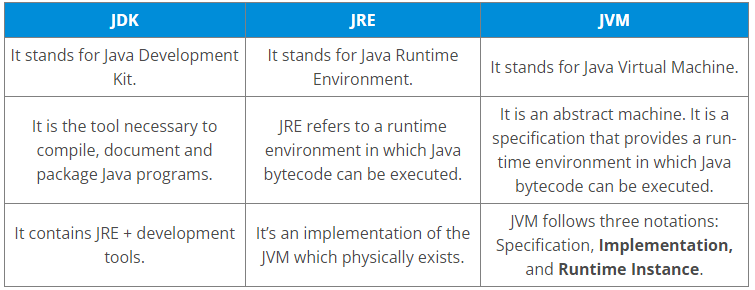
### ****1)BASICS OF JAVA****

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



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

main() in Java is the entry point for any Java program. It is always written as ****public static void main(String[] args)****.

* ****public****: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
* ****static****: It is a keyword in java which identifies it is class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler will throw an error as **main**() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
* ****void****: It is the return type of the method. Void defines the method which will not return any value.
* ****main****: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
* ****String args[]****: It is the parameter passed to the main method.

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

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

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

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

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

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

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

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

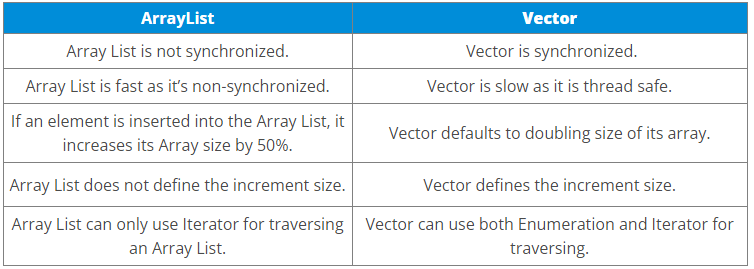
There are two types of constructors:

1. ****Default Constructor:**** In Java, a default constructor is the one which does not take any inputs. In other words, default constructors are the no argument constructors which will be created by default in case you no other constructor is defined by the user. Its main purpose is to initialize the instance variables with the default values. Also, it is majorly used for object creation.
2. ****Parameterized Constructor:**** The parameterized constructor in Java, is the constructor which is capable of initializing the instance variables with the provided values. In other words, the constructors which take the arguments are called parameterized constructors.

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

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

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



### ****Q9. What is the difference between equals() and == in Java?****

Equals() method is defined in Object class in Java and used for checking equality of two objects defined by business logic.

“==” or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. *public boolean equals(Object o)* is the method provided by the Object class. The default implementation uses == operator to compare two objects. For example: method can be overridden like String class. equals() method is used to compare the values of two objects.

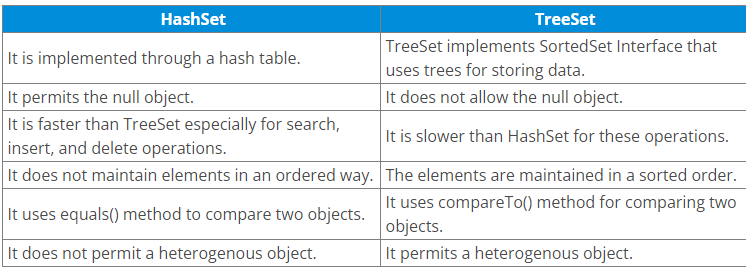
### **Q10. When can you use the super keyword?**

In Java, the super keyword is a reference variable that refers to an immediate parent class object.When you create a subclass instance, you’re also creating an instance of the parent class, which is referenced to by the super reference variable.

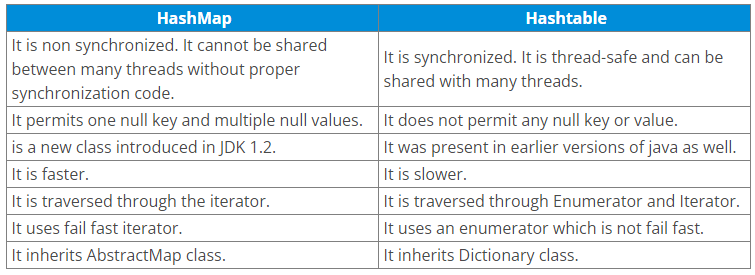
The uses of the Java super Keyword are-

1. To refer to an immediate parent class instance variable, use super.
2. The keyword super can be used to call the method of an immediate parent class.
3. Super() can be used to call the constructor of the immediate parent class.

### **Q11. What makes a HashSet different from a TreeSet?**



****Q12. What are the differences between HashMap and HashTable in Java?****



****Q14. How to not allow serialization of attributes of a class in Java?****

The NonSerialized attribute can be used to prevent member variables from being serialized.  
You should also make an object that potentially contains security-sensitive data nonserializable if possible. Apply the NonSerialized attribute to certain fields that store sensitive data if the object must be serialized. If you don’t exclude these fields from serialisation, the data they store will be visible to any programmes with serialization permission.

****Q15. Can you call a constructor of a class inside another constructor?****

Yes, we can call a constructor of a class inside another constructor. This is also called as constructor chaining. Constructor chaining can be done in 2 ways-

1. ****Within the same class:**** For constructors in the same class, the this() keyword can be used.
2. ****From the base class:**** The super() keyword is used to call the constructor from the base class.  
   The constructor chaining follows the process of inheritance. The constructor of the sub class first calls the constructor of the super class. Due to this, the creation of sub class’s object starts with the initialization of the data members of the super class. The constructor chaining works similarly with any number of classes. Every constructor keeps calling the chain till the top of the chain.

