstract class can have both “Abstract” methods and “Non-abstract” methods that are a concrete class.

**Abstract method:**

The method which has only the declaration and not the implementation is called the abstract method and it has the keyword called “abstract”. Declarations ends with a semicolon.

**Example:**

|  |
| --- |
| public abstract class Manupulation{  public abstract void add();//Abstract method declaration  Public void subtract(){  }  } |

* An abstract class may have a non- abstract method also.
* The concrete Subclass which extends the Abstract class should provide the implementation for abstract methods.

**Q #17) Difference between Array and Array List.**

**Answer:** **The Difference between Array and Array List can be understood from the table below:**

| **Array** | **Array List** |
| --- | --- |
| Size should be given at the time of array declaration.  String[] name = new String[2] | Size may not be required. It changes the size dynamically.  ArrayList name = new ArrayList |
| To put an object into array we need to specify the index.  name[1] = “book” | No index required.  name.add(“book”) |
| Array is not type parameterized | ArrayList in java 5.0 are parameterized.  Eg: This angle bracket is a type parameter which means a list of String. |

**Q #18) Difference between String, String Builder, and String Buffer.**

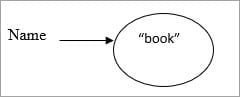
**Answer:**

**String:** String variables are stored in a “constant string pool”. Once the string reference changes the old value that exists in the “constant string pool”, it cannot be erased.

**Example:**

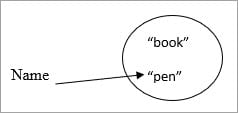
String name = “book”;

**Constant string pool**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Constant-string-pool.jpg).

If the name-value has changed from “book” to “pen”.

**Constant string pool**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Constant-string-pools.jpg)

Then the older value remains in the constant string pool.

**String Buffer:**

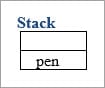
* Here string values are stored in a stack. If the values are changed then the new value replaces the older value.
* The string buffer is synchronized which is thread-safe.
* Performance is slower than the String Builder.

**Example:**

String Buffer name =”book”;

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Stack.jpg)

Once the name value has been changed to “pen” then the “book” is erased in the stack.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Stack1.jpg)

**String Builder:**

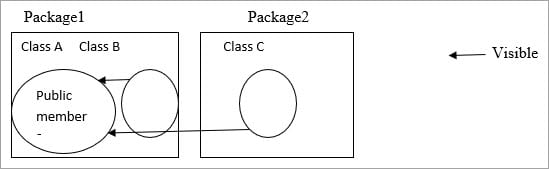
This is the same as String Buffer except for the String Builder which is not threaded safely that is not synchronized. So obviously the performance is fast.

**Q #19) Explain about Public and Private access specifiers.**

**Answer:** Methods and instance variables are known as members.

**Public:**

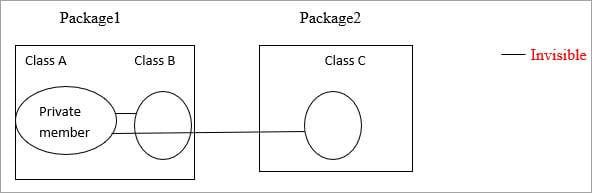
Public members are visible in the same package as well as the outside package that is for other packages.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Public.jpg)

Public members of Class A are visible to Class B (same package) as well as Class C (different packages).

**Private:**

Private members are visible in the same class only and not for the other classes in the same package as well as classes in the outside packages.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Private.jpg)

Private members in class A are visible only in that class. It is invisible for class  B as well as class C.

**Q #20) Difference between Default and Protected access specifiers.**

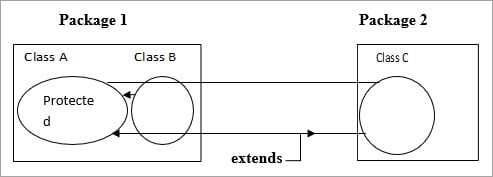
**Answer:**

**Default:**Methods and variables declared in a class without any access specifiers are called default.

Default members in Class A are visible to the other classes which are inside the package and invisible to the classes which are outside the package.

So Class A members are visible to Class B and invisible to Class C.

**Protected:**

**[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Protected.jpg)             .**

Protected is the same as Default but if a class extends then it is visible even if it is outside the package.

Class A members are visible to Class B because it is inside the package. For Class C it is invisible but if Class C extends Class A then the members are visible to Class C even if it is outside the package.

**Q #21) Difference between HashMap and HashTable.**

**Answer:** **The difference between HashMap and HashTable can be seen below:**

| **HashMap** | **HashTable** |
| --- | --- |
| Methods are not synchronized | Key methods are synchronized |
| Not thread safety | Thread safety |
| Iterator is used to iterate the values | Enumerator is used to iterate the values |
| Allows one null key and multiple null values | Doesn’t allow anything that is null |
| Performance is high than HashTable | Performance is slow |

**Q #22) Difference between HashSet and TreeSet.**

**Answer:** **The difference between HashSet and TreeSet can be seen below:**

| **HashSet** | **TreeSet** |
| --- | --- |
| Inserted elements are in random order | Maintains the elements in the sorted order |
| Can able to store null objects | Couldn’t store null objects |
| Performance is fast | Performance is slow |

**Q #23) Difference between Abstract class and Interface.**

**Answer: The differences between Abstract Class and Interface are as follows:**

**Abstract Class:**

* Abstract classes have a default constructor and it is called whenever the concrete subclass is instantiated.
* It contains Abstract methods as well as Non-Abstract methods.
* The class which extends the Abstract class shouldn’t require the implementation of all the methods, only Abstract methods need to be implemented in the concrete sub-class.
* Abstract class contains instance variables.

**Interface:**

* It doesn’t have any constructor and couldn’t be instantiated.
* The abstract method alone should be declared.
* Classes that implement the interface should provide the implementation for all the methods.
* The interface contains only constants.

**Q** **#24)  What is the meaning of Collections in Java?**

**Answer:** Collection is a framework that is designed to store the objects and manipulate the design to store the objects.

**Collections are used to perform the following operations:**

* Searching
* Sorting
* Manipulation
* Insertion
* Deletion

A group of objects is known as collections. All the classes and interfaces for collecting are available in Java util package.

**Q #25) What are all the Classes and Interfaces that are available in the collections?**

**Answer:** **Given below are the Classes and Interfaces that are available in Collections:**

**Interfaces:**

* Collection
* List
* Set
* Map
* Sorted Set
* Sorted Map
* Queue

**Classes:**

* Lists:
* Array List
* Vector
* Linked List

**Sets:**

* Hash set
* Linked Hash Set
* Tree Set

**Maps:**

* Hash Map
* Hash Table
* TreeMap
* Linked Hashed Map

**Queue:**

* Priority Queue

**Q #26) What is meant by Ordered and Sorted in collections?**

**Answer:**

**Ordered:**It means the values that are stored in a collection is based on the values that are added to the collection. So we can iterate the values from the collection in a specific order.

**Sorted:**Sorting mechanisms can be applied internally or externally so that the group of objects sorted in a particular collection is based on the properties of the objects.

**Q #27) Explain the different lists available in the collection.**

**Answer:**Values added to the list are based on the index position and it is ordered by index position. Duplicates are allowed.

**The types of Lists are:**

**a) Array List:**

* Fast iteration and fast Random Access.
* It is an ordered collection (by index) and not sorted.
* It implements the Random Access Interface.

**Example:**

|  |
| --- |
| public class Fruits{  public static void main (String [ ] args){  ArrayList &lt;String&gt;names=new ArrayList &lt;String&gt;();  names.add (“apple”);  names.add (“cherry”);  names.add (“kiwi”);  names.add (“banana”);  names.add (“cherry”);  System.out.println (names);  }  } |

**Output:**

[Apple, cherry, kiwi, banana, cherry]

From the output, Array List maintains the insertion order and it accepts the duplicates. But it’s not sorted.

**b) Vector:**

It is the same as Array List.

* Vector methods are synchronized.
* Thread safety.
* It also implements Random Access.
* Thread safety usually causes a performance hit.

**Example:**

|  |
| --- |
| public class Fruit {  public static void main (String [ ] args){  Vector &lt;String&gt; names = new Vector &lt;String&gt; ( );   names.add (“cherry”);  names.add (“apple”);  names.add (“banana”);  names.add (“kiwi”);  names.add (“apple”);  System.out.println (“names”);  }  } |

**Output:**

[cherry,apple,banana,kiwi,apple]

Vector also maintains the insertion order and accepts the duplicates.

**c) Linked List:**

* Elements are doubly linked to one another.
* Performance is slower than the Array list.
* Good choice for insertion and deletion.
* In Java 5.0 it supports common queue methods peek( ), Pool ( ), Offer ( ) etc.

**Example:**

|  |
| --- |
| public class Fruit {  public static void main (String [ ] args){  Linkedlist &lt;String&gt; names = new linkedlist &lt;String&gt; ( ) ;  names.add(“banana”);  names.add(“cherry”);  names.add(“apple”);  names.add(“kiwi”);  names.add(“banana”);  System.out.println (names);  }  } |

**Output:**

[ banana,cherry,apple,kiwi,banana]

Maintains the insertion order and accepts the duplicates.

**Q #28) Explain about Set and their types in a collection.**

**Answer:** Set cares about uniqueness. It doesn’t allow duplications. Here “equals ( )” method is used to determine whether two objects are identical or not.

**a) Hash Set:**

* Unordered and unsorted.
* Uses the hash code of the object to insert the values.
* Use this when the requirement is “no duplicates and don’t care about the order”.

**Example:**

|  |
| --- |
| public class Fruit {  public static void main (String[ ] args){  HashSet&lt;String&gt; names = new HashSet &lt;=String&gt;( ) ;  names.add(“banana”);  names.add(“cherry”);  names.add(“apple”);  names.add(“kiwi”);  names.add(“banana”);  System.out.println (names);  }  } |

**Output:**

[banana, cherry, kiwi, apple]

It doesn’t follow any insertion order. Duplicates are not allowed.

**b) Linked Hash set:**

* An ordered version of the hash set is known as Linked Hash Set.
* Maintains a doubly-Linked list of all the elements.
* Use this when an iteration order is required.

**Example:**

|  |
| --- |
| public class Fruit {  public static void main (String[ ] args){  LinkedHashSet&lt;String&gt;; names = new LinkedHashSet &lt;String&gt;( ) ;   names.add(“banana”);   names.add(“cherry”);   names.add(“apple”);   names.add(“kiwi”);   names.add(“banana”);   System.out.println (names);   }  } |

**Output:**

[banana, cherry, apple, kiwi]

It maintains the insertion order in which they have been added to the Set. Duplicates are not allowed.