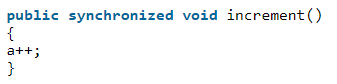
****Q16. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.****

An array generally contains elements of the primitive data types such as int, float, etc. In such cases, the array directly stores these elements at contiguous memory locations. While an ArrayList does not contain primitive data types. An arrayList contains the reference of the objects at different memory locations instead of the object itself. That is why the objects are not stored at contiguous memory locations.

****Q17. How is the creation of a String using new() different from that of a literal?****  
When we create a string using new(), a new object is created. Whereas, if we create a string using the string literal syntax, it may return an already existing object with the same name.

****Q18. Why is synchronization necessary? Explain with the help of a relevant example.****

Java allows multiple threads to execute. They may be accessing the same variable or object. Synchronization helps to execute threads one after another.  
It is important as it helps to execute all concurrent threads while being in sync. It prevents memory consistency errors due to access to shared memory. An example of synchronization code is-



As we have synchronized this function, this thread can only use the object after the previous thread has used it.

****Q19. Explain the term “Double Brace Initialization” in Java?****

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialization. The inner class has a reference to the enclosing outer class, generally using the ‘this’ pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

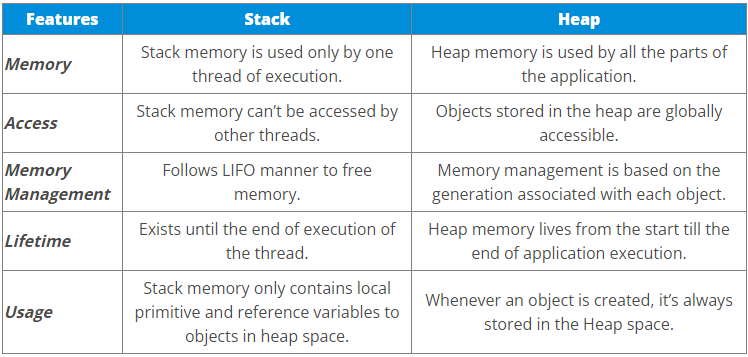
****Q20. Why is it said that the length() method of String class doesn’t return accurate results?****

The length() method of String class doesn’t return accurate results because  
it simply takes into account the number of characters within in the String. In other words, code points outside of the BMP (Basic Multilingual Plane), that is, code points having a value of U+10000 or above, will be ignored.

The reason for this is historical. One of Java’s original goals was to consider all text as Unicode; yet, Unicode did not define code points outside of the BMP at the time. It was too late to modify char by the time Unicode specified such code points.

****Q21. What are the differences between Heap and Stack Memory in Java?****

The major difference between Heap and Stack memory are:



### ****Q22. What is a package in Java? List down various advantages of packages.****

Packages in Java, are the collection of related classes and interfaces which are bundled together. By using packages, developers can easily modularize the code and optimize its reuse. Also, the code within the packages can be imported by other classes and reused. Below I have listed down a few of its advantages:

* Packages help in avoiding name clashes
* They provide easier access control on the code
* Packages can also contain hidden classes which are not visible to the outer classes and only used within the package
* Creates a proper hierarchical structure which makes it easier to locate the related classes

### ****Q23. Why pointers are not used in Java?****

Java doesn’t use pointers because they are unsafe and increases the complexity of the program. Since, Java is known for its simplicity of code, adding the concept of pointers will be contradicting. Moreover, since JVM is responsible for implicit memory allocation, thus in order to avoid direct access to memory by the user,  pointers are discouraged in Java.

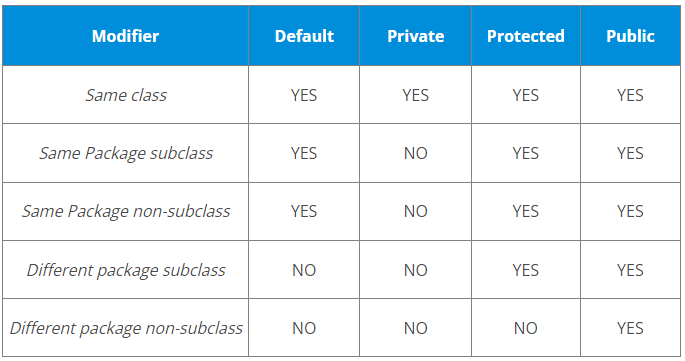
### ****Q24. What is JIT compiler in Java?****

JIT stands for Just-In-Time compiler in Java. It is a program that helps in converting the Java bytecode into instructions that are sent directly to the processor. By default, the JIT compiler is enabled in Java and is activated whenever a Java method is invoked. The JIT compiler then compiles the bytecode of the invoked method into native machine code, compiling it “just in time” to execute. Once the method has been compiled, the JVM summons the compiled code of that method directly rather than interpreting it. This is why it is often responsible for the performance optimization of Java applications at the run time.

### ****Q25. What are access modifiers in Java?****

In Java, access modifiers are special keywords which are used to restrict the access of a class, constructor, data member and method in another class. Java supports four types of access modifiers:

1. Default
2. Private
3. Protected
4. Public



### ****Q26. Define a Java Class.****

A class in Java is a blueprint which includes all your data.  A class contains fields (variables) and methods to describe the behavior of an object. Let’s have a look at the syntax of a class.



### ****Q27. What is an object in Java and how is it created?****

An object is a real-world entity that has a state and behavior. An object has three characteristics:

1. State
2. Behavior
3. Identity

An object is created using the ‘new’ keyword. For example:

ClassName obj = new ClassName();

### ****Q28. What is Object Oriented Programming?****

Object-oriented programming or popularly known as OOPs is a programming model or approach where the programs are organized around objects rather than logic and functions. In other words, OOP mainly focuses on the objects that are required to be manipulated instead of logic. This approach is ideal for the programs large and complex codes and needs to be actively updated or maintained.

### ****Q29. What are the main concepts of OOPs in Java?****

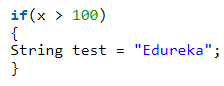
Object-Oriented Programming or OOPs is a programming style that is associated with concepts like:

1. Inheritance: Inheritance is a process where one class acquires the properties of another.
2. Encapsulation: Encapsulation in Java is a mechanism of wrapping up the data and code together as a single unit.
3. Abstraction: Abstraction is the methodology of hiding the implementation details from the user and only providing the functionality to the users.
4. Polymorphism: Polymorphism is the ability of a variable, function or object to take multiple forms.

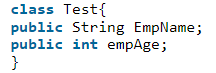
### ****Q30. What is the difference between a local variable and an instance variable?****

In Java, a **local variable** is typically used inside a method, constructor, or a **block** and has only local scope. Thus, this variable can be used only within the scope of a block. The best benefit of having a local variable is that other methods in the class won’t be even aware of that variable.

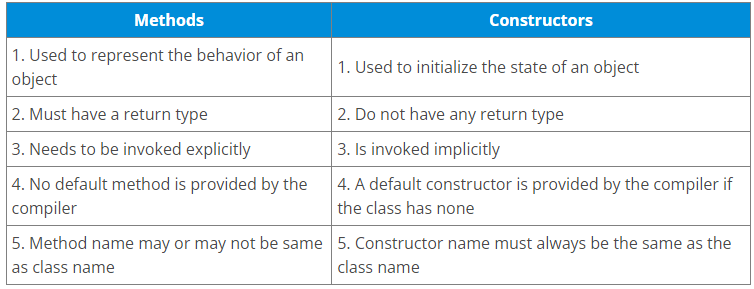
#### Example



Whereas, an **instance variable** in Java, is a variable which is bounded to its object itself. These variables are declared within a **class**, but outside a method. Every object of that class will create it’s own copy of the variable while using it. Thus, any changes made to the variable won’t reflect in any other instances of that class and will be bound to that particular instance only.



### ****Q31. Differentiate between the constructors and methods in Java?****



In case you are facing any challenges with these Core Java interview questions, please comment on your problems in the section below.

### ****Q32. What is final keyword in Java?****

**final**is a special keyword in Java that is used as a non-access modifier. A final variable can be used in different contexts such as:

* ****final variable****

When the final keyword is used with a variable then its value can’t be changed once assigned. In case the no value has been assigned to the final variable then using only the class constructor a value can be assigned to it.

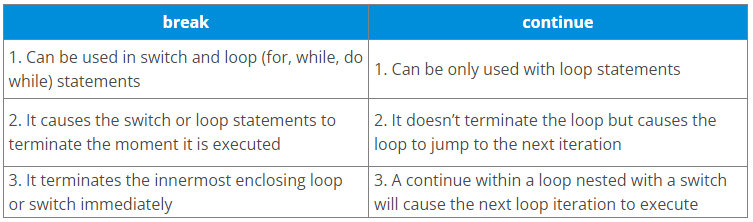
#### ****final method****

When a method is declared final then it can’t be overridden by the inheriting class.

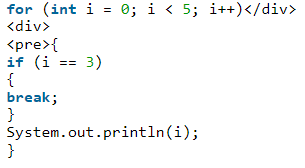
#### ****final class****

When a class is declared as final in Java, it can’t be extended by any subclass class but it can extend other class.

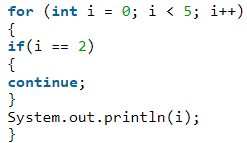
### ****Q33. What is the difference between break and continue statements?****



****Example break:****



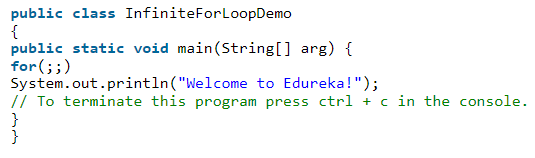
****Example continue:****



### ****Q34. What is an infinite loop in Java? Explain with an example.****

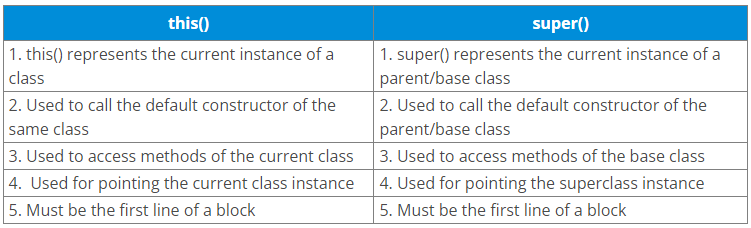
An infinite loop is an instruction sequence in Java that loops endlessly when a functional exit isn’t met. This type of loop can be the result of a programming error or may also be a deliberate action based on the application behavior. An infinite loop will terminate automatically once the application exits.

For example:



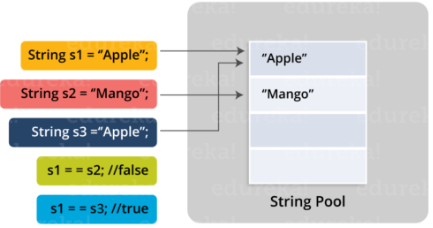
### ****Q35. What is the difference between this() and super() in Java?****

In Java, super() and this(), both are special keywords that are used to call the constructor.