**c) Tree Set:**

* It is one of the two sorted collections.
* Uses the “Read-Black” tree structure and guarantees that the elements will be in ascending order.
* We can construct a tree set with the constructor by using a comparable (or) comparator.

**Example:**

|  |
| --- |
| public class Fruits{  public static void main (String[ ]args) {  Treeset&lt;String&gt; names= new TreeSet&lt;String&gt;( ) ;  names.add(“cherry”);  names.add(“banana”);  names.add(“apple”);  names.add(“kiwi”);  names.add(“cherry”);  System.out.println(names);  }  } |

**Output:**

[apple, banana, cherry, kiwi]

TreeSet sorts the elements in ascending order. And duplicates are not allowed.

**Q #29) Explain about Map and its types.**

**Answer: Map** cares about the unique identifier. We can map a unique key to a specific value. It is a key/value pair. We can search a value, based on the key. Like the set, the map also uses the “equals ( )” method to determine whether two keys are the same or different.

**Map is of following types:**

**a) Hash Map:**

* Unordered and unsorted map.
* Hashmap is a good choice when we don’t care about the order.
* It allows one null key and multiple null values.

**Example:**

|  |
| --- |
| Public class Fruit{  Public static void main(String[ ] args){  HashMap&lt;Sting,String&gt; names =new HashMap&lt;String,String&gt;( );  names.put(“key1”,“cherry”);  names.put (“key2”,“banana”);  names.put (“key3”,“apple”);  names.put (“key4”,“kiwi”);  names.put (“key1”,“cherry”);  System.out.println(names);  }   } |

**Output:**

{key2 =banana, key1=cherry, key4 =kiwi, key3= apple}

Duplicate keys are not allowed in Map.

It doesn’t maintain any insertion order and is unsorted.

**b) Hash Table:**

* Like the vector key, methods of the class are synchronized.
* Thread safety and therefore slows the performance.
* It doesn’t allow anything that is null.

**Example:**

|  |
| --- |
| public class Fruit{  public static void main(String[ ]args){  Hashtable&lt;Sting,String&gt; names =new Hashtable&lt;String,String&gt;( );  names.put(“key1”,“cherry”);  names.put(“key2”,“apple”);  names.put(“key3”,“banana”);  names.put(“key4”,“kiwi”);  names.put(“key2”,“orange”);  System.out.println(names);  }   } |

**Output:**

{key2=apple, key1=cherry,key4=kiwi, key3=banana}

Duplicate keys are not allowed.

**c) Linked Hash Map:**

* Maintains insertion order.
* Slower than Hash map.
* I can expect a faster iteration.

**Example:**

|  |
| --- |
| public class Fruit{  public static void main(String[ ] args){  LinkedHashMap&lt;Sting,String&gt; names =new LinkedHashMap&lt;String,String&gt;( );   names.put(“key1”,“cherry”);   names.put(“key2”,“apple”);   names.put(“key3”,“banana”);   names.put(“key4”,“kiwi”);   names.put(“key2”,“orange”);   System.out.println(names);   }   } |

**Output:**

{key2=apple, key1=cherry,key4=kiwi, key3=banana}

Duplicate keys are not allowed.

**d) TreeMap:**

* Sorted Map.
* Like Tree set, we can construct a sort order with the constructor.

**Example:**

|  |
| --- |
| public class Fruit{  public static void main(String[ ]args){  TreeMap&lt;Sting,String&gt; names =new TreeMap&lt;String,String&gt;( );  names.put(“key1”,“cherry”);  names.put(“key2”,“banana”);  names.put(“key3”,“apple”);  names.put(“key4”,“kiwi”);  names.put(“key2”,“orange”);  System.out.println(names);  }  } |

**Output:**

{key1=cherry, key2=banana, key3 =apple, key4=kiwi}

It is sorted in ascending order based on the key. Duplicate keys are not allowed.

**Q #30) Explain the Priority Queue.**

**Answer: Queue Interface**

**Priority Queue:**Linked list class has been enhanced to implement the queue interface. Queues can be handled with a linked list. The purpose of a queue is “Priority-in, Priority-out”.

Hence elements are ordered either naturally or according to the comparator. The elements ordering represents their relative priority.

**Q #31) What is meant by Exception?**

**Answer:** An Exception is a problem that can occur during the normal flow of execution. A method can throw an exception when something wails at runtime. If that exception couldn’t be handled, then the execution gets terminated before it completes the task.

If we handled the exception, then the normal flow gets continued. Exceptions are a subclass of java.lang.Exception.

**Example for handling Exception:**

|  |
| --- |
| try{  //Risky codes are surrounded by this block  }catch(Exception e){  //Exceptions are caught in catch block  } |

**Q #32) What are the types of Exceptions?**

**Answer:** There are two types of Exceptions. They are explained below in detail.

**a) Checked Exception:**

These exceptions are checked by the compiler at the time of compilation. Classes that extend Throwable class except Runtime exception and Error are called checked Exception.

Checked Exceptions must either declare the exception using throws keyword (or) surrounded by appropriate try/catch.

**For Example,** ClassNotFound Exception

**b) Unchecked Exception:**

These exceptions are not checked during the compile time by the compiler.  The compiler doesn’t force to handle these exceptions. **It includes:**

* Arithmetic Exception
* ArrayIndexOutOfBounds Exception

**Q #33) What are the different ways to handle exceptions?**

**Answer:** **Two different ways to handle exceptions are explained below:**

**a) Using try/catch:**

The risky code is surrounded by try block. If an exception occurs, then it is caught by the catch block which is followed by the try block.

**Example:**

|  |
| --- |
| class Manipulation{  public static void main(String[] args){  add();  }  Public void add(){  try{  addition();  }catch(Exception e){  e.printStacktrace();  }  }  } |

**b) By declaring throws keyword:**

At the end of the method, we can declare the exception using throws keyword.

**Example:**

|  |
| --- |
| class Manipulation{  public static void main(String[] args){  add();  }  public void add() throws Exception{  addition();  }  } |

**Q #34) What are the advantages of Exception handling?**

**Answer: The advantages are as follows:**

* The normal flow of the execution won’t be terminated if an exception gets handled
* We can identify the problem by using catch declaration

**Q #35) What are the Exception handling keywords in Java?**

**Answer: Enlisted below are the two Exception Handling Keywords:**

**a) try:**

When a risky code is surrounded by a try block. An exception occurring in the try block is caught by a catch block. Try can be followed either by catch (or) finally (or) both. But any one of the blocks is mandatory.

**b) catch:**

This is followed by a try block. Exceptions are caught here.

**c) finally:**

This is followed either by try block (or) catch block. This block gets executed regardless of an exception. So generally clean up codes are provided here.

**Q #36) Explain about Exception Propagation.**

**Answer:** Exception is first thrown from the method which is at the top of the stack. If it doesn’t catch, then it pops up the method and moves to the previous method and so on until they are got.

This is called Exception propagation.

**Example:**

|  |
| --- |
| public class Manipulation{  public static void main(String[] args){  add();  }  public void add(){  addition();  } |

**From the above example, the stack looks like as shown below:**

If an exception occurs in the **addition()** method is not caught, then it moves to the method **add()**. Then it is moved to the **main()** method and then it will stop the flow of execution. It is called Exception Propagation.

**Q #37) What is the final keyword in Java?**

**Answer:**

**Final variable:**Once a variable is declared as final, then the value of the variable could not be changed. It is like a constant.

**Example:**

final int = 12;

**Final method:**A final keyword in a method, couldn’t be overridden. If a method is marked as a final, then it can’t be overridden by the subclass.

**Final class:**If a class is declared as final, then the class couldn’t be subclassed. No class can extend the final class.

**Q #38) What is a Thread?**

**Answer:**In Java, the flow of execution is called Thread. Every java program has at least one thread called the main thread, the main thread is created by JVM. The user can define their own threads by extending the Thread class (or) by implementing the Runnable interface. Threads are executed concurrently.

**Example:**

|  |
| --- |
| public static void main(String[] args){//main thread starts here  } |

**Q #39) How do you make a thread in Java?**

**Answer:**There are two ways available to make a thread.

**a) Extend Thread class:**Extending a Thread class and override the run method. The thread is available in java.lang.thread.

**Example:**

|  |
| --- |
| Public class Addition extends Thread {  public void run () {  }  } |

The disadvantage of using a thread class is that we cannot extend any other classes because we have already extended the thread class. We can overload the run () method in our class.

**b) Implement Runnable interface:**Another way is by implementing the runnable interface. For that, we should provide the implementation for the run () method which is defined in the interface.

**Example:**

|  |
| --- |
| Public class Addition implements Runnable {  public void run () {  }  } |

**Q #40) Explain about join () method.**

**Answer:** Join () method is used to join one thread with the end of the currently running thread.

**Example:**

|  |
| --- |
| public static void main (String[] args){  Thread t = new Thread ();  t.start ();  t.join ();  } |

Based on the above code, the main thread has started the execution. When it reaches the code ***t.start()*** then ‘thread t’ starts the own stack for the execution. JVM switches between the main thread and ‘thread t’.

Once it reaches the code ***t.join()*** then ‘thread t’ alone is executed and completes its task, then only the main thread starts the execution.

It is a non-static method. The Join () method has an overloaded version. So we can mention the time duration in join () method also “.s”.

**Q #41) What does the yield method of the Thread class do?**

**Answer:** A yield () method moves the currently running thread to a runnable state and allows the other threads for execution. So that equal priority threads have a chance to run. It is a static method. It doesn’t release any lock.

Yield () method moves the thread back to the Runnable state only, and not the thread to sleep (), wait () (or) block.

**Example:**

|  |
| --- |
| public static void main (String[] args){  Thread t = new Thread ();  t.start ();  }  public void run(){  Thread.yield();  }  } |

**Q #42) Explain about wait () method.**

**Answer: wait ()** method is used to make the thread to wait in the waiting pool. When the wait () method is executed during a thread execution then immediately the thread gives up the lock on the object and goes to the waiting pool. Wait () method tells the thread to wait for a given amount of time.

Then the thread will wake up after notify () (or) notify all () method is called.

Wait() and the other above-mentioned methods do not give the lock on the object immediately until the currently executing thread completes the synchronized code. It is mostly used in synchronization.

**Example:**

|  |
| --- |
| public static void main (String[] args){  Thread t = new Thread ();  t.start ();  Synchronized (t) {  Wait();  }  } |

**Q #43) Difference between notify() method and notifyAll() method in Java.**

**Answer: The differences between notify() method and notifyAll() method are enlisted below:**

| **notify()** | **notifyAll()** |
| --- | --- |
| This method is used to send a signal to wake up a single thread in the waiting pool. | This method sends the signal to wake up all the threads in a waiting spool. |

**Q #44) How to stop a thread in java? Explain about sleep () method in a thread?**

**Answer:** **We can stop a thread by using the following thread methods:**

* Sleeping
* Waiting
* Blocked

**Sleep:**Sleep () method is used to sleep the currently executing thread for the given amount of time. Once the thread is wake up it can move to the runnable state. So sleep () method is used to delay the execution for some period.