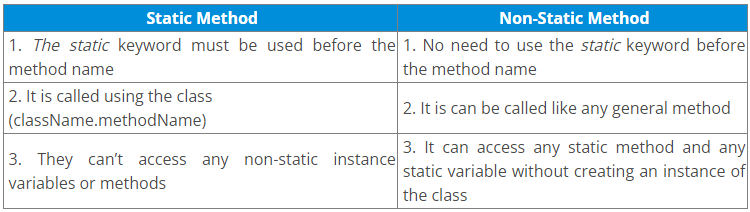


### ****Q36. What is Java String Pool?****

Java String pool refers to a collection of Strings which are stored in heap memory. In this, whenever a new object is created, String pool first checks whether the object is already present in the pool or not. If it is present, then the same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned.

heap

### ****Q37. Differentiate between static and non-static methods in Java.****



### **Q38. Explain the term “Double Brace Initialisation” in Java?**

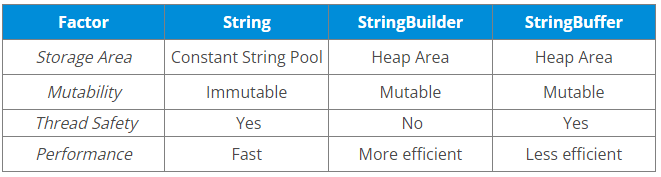
Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialisation. The inner class has a reference to the enclosing outer class, genertally using the ‘this’ pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

### ****Q39. What is constructor chaining in Java?****

In Java, constructor chaining is the process of calling one constructor from another with respect to the current object. Constructor chaining is possible only through legacy where a subclass constructor is responsible for invoking the superclass’ constructor first. There could be any number of classes in the constructor chain. Constructor chaining can be achieved in two ways:

1. Within the same class using this()
2. From base class using super()

****Q40. Difference between String, StringBuilder, and StringBuffer.****



### ****Q41. What is a classloader in Java?****

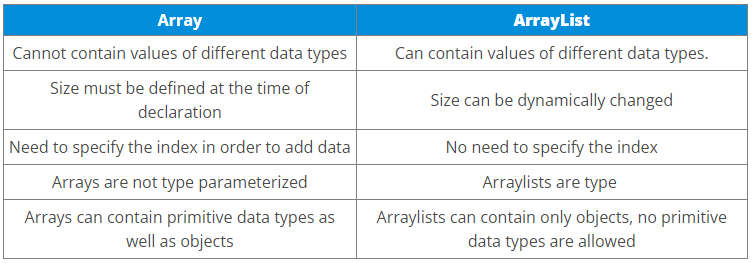
The **Java ClassLoader** is a subset of JVM (Java Virtual Machine) that is responsible for loading the class files. Whenever a Java program is executed it is first loaded by the classloader. Java provides three built-in classloaders:

1. Bootstrap ClassLoader
2. Extension ClassLoader
3. System/Application ClassLoader

### ****Q42. Why Java Strings are immutable in nature?****

In Java, string objects are immutable in nature which simply means once the String object is created its state cannot be modified. Whenever you try to update the value of that object instead of updating the values of that particular object, Java creates a new string object. Java String objects are immutable as String objects are generally cached in the String pool. Since String literals are usually shared between multiple clients, action from one client might affect the rest. It enhances security, caching, synchronization, and performance of the application.

### ****Q43. What is the difference between an array and an array list?****



### ****Q44. What is a Map in Java?****

In Java, Map is an interface of Util package which maps unique keys to values. The Map interface is not a subset of the main Collection interface and thus it behaves little different from the other collection types. Below are a few of the characteristics of Map interface:

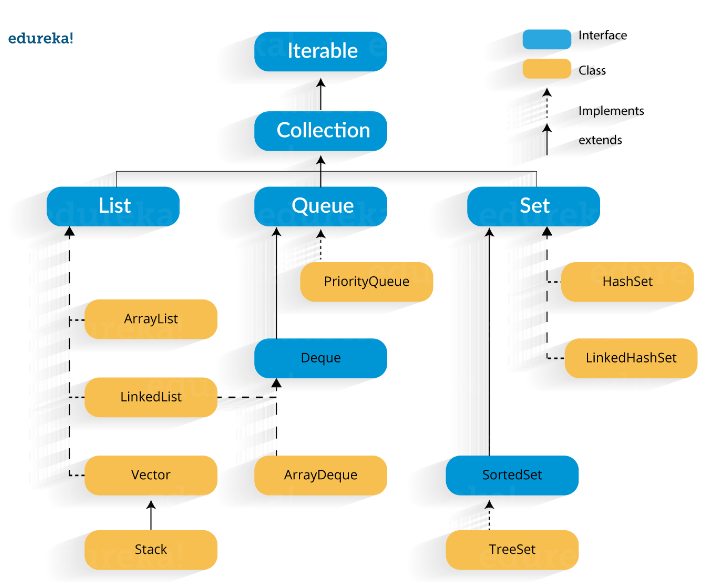
1. Map doesn’t contain duplicate keys.
2. Each key can map at max one value.

### ****Q45. What is collection class in Java? List down its methods and interfaces.****

In Java, the collection is a framework that acts as an architecture for storing and manipulating a group of objects. Using Collections you can perform various tasks like searching, sorting, insertion, manipulation, deletion, etc. Java collection framework includes the following:

* Interfaces
* Classes
* Methods

The below image shows the complete hierarchy of the Java Collection.



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

### ****Q46. What is Polymorphism?****

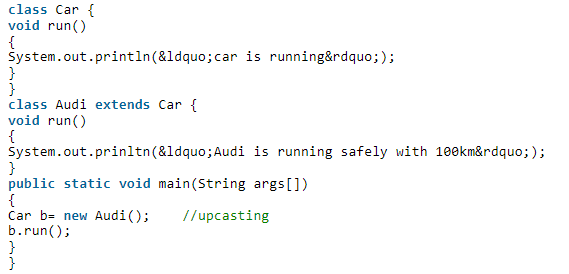
Polymorphism is briefly described as “one interface, many implementations”. Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form. There are two types of polymorphism:

1. Compile time polymorphism
2. Run time polymorphism

Compile time polymorphism is method overloading whereas Runtime time polymorphism is done using inheritance and interface.

### ****Q47. What is runtime polymorphism or dynamic method dispatch?****

In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. Let’s take a look at the example below to understand it better.