It is a static method.

**Example:**

**Thread. Sleep (2000)**

So it delays the thread to sleep 2 milliseconds. Sleep () method throws an uninterrupted exception, hence we need to surround the block with try/catch.

|  |
| --- |
| public class ExampleThread implements Runnable{  public static void main (String[] args){  Thread t = new Thread ();  t.start ();  }  public void run(){  try{  Thread.sleep(2000);  }catch(InterruptedException e){  }  } |

**Q #45) When to use the Runnable interface Vs Thread class in Java?**

**Answer:** If we need our class to extend some other classes other than the thread then we can go with the runnable interface because in java we can extend only one class.

If we are not going to extend any class then we can extend the thread class.

**Q #46) Difference between start() and run() method of thread class.**

**Answer:** Start() method creates a new thread and the code inside the run () method is executed in the new thread. If we directly called the run() method then a new thread is not created and the currently executing thread will continue to execute the run() method.

**Q #47) What is Multi-threading?**

**Answer:** Multiple threads are executed simultaneously. Each thread starts its own stack based on the flow (or) priority of the threads.

**Example Program:**

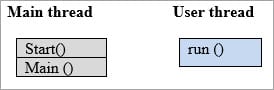
|  |
| --- |
| public class MultipleThreads implements Runnable  {  public static void main (String[] args){//Main thread starts here  Runnable r = new runnable ();  Thread t=new thread ();  t.start ();//User thread starts here  Addition add=new addition ();  }  public void run(){  go();  }//User thread ends here  } |

On the 1st line execution, JVM calls the main method and the main thread stack looks as shown below.

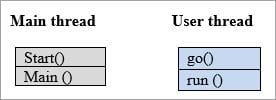
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Thread1.jpg)

Once the execution reaches, **t.start ()**line then a new thread is created and the new stack for the thread is also created. Now JVM switches to the new thread and the main thread are back to the runnable state.

The two stacks look as shown below.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Thread2.jpg)

Now, the user thread executed the code inside the run() method.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Thread3.jpg)

Once the run() method has completed, then JVM switches back to the main thread and the user thread has completed the task and the stack was disappeared.

JVM switches between each thread until both the threads are completed. This is called Multi-threading.

**Q #48) Explain the thread life cycle in Java.**

**Answer:** **Thread has the following states:**

* New
* Runnable
* Running
* Non-runnable (Blocked)
* Terminated

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Thread-Life-Cycle-in-Java.jpg)

* **New:**In New state, a Thread instance has been created but start () method is not yet invoked. Now the thread is not considered alive.
* **Runnable**: The Thread is in the runnable state after the invocation of the start () method, but before the run () method is invoked. But a thread can also return to the runnable state from waiting/sleeping. In this state, the thread is considered alive.
* **Running**: The thread is in a running state after it calls the run () method. Now the thread begins the execution.
* **Non-Runnable**(Blocked): The thread is alive but it is not eligible to run. It is not in the runnable state but also, it will return to the runnable state after some time. **Example:** wait, sleep, block.
* **Terminated**: Once the run method is completed then it is terminated. Now the thread is not alive.

**Q #49) What is Synchronization?**

**Answer:** Synchronization makes only one thread to access a block of code at a time. If multiple threads accesses the block of code, then there is a chance for inaccurate results at the end. To avoid this issue, we can provide synchronization for the sensitive block of codes.

The synchronized keyword means that a thread needs a key in order to access the synchronized code.

Locks are per objects. Every Java object has a lock. A lock has only one key. A thread can access a synchronized method only if the thread can get the key to the objects to lock.

For this, we use the “Synchronized” keyword.

**Example:**

|  |
| --- |
| public class ExampleThread implements Runnable{   public static void main (String[] args){   Thread t = new Thread ();   t.start ();   }   public void run(){   synchronized(object){   {   }  } |

**Q #50) What is the disadvantage of Synchronization?**

**Ans:** Synchronization is not recommended to implement all the methods. Because if one thread accesses the synchronized code then the next thread should have to wait. So it makes a slow performance on the other end.

**Q #51) What is meant by Serialization?**

**Answer:** Converting a file into a byte stream is known as Serialization. The objects in the file are converted to bytes for security purposes. For this, we need to implement a java.io.Serializable interface. It has no method to define.

Variables that are marked as transient will not be a part of the serialization. So we can skip the serialization for the variables in the file by using a transient keyword.

***Learn More =>>***[***Serializable and Cloneable***](https://www.softwaretestinghelp.com/marker-interfaces-java/)

**Q #52) What is the purpose of a transient variable?**

**Answer:** Transient variables are not part of the serialization process. During deserialization, the values of the transient variables are set to the default value. It is not used with static variables.

**Example:**

transient int numbers;

**Q #53) Which methods are used during the Serialization and Deserialization process?**

**Answer:** ObjectOutputStream and ObjectInputStream classes are higher level java.io. package. We will use them with lower level classes FileOutputStream and FileInputStream.

ObjectOutputStream.writeObject**—->**Serialize the object and write the serialized object to a file.

ObjectInputStream.readObject **—>** Reads the file and deserializes the object.

To be serialized, an object must implement the serializable interface. If superclass implements Serializable, then the subclass will automatically be serializable.

**Q #54) What is the purpose of a Volatile Variable?**

**Answer:** Volatile variable values are always read from the main memory and not from thread’s cache memory. This is used mainly during synchronization. It is applicable only for variables.

**Example:**

volatile int number;

**Q #55) Difference between Serialization and Deserialization in Java.**

**Answer:** **These are the differences between serialization and deserialization in java:**

| **Serialization** | **Deserialization** |
| --- | --- |
| Serialization is the process which is used to convert the objects into byte stream | Deserialization is the opposite process of serialization where we can get the objects back from the byte stream. |
| An object is serialized by writing it an ObjectOutputStream. | An object is deserialized by reading it from an ObjectInputStream. |

**Q #56) What is SerialVersionUID?**

**Answer:** Whenever an object is Serialized, the object is stamped with a version ID number for the object class. This ID is called the  SerialVersionUID. This is used during deserialization to verify that the sender and receiver that are compatible with the Serialization.

**Q #57) What do you mean by an Array?**

* Array is a set of similar data type.
* Arrays objects store multiple variables with the same type.
* It can hold primitive types and object references.
* Arrays are always fixed

**Q #58) How to create an Array?**

An Array is declared similar to how a variable is declared, but you need to add [] after the type.

**Q #59) Advantages and disadvantages of Array?**

**Advantages:**

* We can put in place other data structures like stacks, queues, linked lists, trees, graphs, etc. in Array.
* Arrays can sort multiple elements at a time.
* We can access an element of Array by using an index.

**Disadvantages:**

* We have to declare Size of an array in advance. However, we may not know what size we need at the time of array declaration.
* The array is static structure. It means array size is always fixed, so we cannot increase or decrease memory allocation.

**Q #60) Can we change the size of an array at run time?**

No we cannot change the array size. Though there are similar data types available which allow a change in size.

**Q #61) What is the default value of Array?**

Any new Array is always initialized with a default value as follows

* For byte, short, int, long – default value is zero (0).
* For float, double – default value is 0.0.
* For Boolean – default value is false.
* For object – default value is null.

**Q #62) Can we declare array size as a negative number?**

* No. We cannot declare the negative integer as an array size.
* If we declare, there will be no compile-time error.
* However, we will get NegativeArraySizeException at run time.

**Q #63) When will we get ArrayStoreException?**

* It is a runtime exception. For example, we can store only string elements in a String Array. If anybody tries to insert integer element in this String Array, then we will get ArrayStoreException at run time.

**Q #64) Can we add or delete an element after assigning an array?**

* No it is not possible.

**Q #65) What is the meaning of anonymous array? Explain with an example?**

* Anonymous array means array without any reference.

==================================================

**String**

**1. How to declare a string in Java?**

String declaration in Java can be done in two ways:

* **By string literal**: Double quotes are used to create Java String literals.
  + Example: String str= "Scaler";
* **By new keyword**: Keyword "new" is used to create a Java string.
  + Example: String str=new String ("Scaler");

### 2. Explain String pool in Java.

String Pool, also known as SCP (String Constant Pool), is a special storage space in Java heap memory that is used to store unique string objects. Whenever a string object is created, it first checks whether the String object with the same string value is already present in the String pool or not, and if it is available, then the reference to the string object from the string pool is returned. Otherwise, the new string object is added to the string pool, and the respective reference will be return.

### 3. Is String immutable or final in Java? If so, then what are the benefits of Strings being Immutable?

Yes, Strings are immutable in Java. Immutable objects mean they can't be changed or altered once they've been created. However, we can only modify the reference to the string object. The String is immutable in Java because of many reasons like security, caching, synchronization and concurrency, and class loading.

### 4. What does the string intern() method do in Java?

If you apply the intern() method to a few strings, you will ensure that all strings having the same content share the same memory. As soon as a String object is invoked with intern(), it first checks if the string value of the String object is already present in the string pool and if it is available, then the reference to that string from the string constant pool is returned. If not, a new string object is added to the string pool, and a reference to it is returned

### 5.What are the different string methods in Java?

There are various string operations in Java that allow us to work with strings. These methods or operations can be used for string handling in Java as well as string manipulation in Java. Some of such methods are as follows:

* **split()**: Split/divide the string at the specified regex.
* **compareTo()**: Compares two strings on the basis of the Unicode value of each string character.
* **compareToIgnoreCase()**: Similar to compareTo, but it also ignores case differences.
* **length()**: Returns the length of the specified string.
* **substring()**: Returns the substring from the specified string.
* **equalsIgnoreCase()**: Compares two strings ignoring case differences.
* **contains()**: Checks if a string contains a substring.
* **trim()**: Returns the substring after removing any leading and trailing whitespace from the specified string.
* **charAt()**: Returns the character at specified index.
* **toLowerCase()**: Converts string characters to lower case.
* **toUpperCase()**: Converts string characters to upper case.
* **concat()**: Concatenates two strings.

===============================================================================

### MYSQL

### 1. What is a Relational Database Management System?

**Sample answer**:

A Relational Database Management System (RDBMS) refers to the software used to store, manage, and query data. Data is stored in tables and can be linked to other datasets based on shared information, hence the name “relational”.