### ****Q48. What is abstraction in Java?****

Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user. Thus you can say that abstraction in Java is the process of hiding the implementation details from the user and revealing only the functionality to them. Abstraction can be achieved in two ways:

1. ****Abstract Classes**** (0-100% of abstraction can be achieved)
2. ****Interfaces**** (100% of abstraction can be achieved)

### ****Q49. What do you mean by an interface in Java?****

An interface in Java is a blueprint of a class or you can say it is a collection of abstract methods and static constants. In an interface, each method is public and abstract but it does not contain any constructor. Thus, interface basically is a group of related methods with empty bodies. Example:

public interface Animal {

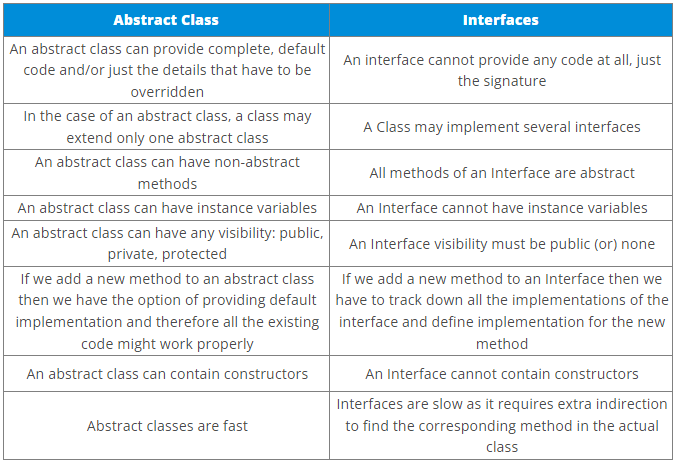
  public void eat();

  public void sleep();

  public void run();

}

### ****Q50. What is the difference between abstract classes and interfaces?****



****Q51. What is inheritance in Java?****

Inheritance in Java is the concept where the properties of one class can be inherited by the other. It helps to reuse the code and establish a relationship between different classes. Inheritance is performed between two types of classes:

1. Parent class (Super or Base class)
2. Child class (Subclass or Derived class)

A class which inherits the properties is known as Child Class whereas a class whose properties are inherited is known as Parent class.

### ****Q52. What are the different types of inheritance in Java?****

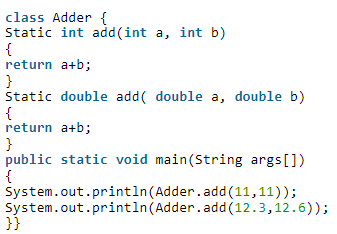
Java supports four types of inheritance which are:

1. ****Single Inheritance:**** In single inheritance, one class inherits the properties of another i.e there will be only one parent as well as one child class.
2. ****Multilevel Inheritance:****When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent class but at different levels, such type of inheritance is called Multilevel Inheritance.
3. ****Hierarchical Inheritance:****When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.
4. ****Hybrid Inheritance:****Hybrid inheritance is a combination of two*or more types* of inheritance.

### ****Q53. What is method overloading and method overriding?****

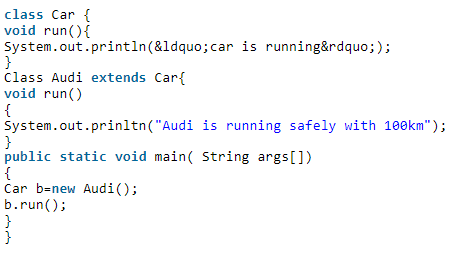
#### ****Method Overloading :****

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



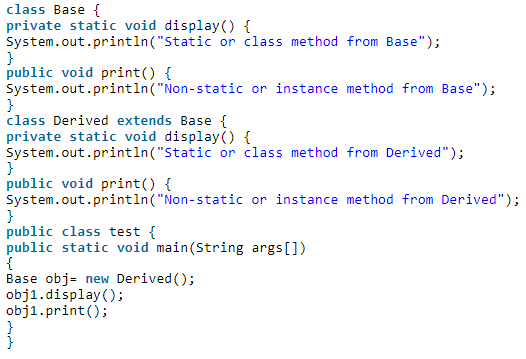
#### ****Method Overriding:****

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



### ****Q54. Can you override a private or static method in Java?****

You cannot override a private or static method in Java. If you create a similar method with the same return type and same method arguments in child class then it will hide the superclass method; this is known as method hiding. Similarly, you cannot override a private method in subclass because it’s not accessible there. What you can do is create another private method with the same name in the child class. Let’s take a look at the example below to understand it better.

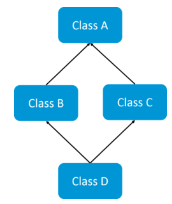


### ****Q55. What is multiple inheritance? Is it supported by Java?****

If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

The problem with multiple inheritance is that if multiple parent classes have the same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.

Therefore, Java doesn’t support multiple inheritance. The problem is commonly referred to as Diamond Problem.



### ****Q56. What is encapsulation in Java?****

Encapsulation is a mechanism where you bind your data(variables) and code(methods) together as a single unit. Here, the data is hidden from the outer world and can be accessed only via current class methods. This helps in protecting the data from any unnecessary modification. We can achieve encapsulation in Java by:

* Declaring the variables of a class as private.
* Providing public setter and getter methods to modify and view the values of the variables.

### ****Q57. What is an association?****

Association is a relationship where all object have their own lifecycle and there is no owner. Let’s take the example of Teacher and Student. Multiple students can associate with a single teacher and a single student can associate with multiple teachers but there is no ownership between the objects and both have their own lifecycle. These relationships can be one to one, one to many, many to one and many to many.

### ****Q58. What do you mean by aggregation?****

An aggregation is a specialized form of Association where all object has their own lifecycle but there is ownership and child object can not belong to another parent object. Let’s take an example of Department and teacher. A single teacher can not belong to multiple departments, but if we delete the department teacher object will not destroy.