### 2. How does a Relational Database Management System differ from a Database Management System?

**Sample answer**:

The key differences between Relational Database Management Systems (RDBMS) and Database Management Systems (DBMS) are:

* An RDBMS stores data in a relational table with rows and columns, whereas a DBMS stores data as a file
* An RDBMS provides access to multiple users (including client-server side interaction), whereas a DBMS only supports single users

### 3. What are some of the most popular Relational Database Management Systems?

**Sample answer**:

Some of the most popular RDBMSs are:

* Oracle Database
* MySQL
* Microsoft SQL Server
* PostgreSQL
* IBM DB2
* SQLite

### 4. What is the role of SQL?

**Sample answer**:

SQL is a programming language used to perform data-related tasks; every RDBMS uses SQL as its standard programming language. In these databases, SQL allows users to create tables, update data, make queries, and perform analytics.

### 5. What is the difference between SQL and MySQL?

**Sample answer**:

SQL is the programming language used in an RDBMS, while MySQL is an example of an RDBMS. MySQL was one of the first open-source database systems on the market, and it is still fairly popular today.

### 6. How do you create a table with SQL?

**Sample answer**:

The CREATE TABLE command is used to create a new table in an RDBMS. This command prompts users to fill in the table name, the column names, and the types of data. The same command can also be used to make copies of existing tables.

### 7. How do you insert dates with SQL?

**Sample answer**:

With SQL, the DATE data type is used to store data or time values in the database. The format for inserting dates can vary depending on the RDBMS, but generally it’s ‘YYYY-MM-DD’.

### 8. What is a query?

**Sample answer**:

A query is a request for data or information from a database. There are two main types of SQL queries:

* **A select query** is a query that groups data from a table for analytical purposes
* **An action query** is a query that changes the contents of the database based on specified criteria

### 9. What is a subquery?

**Sample answer**:

A subquery is a query that is embedded within another statement that requires multiple steps. The subquery provides the enclosing query with additional information needed to execute a task, such as when the completion of one query depends firstly on the results of another.

### 10. How do you perform a select query with SQL?

**Sample answer**:

The process for performing a select query in SQL is as follows:

* The **SELECT**statement is used to specify the columns you want to query
* The **FROM**statement is used to specify the particular table holding the data
* The **WHERE**statement is used to filter data based on specified conditions

### 11. What are the most important types of action queries?

**Sample answer**:

There are several SQL statements for running an action query. Their purposes and procedures vary. Some of the important action statements include:

* **UPDATE**modifies the values of fields in a table
* **DELETE**removes records from a table
* **CREATE TABLE** creates a new table
* **INSERT INTO** adds records to a table

### 12. What are constraints?

**Sample answer**:

SQL constraints are a set of rules or conditions implemented on an RDBMS to specify what data can be inserted, updated, or deleted in its tables. This is done to maintain data integrity and ensure that the information stored in database tables is accurate.

### 13. What are join clauses?

**Sample answer**:

The join clause combines columns with related values from two or more tables to create a new table. There are four main types of SQL join clause:

* **JOIN** returns records with matching values in both tables
* **LEFT JOIN** returns all records from the left table and matching records from the right table
* **RIGHT JOIN** returns all records from the right table and matching records from the left table
* **FULL JOIN** returns all records from both tables

### 14. What is the role of indexes?

**Sample answer**:

An SQL index stores important parts of a database table to allow for a quick and efficient lookup. Rather than searching the entire database, users only have to consult the index during data retrieval. Indexes, therefore, help improve performance in an RDBMS.

### 15. What does a NULL value represent?

**Sample answer**:

A NULL value indicates the data is unknown. This is not the same as 0; NULL values mean no data is stored at all.

### 16. How do you delete a column?

**Sample answer**:

A column in a table can be deleted by following these steps:

* Use ‘ALTER TABLE table name’ to select the table with the column you want to delete
* Use ‘DROP COLUMN column name’ to select the column you want to delete

### 17. How would you write a query to identify employees belonging to a particular department?

**Sample answer**:

Finding data entries belonging to a particular group (in this case, employees belonging to a particular department) can be achieved in a few ways. These include:

* Use SELECT, FROM, and WHERE statements
* Use SELECT, FROM, GROUP BY, and HAVING statements
* Use SELECT, FROM, INNER JOIN, and WHERE statements

### 18.How would you write a SQL query to find entrants whose names begin with A?

**Sample answer**:

You can retrieve data entries beginning with a particular letter using the LIKE command by following these steps:

* Use the SELECT statement to specify the column with the names you want to vet
* Use the FROM statement to specify the table containing that column
* Use ‘WHERE column name’ followed by ‘LIKE x%’, with x representing the letter you are searching for
* Use ‘ORDER by column name’ to complete the query

### 19. What are some of the most important aggregate functions?

**Sample answer**:

Aggregate values are used to perform calculations on a set of values to return a single value. Some of the most widely used aggregate functions are:

* **AVG**calculates the average set of values
* **COUNT**counts the total number of rows in a table
* **MIN**finds the minimum value in a table
* **MAX**finds the maximum value in a table
* **SUM**calculates the sum of the values

### 20. What does schema mean?

**Sample answer**:

A schema refers to a collection of database objects—such as tables, functions, indexes, and procedures—associated with a database.

The schema helps segregate database objects for different applications and access rights; it’s generally used to define who can and who cannot view specific objects in the database.

### 21. What are some of the most important scalar functions?

**Sample answer**:

Scalar functions are user-defined functions applied to a set of data to return a single value. Some of the most common scalar functions include:

* **UCASE**converts values to uppercase
* **LCASE**converts values to lowercase
* **MID**extracts textual data based on specified criteria
* **ROUND**rounds numerical data to a specified number of decimals
* **NOW**returns the current system date and time

### 22. What are SQL injections and how can they be prevented?

**Sample answer**:

An SQL injection is a type of cyber attack in which hackers insert malicious SQL code into the database to gain access to potentially valuable or sensitive information. It’s a fairly common occurrence with web applications or websites that use an SQL-based database.

It’s possible to prevent SQL injections by creating multiple database accounts to limit access or by using a third-party web application firewall.

### 23. How do you remove duplicate rows from a table?

**Sample answer**:

There are several ways to remove duplicate rows from a table. These include:

* Using Common Table Expressions (CTE) with the ROW\_NUMBER function to identify and remove duplicate rows
* Using the RANK function with the PARTITION BY clause
* Using the GROUP BY clause with the COUNT function, and then replacing SELECT with DELETE FROM

e article will walk you through the Spring Boot interview questions for basic to advanced level.

================================================================================Spring Boot

## **1.** What is Spring boot?

[**Sprint boot**](https://www.interviewbit.com/java-interview-questions/) is a Java-based spring framework used for Rapid Application Development (to build stand-alone microservices). It has extra support of auto-configuration and embedded application server like tomcat, jetty, etc.

## Features of Spring Boot that make it different?

* Creates stand-alone spring application with minimal configuration needed.
* It has embedded tomcat, jetty which makes it just code and run the application.
* Provide production-ready features such as metrics, health checks, and externalized configuration.
* Absolutely no requirement for XML configuration.

1. What are the advantages of using Spring Boot? Spring Boot Features

The advantages of Spring Boot are listed below:

* Easy to understand and develop spring applications.
* Spring Boot is nothing but an existing framework with the addition of an embedded HTTP server and annotation configuration which makes it easier to understand and faster the process of development.
* Increases productivity and reduces development time.
* Minimum configuration.
* We don’t need to write any XML configuration, only a few annotations are required to do the configuration.

### 2. What are the Spring Boot key components?

Below are the four key components of spring-boot:

* Spring Boot auto-configuration.
* Spring Boot CLI.
* Spring Boot starter POMs.
* Spring Boot Actuators.

### 3. Why Spring Boot over Spring?

Below are some key points which spring boot offers but spring doesn’t:

* Starter POM.
* Version Management.
* Auto Configuration.
* Component Scanning.
* Embedded server.
* InMemory DB.
* Actuators

Spring Boot simplifies the spring feature for the user:

### 4. What is the starter dependency of the Spring boot module?

Spring boot provides numbers of starter dependency, here are the most commonly used -

* Data JPA starter.
* Test Starter.
* Security starter.
* Web starter.
* Mail starter.
* Thymeleaf starter.

### 5. How does Spring Boot works?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using annotation. The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.

### 6. What does the @SpringBootApplication annotation do internally?

The @SpringBootApplication annotation is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes. Spring Boot enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs.

### 7. What is the purpose of using @ComponentScan in the class files?

Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the @ComponentScan annotation for your class file to scan your components added to your project.

### 8. How does a spring boot application get started?

Just like any other Java program, a Spring Boot application must have a main method. This method serves as an entry point, which invokes the SpringApplication#run method to bootstrap the application.

@SpringBootApplication

**public** **class** **MyApplication** {

**public** **static** **void** **main**(String[] args) {

SpringApplication.run(MyApplication.class);

// other statements

}

}

### 9. What are starter dependencies?

Spring boot starter is a maven template that contains a collection of all the relevant transitive dependencies that are needed to start a particular functionality.  
Like we need to import spring-boot-starter-web dependency for creating a web application.

<dependency>

<groupId> org.springframework.boot</groupId>

<artifactId> spring-boot-starter-web </artifactId>

</dependency>

### 10. What is Spring Initializer?

Spring Initializer is a web application that helps you to create an initial spring boot project structure and provides a maven or gradle file to build your code. It solves the problem of setting up a framework when you are starting a project from scratch.

### 11. What is Spring Boot CLI and what are its benefits?

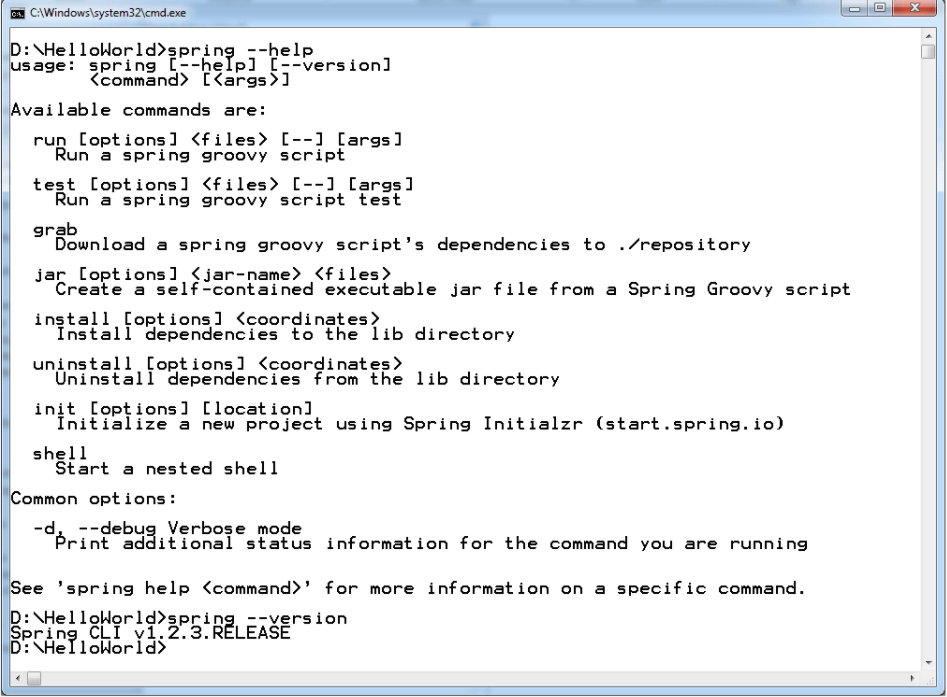
Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

Example: You don’t need to create getter and setter method or access modifier, return statement. If you use the JDBC template, it automatically loads for you.

### 12. What are the most common Spring Boot CLI commands?

-run, -test, -grap, -jar, -war, -install, -uninstall, --init, -shell, -help.

To check the description, run spring --help from the terminal.

Spring Boot CLI Commands

## Advanced Spring Boot Questions

### 13. What Are the Basic Annotations that Spring Boot Offers?

The primary annotations that Spring Boot offers reside in its org.springframework.boot.autoconfigure and its sub-packages. Here are a couple of basic ones:

@EnableAutoConfiguration – to make Spring Boot look for auto-configuration beans on its classpath and automatically apply them.

@SpringBootApplication – used to denote the main class of a Boot Application. This annotation combines @Configuration, @EnableAutoConfiguration, and @ComponentScan annotations with their default attributes.

### 14. What is Spring Boot dependency management?

Spring Boot dependency management is used to manage dependencies and configuration automatically without you specifying the version for any of that dependencies.

### 15. Can we create a non-web application in Spring Boot?

Yes, we can create a non-web application by removing the web dependencies from the classpath along with changing the way Spring Boot creates the application context.

### 16. Is it possible to change the port of the embedded Tomcat server in Spring Boot?

Yes, it is possible. By using the **server.port** in the **application.properties**.

### 17. What is the default port of tomcat in spring boot?

The default port of the tomcat server-id 8080. It can be changed by adding **sever.port** properties in the **application.property** file.

### 18. Can we override or replace the Embedded tomcat server in Spring Boot?

Yes, we can replace the Embedded Tomcat server with any server by using the Starter dependency in the **pom.xml** file. Like you can use spring-boot-starter-jetty as a dependency for using a jetty server in your project.

### 19. Can we disable the default web server in the Spring boot application?

Yes, we can use **application.properties** to configure the web application type i.e **spring.main.web-application-type=none.**

### 20. How to disable a specific auto-configuration class?

You can use exclude attribute of @EnableAutoConfiguration if you want auto-configuration not to apply to any specific class.

//use of exclude

@EnableAutoConfiguration(exclude={className})

### 21. Explain @RestController annotation in Sprint boot?

It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

### 22. What is the difference between @RestController and @Controller in Spring Boot?

@Controller Map of the model object to view or template and make it human readable but @RestController simply returns the object and object data is directly written in HTTP response as JSON or XML.

### 23. Describe the flow of HTTPS requests through the Spring Boot application?

On a high-level spring boot application follow the MVC pattern which is depicted in the below flow diagram.

Spring Boot Flow Architecture

### 24. What is the difference between RequestMapping and GetMapping?

RequestMapping can be used with GET, POST, PUT, and many other request methods using the method attribute on the annotation. Whereas getMapping is only an extension of RequestMapping which helps you to improve on clarity on request.

### 25. What is the use of Profiles in spring boot?

While developing the application we deal with multiple environments such as dev, QA, Prod, and each environment requires a different configuration. For eg., we might be using an embedded H2 database for dev but for prod, we might have proprietary Oracle or DB2. Even if DBMS is the same across the environment, the URLs will be different.

To make this easy and clean, Spring has the provision of Profiles to keep the separate configuration of environments.

### 26. What is Spring Actuator? What are its advantages?

An actuator is an additional feature of Spring that helps you to monitor and manage your application when you push it to production. These actuators include auditing, health, CPU usage, HTTP hits, and metric gathering, and many more that are automatically applied to your application.

### 27. How to enable Actuator in Spring boot application?

To enable the spring actuator feature, we need to add the dependency of “spring-boot-starter-actuator” in pom.xml.

<dependency>

<groupId> org.springframework.boot</groupId>

<artifactId> spring-boot-starter-actuator </artifactId>

</dependency>

### 28. What are the actuator-provided endpoints used for monitoring the Spring boot application?

Actuators provide below pre-defined endpoints to monitor our application -

* Health
* Info
* Beans
* Mappings
* Configprops
* Httptrace
* Heapdump
* Threaddump
* Shutdown

### 29. How to get the list of all the beans in your Spring boot application?

Spring Boot actuator “/Beans” is used to get the list of all the spring beans in your application.

### 30. How to check the environment properties in your Spring boot application?

Spring Boot actuator “/env” returns the list of all the environment properties of running the spring boot application.

### 31. How to enable debugging log in the spring boot application?

Debugging logs can be enabled in three ways -

* We can start the application with --debug switch.
* We can set the logging.level.root=debug property in application.property file.
* We can set the logging level of the root logger to debug in the supplied logging configuration file.

### 32. Where do we define properties in the Spring Boot application?

You can define both application and Spring boot-related properties into a file called application.properties. You can create this file manually or use Spring Initializer to create this file. You don’t need to do any special configuration to instruct Spring Boot to load this file, If it exists in classpath then spring boot automatically loads it and configure itself and the application code accordingly.

### 33. What is dependency Injection?

The process of injecting dependent bean objects into target bean objects is called dependency injection.

* Setter Injection: The IOC container will inject the dependent bean object into the target bean object by calling the setter method.
* Constructor Injection: The IOC container will inject the dependent bean object into the target bean object by calling the target bean constructor.
* Field Injection: The IOC container will inject the dependent bean object into the target bean object by Reflection API.

### 34. What is an IOC container?

IoC Container is a framework for implementing automatic dependency injection. It manages object creation and its life-time and also injects dependencies into the class

### Collection

### 1) What is the Collection framework in Java?

Collection Framework is a combination of classes and interface, which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.

### 2) What are the main differences between array and collection?

Array and Collection are somewhat similar regarding storing the references of objects and manipulating the data, but they differ in many ways. The main differences between the array and Collection are defined below:

* Arrays are always of fixed size, i.e., a user can not increase or decrease the length of the array according to their requirement or at runtime, but In Collection, size can be changed dynamically as per need.
* Arrays can only store homogeneous or similar type objects, but in Collection, heterogeneous objects can be stored.
* Arrays cannot provide the ?ready-made? methods for user requirements as sorting, searching, etc. but Collection includes readymade methods to use.

### 3) Explain various interfaces used in Collection framework?

Collection framework implements various interfaces, Collection interface and Map interface (java.util.Map) are the mainly used interfaces of Java Collection Framework. List of interfaces of Collection Framework is given below:

**1. Collection interface:** Collection (java.util.Collection) is the primary interface, and every collection must implement this interface.

**Syntax:**

1. **public** **interface** Collection<E>**extends** Iterable

Where <E> represents that this interface is of Generic type

**2. List interface:**List interface extends the Collection interface, and it is an ordered collection of objects. It contains duplicate elements. It also allows random access of elements.

**Syntax:**

1. **public** **interface** List<E> **extends** Collection<E>

**3. Set interface:** Set (java.util.Set) interface is a collection which cannot contain duplicate elements. It can only include inherited methods of Collection interface

**Syntax:**

1. **public** **interface** Set<E> **extends** Collection<E>

**Queue interface:**Queue (java.util.Queue) interface defines queue data structure, which stores the elements in the form FIFO (first in first out).

**Syntax:**

1. **public** **interface** Queue<E> **extends** Collection<E>

**4. Dequeue interface:** it is a double-ended-queue. It allows the insertion and removal of elements from both ends. It implants the properties of both Stack and queue so it can perform LIFO (Last in first out) stack and FIFO (first in first out) queue, operations.

**Syntax:**

1. **public** **interface** Dequeue<E> **extends** Queue<E>

**5. Map interface:**A Map (java.util.Map) represents a key, value pair storage of elements. Map interface does not implement the Collection interface. It can only contain a unique key but can have duplicate elements. There are two interfaces which implement Map in java that are Map interface and Sorted Map.

### 4) What is the difference between ArrayList and Vector?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList is not synchronized. | Vector is synchronized. |
| 2) | ArrayList is not a legacy class. | Vector is a legacy class. |
| 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by doubling the array size. |
| 4) | ArrayList is not ?thread-safe? as it is not synchronized. | Vector list is ?thread-safe? as it?s every method is synchronized. |

### 5) What is the difference between ArrayList and LinkedList?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses a doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because too much is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |
| 4) | ArrayList provides random access. | LinkedList does not provide random access. |
| 5) | ArrayList takes less memory overhead as it stores only object | LinkedList takes more memory overhead, as it stores the object as well as the address of that object. |

### 6) What is the difference between Iterator and ListIterator?

Iterator traverses the elements in the forward direction only whereas ListIterator traverses the elements into forward and backward direction.

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | The Iterator traverses the elements in the forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | The Iterator can be used in List, Set, and Queue. | ListIterator can be used in List only. |
| 3) | The Iterator can only perform remove operation while traversing the collection. | ListIterator can perform ?add,? ?remove,? and ?set? operation while traversing the collection. |

### 7) What is the difference between Iterator and Enumeration?

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | The Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | The Iterator is fail-fast. | Enumeration is not fail-fast. |
| 3) | The Iterator is slower than Enumeration. | Enumeration is faster than Iterator. |
| 4) | The Iterator can perform remove operation while traversing the collection. | The Enumeration can perform only traverse operation on the collection. |

### 8) What is the difference between List and Set?

The List and Set both extend the collection interface. However, there are some differences between the both which are listed below.

* The List can contain duplicate elements whereas Set includes unique items.
* The List is an ordered collection which maintains the insertion order whereas Set is an unordered collection which does not preserve the insertion order.
* The List interface contains a single legacy class which is Vector class whereas Set interface does not have any legacy class.
* The List interface can allow n number of null values whereas Set interface only allows a single null value.

### 9) What is the difference between HashSet and TreeSet?

The HashSet and TreeSet, both classes, implement Set interface. The differences between the both are listed below.