### ****Q59. What is composition in Java?****

Composition is again a specialized form of Aggregation and we can call this as a “death” relationship. It is a strong type of Aggregation. Child object does not have their lifecycle and if parent object deletes all child object will also be deleted. Let’s take again an example of a relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different houses if we delete the house room will automatically delete.

****Q60. What is a marker interface?****

A Marker interface can be defined as the interface having no data member and member functions. In simpler terms, an empty interface is called the Marker interface. The most common examples of Marker interface in Java are Serializable, Cloneable etc. The marker interface can be declared as follows.



****Q61. What is object cloning in Java?****

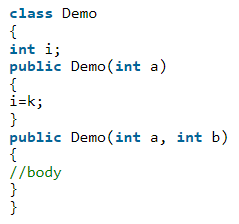
Object cloning in Java is the process of creating an exact copy of an object. It basically means the ability to create an object with a similar state as the original object. To achieve this, Java provides a method ****clone()**** to make use of this functionality. This method creates a new instance of the class of the current object and then initializes all its fields with the exact same contents of corresponding fields. To object clone(), the marker interface ****java.lang.Cloneable****must be implemented to avoid any runtime exceptions. One thing you must note is Object clone() is a protected method, thus you need to override it.

### ****Q62. What is a copy constructor in Java?****

Copy constructor is a member function that is used to initialize an object using another object of the same class. Though there is no need for copy constructor in Java since all objects are passed by reference. Moreover, Java does not even support automatic pass-by-value.

### ****Q63. What is a constructor overloading in Java?****

In Java, constructor overloading is a technique of adding any number of constructors to a class each having a different parameter list. The compiler uses the number of parameters and their types in the list to differentiate the overloaded constructors.



## ****Exception and Thread Java Interview Questions****

### ****Q64. What is the difference between Error and Exception?****

An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors you cannot repair them at runtime. Though error can be caught in the catch block but the execution of application will come to a halt and is not recoverable.

While exceptions are conditions that occur because of bad input or human error etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most of the cases it is possible to recover from an exception (probably by giving the user feedback for entering proper values etc.

### ****Q65. How can you handle Java exceptions?****

There are five keywords used to handle exceptions in Java:

1. try
2. catch
3. finally
4. throw
5. throws

### ****Q66. What are the differences between Checked Exception and Unchecked Exception?****

#### **Checked Exception**

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

#### **Unchecked Exception**

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

### **Q67. What are the different ways of thread usage?**

There are two ways to create a thread:

* Extending Thread class

This creates a thread by creating an instance of a new class that extends the Thread class. The extending class must override the run() function, which is the thread’s entry point.

* Implementing Runnable interface

This is the easiest way to create a thread, by creating a class that implements the runnable interface. After implementing the runnable interface, the class must implement the public void run() method ()

The run() method creates a parallel thread in your programme. When run() returns, the thread will come to an end.

The run() method creates a parallel thread in your programme. When run() returns, the thread will come to an end.

Within the run() method, you must specify the thread’s code.

Like any other method, the run() method can call other methods, use other classes, and define variables.

Java works as “pass by value” or “pass by reference” phenomenon?

Java is always pass-by-value. This means that it creates a copy of the contents of the parameter in memory. In Java, object variables always refer to the memory heap’s real object.

### **Q68. Will the finally block get executed when the return statement is written at the end of try block and catch block as shown below?**

The finally block always gets executed even hen the return statement is written at the end of the try block and the catch block. It always executes , whether there is an exception or not. There are only a few situations in which the finally block does not execute, such as VM crash, power failure, software crash, etc. If you don’t want to execute the finally block, you need to call the System.exit() method explicitly in the finally block.

### **Q69. How does an exception propagate in the code?**

If an exception is not caught, it is thrown from the top of the stack and falls down the call stack to the previous procedure. If the exception isn’t caught there, it falls back to the previous function, and so on, until it’s caught or the call stack reaches the bottom. The term for this is Exception propagation.

### **Q70. Can you explain the Java thread lifecycle?**

The java thread lifecycle has the following states-

**New-**

When a thread is created, and before the program starts the thread, it is in the new state. It is also referred to as a born thread.

**Runnable**

When a thread is started, it is in the Runnable state. In this state, the thread is executing its task.

**Waiting**

Sometimes, a thread goes to the waiting state, where it remains idle because another thread is executing. When the other thread has finished, the waiting thread again comes into the running state.

**Timed Waiting**

In timed waiting, the thread goes to waiting state. But, it remains in waiting state for only a specified interval of time after which it starts executing.It remains waiting either till the time interval ends or till the other thread has finished.

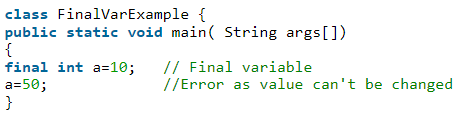
**Terminated**

A thread is said to be in this state once it terminates. It may be because the thread has completed its task or due to any other reason.

### ****Q71. What purpose do the keywords final, finally, and finalize fulfill?****

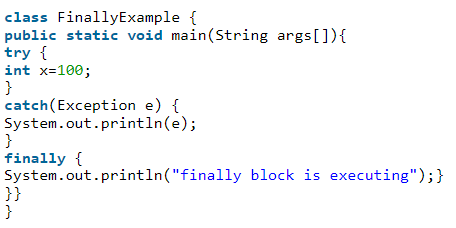
#### ****Final:****

Final is used to apply restrictions on class, method, and variable. A final class can’t be inherited, final method can’t be overridden and final variable value can’t be changed. Let’s take a look at the example below to understand it better.