* HashSet maintains no order whereas TreeSet maintains ascending order.
* HashSet impended by hash table whereas TreeSet implemented by a Tree structure.
* HashSet performs faster than TreeSet.
* HashSet is backed by HashMap whereas TreeSet is backed by TreeMap.

### 10) What is the difference between Set and Map?

The differences between the Set and Map are given below.

* Set contains values only whereas Map contains key and values both.
* Set contains unique values whereas Map can contain unique Keys with duplicate values.
* Set holds a single number of null value whereas Map can include a single null key with n number of null values.

### 11) What is the difference between HashSet and HashMap?

The differences between the HashSet and HashMap are listed below.

* HashSet contains only values whereas HashMap includes the entry (key, value). HashSet can be iterated, but HashMap needs to convert into Set to be iterated.
* HashSet implements Set interface whereas HashMap implements the Map interface
* HashSet cannot have any duplicate value whereas HashMap can contain duplicate values with unique keys.
* HashSet contains the only single number of null value whereas HashMap can hold a single null key with n number of null values.

### 12) What is the difference between HashMap and TreeMap?

The differences between the HashMap and TreeMap are given below.

* HashMap maintains no order, but TreeMap maintains ascending order.
* HashMap is implemented by hash table whereas TreeMap is implemented by a Tree structure.
* HashMap can be sorted by Key or value whereas TreeMap can be sorted by Key.
* HashMap may contain a null key with multiple null values whereas TreeMap cannot hold a null key but can have multiple null values.

### 13) What is the difference between HashMap and Hashtable?

|  |  |  |
| --- | --- | --- |
| **No.** | **HashMap** | **Hashtable** |
| 1) | HashMap is not synchronized. | Hashtable is synchronized. |
| 2) | HashMap can contain one null key and multiple null values. | Hashtable cannot contain any null key or null value. |
| 3) | HashMap is not ?thread-safe,? so it is useful for non-threaded applications. | Hashtable is thread-safe, and it can be shared between various threads. |
| 4) | 4) HashMap inherits the AbstractMap class | Hashtable inherits the Dictionary class. |

### 14) What is the difference between Collection and Collections?

The differences between the Collection and Collections are given below.

* The Collection is an interface whereas Collections is a class.
* The Collection interface provides the standard functionality of data structure to List, Set, and Queue. However, Collections class is to sort and synchronize the collection elements.
* The Collection interface provides the methods that can be used for data structure whereas Collections class provides the static methods which can be used for various operation on a collection.

### 15) What is the difference between Comparable and Comparator?

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 1) | Comparable provides only one sort of sequence. | The Comparator provides multiple sorts of sequences. |
| 2) | It provides one method named compareTo(). | It provides one method named compare(). |
| 3) | It is found in java.lang package. | It is located in java.util package. |
| 4) | If we implement the Comparable interface, The actual class is modified. | The actual class is not changed. |

### 16) What do you understand by BlockingQueue?

BlockingQueue is an interface which extends the Queue interface. It provides concurrency in the operations like retrieval, insertion, deletion. While retrieval of any element, it waits for the queue to be non-empty. While storing the elements, it waits for the available space. BlockingQueue cannot contain null elements, and implementation of BlockingQueue is thread-safe.

**Syntax:**

1. **public** **interface** BlockingQueue<E> **extends** Queue <E>

### 17) What is the advantage of Properties file?

If you change the value in the properties file, you don't need to recompile the java class. So, it makes the application easy to manage. It is used to store information which is to be changed frequently. Consider the following example.

1. **import** java.util.\*;
2. **import** java.io.\*;
3. **public** **class** Test {
4. **public** **static** **void** main(String[] args)**throws** Exception{
5. FileReader reader=**new** FileReader("db.properties");
7. Properties p=**new** Properties();
8. p.load(reader);
10. System.out.println(p.getProperty("user"));
11. System.out.println(p.getProperty("password"));
12. }
13. }

**Output**

system

oracle

### 18) What does the hashCode() method?

The hashCode() method returns a hash code value (an integer number).

The hashCode() method returns the same integer number if two keys (by calling equals() method) are identical.

However, it is possible that two hash code numbers can have different or the same keys.

If two objects do not produce an equal result by using the equals() method, then the hashcode() method will provide the different integer result for both the objects.

### 19) Why we override equals() method?

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

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

### 20) How to synchronize List, Set and Map elements?

Yes, Collections class provides methods to make List, Set or Map elements as synchronized:

|  |
| --- |
| public static List synchronizedList(List l){} |
| public static Set synchronizedSet(Set s){} |
| public static SortedSet synchronizedSortedSet(SortedSet s){} |
| public static Map synchronizedMap(Map m){} |
| public static SortedMap synchronizedSortedMap(SortedMap m){} |

### 21) What is the advantage of the generic collection?

There are three main advantages of using the generic collection.

* If we use the generic class, we don't need typecasting.
* It is type-safe and checked at compile time.
* Generic confirms the stability of the code by making it bug detectable at compile time.

### 22) What is hash-collision in Hashtable and how it is handled in Java?

Two different keys with the same hash value are known as hash-collision. Two separate entries will be kept in a single hash bucket to avoid the collision. There are two ways to avoid hash-collision.

* Separate Chaining
* Open Addressing

### 23) What is the Dictionary class?

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

### 24) What is the default size of load factor in hashing based collection?

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

### 25) What do you understand by fail-fast?

The Iterator in java which immediately throws ConcurrentmodificationException, if any structural modification occurs in, is called as a Fail-fast iterator. Fail-fats iterator does not require any extra space in memory.

### 26) [What is the difference between Array and ArrayList?](https://www.javatpoint.com/array-vs-arraylist-in-java)

The main differences between the Array and ArrayList are given below.

|  |  |  |
| --- | --- | --- |
| **SN** | **Array** | **ArrayList** |
| 1 | The Array is of fixed size, means we cannot resize the array as per need. | ArrayList is not of the fixed size we can change the size dynamically. |
| 2 | Arrays are of the static type. | ArrayList is of dynamic size. |
| 3 | Arrays can store primitive data types as well as objects. | ArrayList cannot store the primitive data types it can only store the objects. |

### 27) [What is the difference between the length of an Array and size of ArrayList?](https://www.javatpoint.com/difference-between-length-of-array-and-size-of-arraylist-in-java)

The length of an array can be obtained using the property of length whereas ArrayList does not support length property, but we can use size() method to get the number of objects in the list.

**Finding the length of the array**

1. Int [] array = **new** **int**[4];
2. System.out.println("The size of the array is " + array.length);

**Finding the size of the ArrayList**

1. ArrayList<String> list=**new** ArrayList<String>();
2. list.add("ankit");
3. list.add("nippun");
4. System.out.println(list.size());

### 28) [How to convert ArrayList to Array and Array to ArrayList?](https://www.javatpoint.com/how-to-convert-arraylist-to-array-and-array-to-arraylist-in-java)

We can convert an Array to ArrayList by using the asList() method of Arrays class. asList() method is the static method of Arrays class and accepts the List object. Consider the following syntax:

1. Arrays.asList(item)

We can convert an ArrayList to Array using toArray() method of the ArrayList class. Consider the following syntax to convert the ArrayList to the List object.

1. List\_object.toArray(**new** String[List\_object.size()])

### 29) [How to make Java ArrayList Read-Only?](https://www.javatpoint.com/how-to-make-java-arraylist-read-only)

We can obtain java ArrayList Read-only by calling the Collections.unmodifiableCollection() method. When we define an ArrayList as Read-only then we cannot perform any modification in the collection through  add(), remove() or set() method.

### 30) [How to remove duplicates from ArrayList?](https://www.javatpoint.com/how-to-remove-duplicates-from-arraylist-in-java)

There are two ways to remove duplicates from the ArrayList.

* **Using HashSet:** By using HashSet we can remove the duplicate element from the ArrayList, but it will not then preserve the insertion order.
* **Using LinkedHashSet:** We can also maintain the insertion order by using LinkedHashSet instead of HashSet.

The Process to remove duplicate elements from ArrayList using the LinkedHashSet:

* Copy all the elements of ArrayList to LinkedHashSet.
* Empty the ArrayList using clear() method, which will remove all the elements from the list.
* Now copy all the elements of LinkedHashset to ArrayList.

### 31) [How to reverse ArrayList?](https://www.javatpoint.com/how-to-reverse-arraylist-in-java)

To reverse an ArrayList, we can use reverse() method of Collections class. Consider the following example.

1. **import** java.util.ArrayList;
2. **import** java.util.Collection;
3. **import** java.util.Collections;
4. **import** java.util.Iterator;
5. **import** java.util.List;
6. **public** **class** ReverseArrayList {
7. **public** **static** **void** main(String[] args) {
8. List list = **new** ArrayList<>();
9. list.add(10);
10. list.add(50);
11. list.add(30);
12. Iterator i = list.iterator();
13. System.out.println("printing the list....");
14. **while**(i.hasNext())
15. {
16. System.out.println(i.next());
17. }
18. Iterator i2 = list.iterator();
19. Collections.reverse(list);
20. System.out.println("printing list in reverse order....");
21. **while**(i2.hasNext())
22. {
23. System.out.println(i2.next());
24. }
25. }
26. }

**Output**

printing the list....

10

50

30

printing list in reverse order....

30

50

10

### 32) [How to sort ArrayList in descending order?](https://www.javatpoint.com/how-to-sort-java-arraylist-in-descending-order)

To sort the ArrayList in descending order, we can use the reverseOrder method of Collections class. Consider the following example.

1. **import** java.util.ArrayList;
2. **import** java.util.Collection;
3. **import** java.util.Collections;
4. **import** java.util.Comparator;
5. **import** java.util.Iterator;
6. **import** java.util.List;
8. **public** **class** ReverseArrayList {
9. **public** **static** **void** main(String[] args) {
10. List list = **new** ArrayList<>();
11. list.add(10);
12. list.add(50);
13. list.add(30);
14. list.add(60);
15. list.add(20);
16. list.add(90);
18. Iterator i = list.iterator();
19. System.out.println("printing the list....");
20. **while**(i.hasNext())
21. {
22. System.out.println(i.next());
23. }
25. Comparator cmp = Collections.reverseOrder();
26. Collections.sort(list,cmp);
27. System.out.println("printing list in descending order....");
28. Iterator i2 = list.iterator();
29. **while**(i2.hasNext())
30. {
31. System.out.println(i2.next());
32. }
34. }
35. }

**Output**

printing the list....

10

50

30

60

20

90

printing list in descending order....

90

60

50

30

20

10

### 33) [How to synchronize ArrayList?](https://www.javatpoint.com/how-to-synchronize-arraylist-in-java)

We can synchronize ArrayList in two ways.