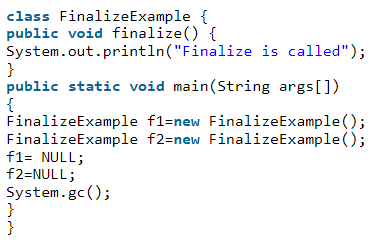
#### ****Finally****

Finally is used to place important code, it will be executed whether the exception is handled or not. Let’s take a look at the example below to understand it better.

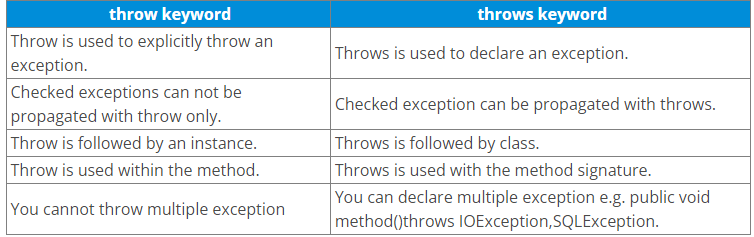


#### ****Finalize****

Finalize is used to perform clean up processing just before the object is garbage collected. Let’s take a look at the example below to understand it better.



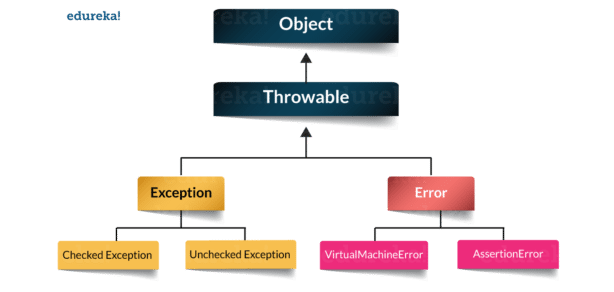
### ****Q72. What are the differences between throw and throws?****



### ****Q73. What is exception hierarchy in java?****

The hierarchy is as follows:

Throwable is a parent class of all Exception classes. There are two types of Exceptions: Checked exceptions and UncheckedExceptions or RunTimeExceptions. Both type of exceptions extends Exception class whereas errors are further classified into Virtual Machine error and Assertion error.



### ****Q74. How to create a custom Exception?****

To create you own exception extend the Exception class or any of its subclasses.

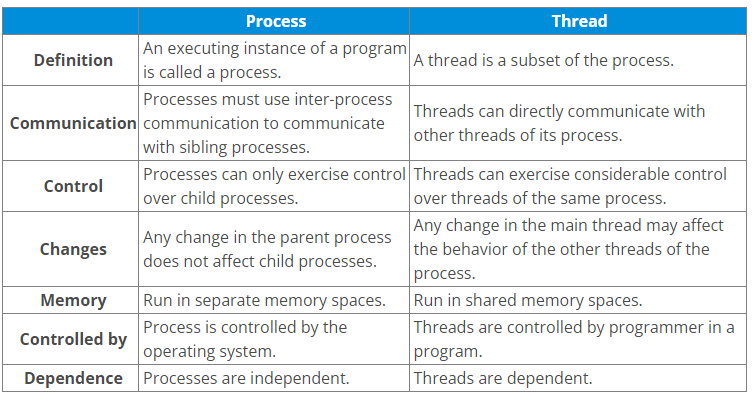
* class New1Exception extends Exception { }               // this will create Checked Exception
* class NewException extends IOException { }             // this will create Checked exception
* class NewException extends NullPonterExcpetion { }  // this will create UnChecked exception

### ****Q75. What are the important methods of Java Exception Class?****

Exception and all of it’s subclasses doesn’t provide any specific methods and all of the methods are defined in the base class Throwable.

1. ****String getMessage()**** – This method returns the message String of Throwable and the message can be provided while creating the exception through it’s constructor.
2. ****String getLocalizedMessage(****) – This method is provided so that subclasses can override it to provide locale specific message to the calling program. Throwable class implementation of this method simply use getMessage() method to return the exception message.
3. ****Synchronized Throwable getCause()**** – This method returns the cause of the exception or null id the cause is unknown.
4. ****String toString()**** – This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.
5. ****void printStackTrace()**** – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as an argument to write the stack trace information to the file or stream.

### ****Q76. What are the differences between processes and threads?****

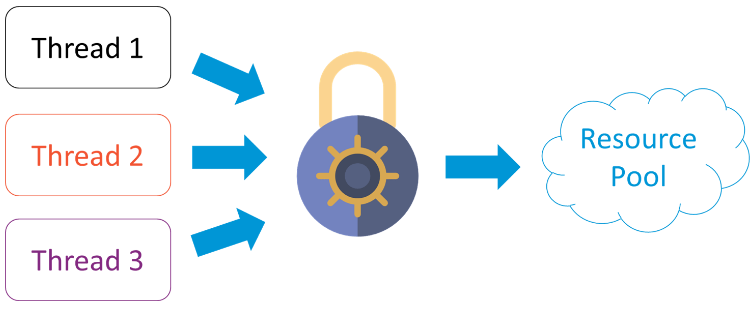


### ****Q77. What is a finally block? Is there a case when finally will not execute?****

Finally block is a block which always executes a set of statements. It is always associated with a try block regardless of any exception that occurs or not.   
Yes, finally will not be executed if the program exits either by calling System.exit() or by causing a fatal error that causes the process to abort.