* Using Collections.synchronizedList() method
* Using CopyOnWriteArrayList<T>

### 34) [When to use ArrayList and LinkedList?](https://www.javatpoint.com/when-to-use-arraylist-and-linkedlist-in-java)

LinkedLists are better to use for the update operations whereas ArrayLists are better to use for the search operations.

==================================================

**Hibernate:**

### 1) What is hibernate?

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

### 2) What is ORM?

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

### ) What3 are the core interfaces of Hibernate?

The core interfaces of Hibernate framework are:

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

### 4) Mention some of the advantages of using ORM over JDBC.

ORM has the following advantages over JDBC:

* Application development is fast.
* Management of transaction.
* Generates key automatically.
* Details of SQL queries are hidden.

### 5) Define criteria in terms of Hibernate.

The objects of criteria are used for the creation and execution of the object-oriented criteria queries.

### 6) List some of the databases supported by Hibernate.

Some of the databases supported by Hibernate are:

* DB2
* MySQL
* Oracle
* Sybase SQL Server
* Informix Dynamic Server
* HSQL
* PostgreSQL
* FrontBase

### 7) List the key components of Hibernate.

Key components of Hibernate are:

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

### 8) Mention two components of Hibernate configuration object.

Database Connection

Class Mapping Setup

### 10) How is SQL query created in Hibernate?

The SQL query is created with the help of the following syntax:

Session.createSQLQuery

### 9) What does HQL stand for?

Hibernate Query Language

### 10) How is HQL query created?

The HQL query is created with the help of the following syntax:

Session.createQuery

### 11) How can we add criteria to a SQL query?

A criterion is added to a SQL query by using the Session.createCriteria.

### 12) Define persistent classes.

Classes whose objects are stored in a database table are called as persistent classes.

### 13) What is SessionFactory?

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

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

Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

### 15) What is Session?

It maintains a connection between the hibernate application and database.

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

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

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

Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

### 17) What is Session?

It maintains a connection between the hibernate application and database.

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

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

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

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

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

======================================================

**Java 8**

1. What new features did Java 8 introduce?

The latest version has:

* An improved, immutable JodaTime-inspired Date and time API
* A new language called Lambda Expressions that treats actions as objects
* Method References, which enable defining Lambda Expressions by referring to methods directly using their names
* Default methods, which give users the ability to add full implementations in interfaces besides abstract methods
* Nashorn, a high-performance Java-based engine integrated to JDK used to evaluate and execute [JavaScript](https://www.simplilearn.com/reasons-to-learn-javascript-article) code
* Stream API, a special iterator class that allows processing object collections in a functional manner

### 2. Why was a new version of Java needed in the first place?

There are two main reasons:

* Dramatic changes in hardware created the need for Java to use current multi-core CPUs more efficiently
* Enable users to use new Functional Programming (FP) features

### 3. So, what actual advantages does Java 8 bring?

The advantages include:

* [Code](https://www.simplilearn.com/free-and-low-cost-online-resources-for-practicing-code-article) is more concise and readable
* Code is more reusable
* Code is more testable and maintainable
* Code is now both highly concurrent and scalable
* Users can write parallel code
* Users can write database-like operations
* Applications now perform better
* Code is far more productive

### 4. What is a Lambda Expression, and why use them?

It’s a function that can be referenced and shared as an object. [Lambda Expressions](https://www.simplilearn.com/tutorials/java-tutorial/java-lambda-expression) require less[coding](https://www.simplilearn.com/what-is-a-coding-bootcamp-article), provide a means of implementing the Java 8 functional interface, and let users encapsulate one behavior unit to pass around to other code.

### 5. What is a functional interface?

A functional interface is an interface that contains just one abstract method.

### 6. How are functional interfaces and Lambda Expressions related?

Lambda expressions are applied only to the functional interface’s [abstract method.](https://www.simplilearn.com/tutorials/java-tutorial/abstract-class-in-java)

### 7. Can users create a personal functional interface?

Yes, they can

### 8. What does the term “method reference” mean in the context of Java 8?

Method reference is a Java 8 construct used to reference a method without having to invoke it. It is a compact method of Lambda expression.

### 9. What is optional, and what is it best used for?

Optional is a new container class defined in the java.util package, and used to represent optional values that either exist or do not exist. Optional’s chief benefit is avoiding null checks, and there are no longer any “NullPointerException” results at run-time.

### 10. What is Type Inference?

Type inference helps the compiler determine the argument types by looking at each method invocation and corresponding declaration

### 11. List some Java 8 Date and Time API’s

The core API classes are:

* LocalDate
* LocalTime
* LocalDateTime

### 12. Why are default methods needed in the interface?

Default methods let you add new functionality to your libraries’ interfaces and ensure binary compatibility with older code written for the interfaces.

### 13. What is Java 8 StringJoiner class used for?

Java 8 StringJoiner class constructs a sequence of characters separated by a delimiter so that users can create a string by passing delimiters such as hyphens and commas.

### 14. Describe the more commonly found functional interfaces in the standard library.

Although many functional interfaces exist, these are the one's users most likely encounter:

* Function. Takes one argument and returns a result
* Consumer. Takes one argument and returns no result
* Supplier. Takes a not argument and returns a result
* Predicate. Takes one argument and returns a boolean
* BiFunction. Takes two arguments and returns a result
* BinaryOperator. It’s like a BiFunction, except it takes two arguments and returns a result, and they are all the same type
* UnaryOperator. It’s like a Function, but it takes a single argument and returns a result of the same type

### 15. What is a stream, and how does it differ from a collection?

A stream is an iterator whose function is to accept a set of actions and apply them to each of the elements it contains. A stream represents an object sequence from a collection or other source that supports aggregate operations. Unlike collections, iteration logic implements inside the stream.

Also, streams are inherently lazily loaded and processed, unlike collections.

### 16. What is a default method, and when does it get used?

The default method involves an implementation, and it is found in the interface. The method adds new functionalities to an interface while preserving backward compatibility with the classes that already implement the interface

================================================================================

**JPA**

## **1) What is the Java Persistence API?**

The Java Persistence API (JPA) is the specification of Java that is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems. As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation. Therefore, ORM tools like Hibernate, TopLink, and iBatis implements JPA specifications for data persistence. The first version of the Java Persistence API, JPA 1.0 was released in 2006 as a part of EJB 3.0 specification

## **2) Does JPA performs the actual task like access, persist and manage data?**

No, JPA is only a specification. The ORM tools like Hibernate, iBatis, and TopLink implements the JPA specification and perform these type of tasks.

## **3) What is the object-relational mapping?**

The object-relational mapping is a mechanism which is used to develop and maintain a relationship between an object and the relational database by mapping an object state into the database column. It converts attributes of programming code into columns of the table. It is capable of handling various database operations easily such as insertion, updation, deletion, etc.

## **4) What are the advantages of JPA?**

The advantages of JPA are given below.

* The burden of interacting with the database reduces significantly by using JPA.
* The user programming becomes easy by concealing the O/R mapping and database access processing.
* The cost of creating the definition file is reduced by using annotations.
* We can merge the applications used with other JPA providers
* Using different implementations can add the features to the standard Implementation which can later be the part of JPA specification.

## **5) What are the embeddable classes?**

Embeddable classes represent the state of an entity but do not have a persistent identity of their own. The objects of such classes share the identity of the entity classes that owns it. An Entity may have single-valued or multivalued embeddable class attributes.

## **6) List some ORM frameworks.**

Following are the various frameworks that function on ORM mechanism: -

* Hibernate
* TopLink
* ORMLite
* iBATIS
* JPOX

## **7) What is the JPQL?**

JPQL is the Java Persistence query language defined in JPA specification. It is used to construct the queries.

## **8) What are the different directions of entity mapping?**

The direction of a mapping can be either unidirectional or bidirectional. In unidirectional mapping, only one entity can be mapped to another entity, whereas in bidirectional mapping each entity can be mapped or referred to another entity.

## **9) What are the different types of entity mapping?**

Following are the types of object-relational mapping: -

* **One-to-one mapping:** The one-to-one mapping represents a single-valued association where an instance of one entity is associated with an instance of another entity. In this type of association, one instance of source entity can be mapped with at most one instance of the target entity.
* **One-To-Many mapping:** The One-To-Many mapping comes into the category of collection-valued association where an entity is associated with a collection of other entities. In this type of association, the instance of one entity can be mapped with any number of instances of another entity.
* **Many-to-one mapping** The Many-To-One mapping represents a single-valued association where a collection of entities can be associated with the similar entity. In the relational database, more than one row of an entity can refer to the same row of another entity.
* **Many-to-many mapping** The Many-To-Many mapping represents a collection-valued association where any number of entities can be associated with a collection of other entities. In the relational database, more than one row of one entity can refer to more than one row of another entity.

## **10) What type of collections can be used in JPA?**

To store multivalued entity associations and a collection of objects, following types of Java collections is used: -

* List
* Set
* Map

**Spring**

### 1) What is Spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

### 2) What are the advantages of spring framework?

1. Predefined Templates
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. Declarative support

### 3) What are the modules of spring framework?

1. Test
2. Spring Core Container
3. AOP, Aspects and Instrumentation
4. Data Access/Integration
5. Web

### 4) What is IOC and DI?

IOC (Inversion of Control) and DI (Dependency Injection) is a design pattern to provide loose coupling. It removes the dependency from the program.

### 5) What is the role of IOC container in spring?

IOC container is responsible to:

* create the instance
* configure the instance, and
* assemble the dependencies

### 6) What are the types of IOC container in spring?

There are two types of IOC containers in spring framework.

1. BeanFactory
2. ApplicationContext

### 7) What is the difference between BeanFactory and ApplicationContext?

BeanFactory is the **basic container** whereas ApplicationContext is the **advanced container**. ApplicationContext extends the BeanFactory interface. ApplicationContext provides more facilities than BeanFactory.

8) What is the difference between constructor injection and setter injection?

|  |  |
| --- | --- |
| **No.** | **Constructor Injection** |
| 1) | No Partial Injection |
| 2) | Desn't override the setter property |
| 3) | Creates new instance if any modification occurs |
| 4) | Better for too many properties |
| **Setter Injection** | |
| Partial Injection | |
| Overrides the constructor property if both are defined. | |
| Doesn't create new instance if you change the property value | |
| Better for few properties. | |

9) What are the different bean scopes in spring?

There are 5 bean scopes in spring framework.

### 10) In which scenario, you will use singleton and prototype scope?

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

### 11) What are classes for spring JDBC API?

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall

### 12) What is the advantage of NamedParameterJdbcTemplate?

NamedParameterJdbcTemplate class is used to pass value to the named parameter. A named parameter is better than ? (question mark of PreparedStatement).

It is **better to remember**

### 13) What is AOP?

AOP is an acronym for Aspect Oriented Programming. It is a methodology that divides the program logic into pieces or parts or concerns.

It increases the modularity and the key unit is Aspect.

### 14) What are the advantages of spring AOP?

AOP enables you to dynamically add or remove concern before or after the business logic. It is **pluggable** and **easy to maintain**.

**MVC**

### 1) What is MVC?

The MVC (Model-View-Controller) is a software architectural design pattern. It separates the functionality of an application into three interconnected parts - Model, View, and Controller. This approach facilitates the reusability of the code and parallel development.

### 2) What is Spring MVC?

A Spring MVC is a Java Framework which is used to develop dynamic web applications. It implements all the basic features of a core spring framework like Inversion of Control and Dependency Injection. It follows the Model-View-Controller

### 3) What is the front controller of Spring MVC?

The front controller is a **DispatcherServlet** class present in **org.springframework.web.servlet** package. It dispatches the request to the appropriate controller and manages the flow of the application. It is required to specify the **DispatcherServlet** class in the web.xml file.

### 4) What does an additional configuration file contain in Spring MVC application?

The Spring MVC application contains an additional configuration file that contains the properties information. This file can be created either in the form of **an xml** file or **properties** file. In this file, we generally define the base-package and view resolver where **DispatcherServlet** searches for the controller classes and view components path. However, it can also contain various other configuration propertie

### 5) Name the annotations used to handle different types of incoming HTTP request methods?

The following annotations are used to handle different types of incoming HTTP request methods: -

* @GetMapping
* @PostMapping
* @PutMapping
* @PatchMapping
* @DeleteMapping

### 6) What are the ways of reading data from the form in Spring MVC?

The following ways to read the data from the form are: -

* **HttpServletRequest interface** - The **HttpServletRequest** is a java interface present in javax.servlet.http package. Like Servlets, you can use HttpServletRequest in Spring to read the HTML form data provided by the user.
* **@RequestParam annotation** - The **@RequestParam** annotation reads the form data and binds it automatically to the parameter present in the provided method.
* **@ModelAttribute annotation** - The **@ModelAttribute** annotation binds a method parameter or its return value to a named model attribute.

### 7) What do you understand by validations in Spring MVC?

The validation is one of the most important features of Spring MVC, that is used to restrict the input provided by the user. To validate the user's input, it is required to use the Spring 4 or higher version and Bean Validation API. Spring validations can validate both server-side as well as client-side applications

### 8) What is the use of @Valid annotation in Spring MVC?

The **@Valid** annotation is used to apply validation rules on the provided object.

**Rest API**

### 1) What do you understand by RESTful Web Services?

RESTful web services are services that follow REST architecture. REST stands for Representational State Transfer and uses HTTP protocol (web protocol) for implementation. These services are lightweight, provide maintainability, scalability, support communication among multiple applications that are developed using different programming languages. They provide means of accessing resources present at server required for the client via the web browser by means of request headers, request body, response body, status codes, etc.

### 2. What is a REST Resource?

Every content in the REST architecture is considered a resource. The resource is analogous to the object in the object-oriented programming world. They can either be represented as text files, HTML pages, images, or any other dynamic data.

* The REST Server provides access to these resources whereas the REST client consumes (accesses and modifies) these resources. Every resource is identified globally by means of a URI.

### 3. What are the features of RESTful Web Services?

Every RESTful web service has the following features:

* The service is based on the Client-Server model.
* The service uses HTTP Protocol for fetching data/resources, query execution, or any other functions.
* The medium of communication between the client and server is called “Messaging”.
* Resources are accessible to the service by means of URIs.
* It follows the statelessness concept where the client request and response are not dependent on others and thereby provides total assurance of getting the required data.
* These services also use the concept of caching to minimize the server calls for the same type of repeated requests.
* These services can also use SOAP services as implementation protocol to REST architectural pattern.

### 4. What is the concept of statelessness in REST?

The REST architecture is designed in such a way that the client state is not maintained on the server. This is known as statelessness. The context is provided by the client to the server using which the server processes the client’s request. The session on the server is identified by the session identifier sent by the client.

### 5. What do you understand by JAX-RS?

As the name itself stands (JAX-RS= Java API for RESTful Web Services) is a Java-based specification defined by JEE for the implementation of RESTful services. The JAX-RS library makes usage of annotations from Java 5 onwards to simplify the process of web services development. The latest version is 3.0 which was released in June 2020. This specification also provides necessary support to create REST clients.

### 6. What are HTTP Status codes?

These are the standard codes that refer to the predefined status of the task at the server. Following are the status codes formats available:

* 1xx - represents informational responses
* 2xx - represents successful responses
* 3xx - represents redirects
* 4xx - represents client errors
* 5xx - represents server errors
* 200 - success/OK
* 201 - CREATED - used in POST or PUT methods.
* 304 - NOT MODIFIED - used in conditional GET requests to reduce the bandwidth use of the network. Here, the body of the response sent should be empty.
* 400 - BAD REQUEST - This can be due to validation errors or missing input data.
* 401- UNAUTHORIZED - This is returned when there is no valid authentication credentials sent along with the request.
* 403 - FORBIDDEN - sent when the user does not have access (or is forbidden) to the resource.
* 404 - NOT FOUND - Resource method is not available.
* 500 - INTERNAL SERVER ERROR - server threw some exceptions while running the method.
* 502 - BAD GATEWAY - Server was not able to get the response from another upstream server.

### 7.What are the HTTP Methods?

HTTP Methods are also known as HTTP Verbs. They form a major portion of uniform interface restriction followed by the REST that specifies what action has to be followed to get the requested resource. Below are some examples of HTTP Methods:

* GET: This is used for fetching details from the server and is basically a read-only operation.
* POST: This method is used for the creation of new resources on the server.
* PUT: This method is used to update the old/existing resource on the server or to replace the resource.
* DELETE: This method is used to delete the resource on the server.
* PATCH: This is used for modifying the resource on the server.
* OPTIONS: This fetches the list of supported options of resources present on the server.

The POST, GET, PUT, DELETE corresponds to the create, read, update, delete operations which are most commonly called **CRUD Operations**.

### 8.Can you tell the disadvantages of RESTful web services?

The disadvantages are:

* As the services follow the idea of statelessness, it is not possible to maintain sessions. (Session simulation responsibility lies on the client-side to pass the session id)
* REST does not impose security restrictions inherently. It inherits the security measures of the protocols implementing it. Hence, care must be chosen to implement security measures like integrating SSL/TLS based authentications, etc.

### 9. Can you tell what constitutes the core components of HTTP Request?

In REST, any HTTP Request has 5 main components, they are:

* Method/Verb − This part tells what methods the request operation represents. Methods like GET, PUT, POST, DELETE, etc are some examples.
* URI − This part is used for uniquely identifying the resources on the server.
* HTTP Version − This part indicates what version of HTTP protocol you are using. An example can be HTTP v1.1.
* Request Header − This part has the details of the request metadata such as client type, the content format supported, message format, cache settings, etc.
* Request Body − This part represents the actual message content to be sent to the server.

### 10.What makes REST services to be easily scalable?

REST services follow the concept of statelessness which essentially means no storing of any data across the requests on the server. This makes it easier to scale horizontally because the servers need not communicate much with each other while serving requests

### 11.List the key annotations that are present in the JAX-RS API?

* @Path - This specifies the relative URI path to the REST resource.
* @GET - This is a request method designator which is corresponding to the HTTP GET requests. They process GET requests.
* @POST - This is a request method designator which is corresponding to the HTTP POST requests. They process POST requests.
* @PUT - This is a request method designator which is corresponding to the HTTP PUT requests. They process PUT requests.
* @DELETE - This is a request method designator which is corresponding to the HTTP DELETE requests. They process DELETE requests.
* @HEAD - This is a request method designator which is corresponding to the HTTP HEAD requests. They process HEAD requests.
* @PathParam - This is the URI path parameter that helps developers to extract the parameters from the URI and use them in the resource class/methods.
* @QueryParam - This is the URI query parameter that helps developers extract the query parameters from the URI and use them in the resource class/methods.
* @Produces - This specifies what MIME media types of the resource representations are produced and sent to the client as a response.

### 12.What does the annotation @PathVariable do?

@PathVariable annotation is used for passing the parameter with the URL that is required to get the data. Spring MVC provides support for URL customization for data retrieval using @PathVariable annotation.

* **@PostMapping:** It maps the **HTTP POST**requests on the specific handler method. It is used to create a web service endpoint that **creates** It is used instead of using: **@RequestMapping(method = RequestMethod.POST)**
* **@PutMapping:** It maps the **HTTP PUT** requests on the specific handler method. It is used to create a web service endpoint that **creates** or **updates** It is used instead of using: **@RequestMapping(method = RequestMethod.PUT)**
* **@DeleteMapping:** It maps the **HTTP DELETE** requests on the specific handler method. It is used to create a web service endpoint that **deletes**a resource. It is used instead of using: **@RequestMapping(method = RequestMethod.DELETE)**
* **@PatchMapping:** It maps the **HTTP PATCH**requests on the specific handler method. It is used instead of using: **@RequestMapping(method = RequestMethod.PATCH)**
* **@RequestBody:** It is used to **bind** HTTP request with an object in a method parameter. Internally it uses **HTTP MessageConverters** to convert the body of the request. When we annotate a method parameter with **@RequestBody,** the Spring framework binds the incoming HTTP request body to that parameter.
* **@ResponseBody:** It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return an object into JSON and XML format.
* **@PathVariable:** It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple @PathVariable in a method.
* **@RequestParam:** It is used to extract the query parameters form the URL. It is also known as a **query parameter**. It is most suitable for web applications. It can specify default values if the query parameter is not present in the URL.
* **@RequestHeader:** It is used to get the details about the HTTP request headers. We use this annotation as a **method parameter**. The optional elements of the annotation are **name, required, value, defaultValue.**For each detail in the header, we should specify separate annotations. We can use it multiple time in a method
* **@RestController:** It can be considered as a combination of **@Controller** and **@ResponseBody**annotations**.** The @RestController annotation is itself annotated with the @ResponseBody annotation. It eliminates the need for annotating each method with @ResponseBody.
* **@RequestAttribute:** It binds a method parameter to request attribute. It provides convenient access to the request attributes from a controller method. With the help of @RequestAttribute annotation, we can access objects that are populated on the server-side.
* **@GetMapping:** It maps the **HTTP GET** requests on the specific handler method. It is used to create a web service endpoint that **fetches** It is used instead of using: **@RequestMapping(method = RequestMethod.GET)**

================================================================================