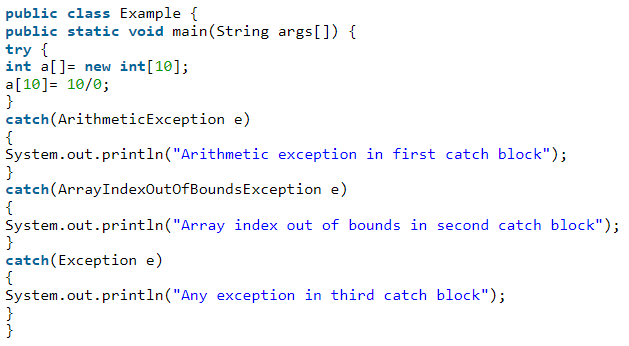
### ****Q78. What is synchronization?****

Synchronization refers to multi-threading. A synchronized block of code can be executed by only one thread at a time. As Java supports execution of multiple threads, two or more threads may access the same fields or objects. Synchronization is a process which keeps all concurrent threads in execution to be in sync. Synchronization avoids memory consistency errors caused due to inconsistent view of shared memory. When a method is declared as synchronized the thread holds the monitor for that method’s object. If another thread is executing the synchronized method the thread is blocked until that thread releases the monitor.



### ****Q79.Can we write multiple catch blocks under single try block?****

Yes we can have multiple catch blocks under single try block but the approach should be from specific to general. Let’s understand this with a programmatic example.



### ****Q80. What are the important methods of Java Exception Class?****

Methods are defined in the base class Throwable. Some of the important methods of Java exception class are stated below.

1. ****String getMessage()**** – This method returns the message String about the exception. The message can be provided through its constructor.
2. **public StackTraceElement[] getStackTrace() –**This method returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack whereas the last element in the array represents the method at the bottom of the call stack.
3. ****Synchronized Throwable getCause()**** – This method returns the cause of the exception or null id as represented by a Throwable object.
4. ****String toString()**** – This method returns the information in String format. The returned String contains the name of Throwable class and localized message.
5. ****void printStackTrace()**** – This method prints the stack trace information to the standard error stream.

### ****Q81. What is OutOfMemoryError in Java?****

OutOfMemoryError is the subclass of java.lang.Error which generally occurs when our JVM runs out of memory.

### ****Q82. What is a Thread?****

A thread is the smallest piece of programmed instructions which can be executed independently by a scheduler. In Java, all the programs will have at least one thread which is known as the main thread. This main thread is created by the JVM when the program starts its execution. The main thread is used to invoke the main() of the program.

### ****Q83. What are the two ways to create a thread?****

In Java, threads can be created in the following two ways:-

* By implementing the Runnable interface.
* By extending the Thread

### ****Q84. What are the different types of garbage collectors in Java?****

Garbage collection in Java a program which helps in implicit memory management. Since in Java, using the new keyword you can create objects dynamically, which once created will consume some memory. Once the job is done and there are no more references left to the object, Java using garbage collection destroys the object and relieves the memory occupied by it. Java provides four types of garbage collectors:

* Serial Garbage Collector
* Parallel Garbage Collector
* CMS Garbage Collector
* G1 Garbage Collector

BASIC OOPS INTERVIEW QUESTIONS

### **85. What is meant by the term OOPs?**

OOPs refers to Object-Oriented Programming. It is the programming paradigm that is defined using objects. Objects can be considered as real-world instances of entities like class, that have some characteristics and behaviors.

### **86. What is the need for OOPs?**

* There are many reasons why OOPs is mostly preferred, but the most important among them are:
* OOPs helps users to understand the software easily, although they don’t know the actual implementation.
* With OOPs, the readability, understandability, and maintainability of the code increase multifold.
* Even very big software can be easily written and managed easily using OOPs.

### **87. What are some major Object Oriented Programming languages?**

The programming languages that use and follow the Object-Oriented Programming paradigm or OOPs, are known as Object-Oriented Programming languages. Some of the major Object-Oriented Programming languages include:

* **[Java](https://www.interviewbit.com/java-interview-questions/" \t "https://www.interviewbit.com/oops-interview-questions/_blank)**
* **[C++](https://www.interviewbit.com/cpp-interview-questions/" \t "https://www.interviewbit.com/oops-interview-questions/_blank)**
* **[Javascript](https://www.interviewbit.com/javascript-interview-questions/" \t "https://www.interviewbit.com/oops-interview-questions/_blank)**
* **[Python](https://www.interviewbit.com/python-interview-questions/" \t "https://www.interviewbit.com/oops-interview-questions/_blank)**
* **[PHP](https://www.interviewbit.com/php-interview-questions/" \t "https://www.interviewbit.com/oops-interview-questions/_blank)**

And many more.

### **88. What are some other programming paradigms other than OOPs?**

Programming paradigms refers to the method of classification of programming languages based on their features. There are mainly two types of Programming Paradigms:

* Imperative Programming Paradigm
* Declarative Programming Paradigm

Now, these paradigms can be further classified based:  
  
****1. Imperative Programming Paradigm****: Imperative programming focuses on HOW to execute program logic and defines control flow as statements that change a program state. This can be further classified as:  
a) Procedural Programming Paradigm: Procedural programming specifies the steps a program must take to reach the desired state, usually read in order from top to bottom.  
b) Object-Oriented Programming or OOP: Object-oriented programming (OOP) organizes programs as objects, that contain some data and have some behavior.  
c) Parallel Programming: Parallel programming paradigm breaks a task into subtasks and focuses on executing them simultaneously at the same time.  
  
****2. Declarative Programming Paradigm****: Declarative programming focuses on WHAT to execute and defines program logic, but not a detailed control flow. Declarative paradigm can be further classified into:  
a) Logical Programming Paradigm: Logical programming paradigm is based on formal logic, which refers to a set of sentences expressing facts and rules about how to solve a problem  
b) Functional Programming Paradigm: Functional programming is a programming paradigm where programs are constructed by applying and composing functions.  
c) Database Programming Paradigm: Database programming model is used to manage data and information structured as fields, records, and files.

### **89. What is meant by Structured Programming?**

****Structured Programming**** refers to the method of programming which consists of a completely structured control flow. Here structure refers to a block, which contains a set of rules, and has a definitive control flow, such as (if/then/else), (while and for), block structures, and subroutines.

Nearly all programming paradigms include Structured programming, including the OOPs model.

### **90. What are the main features of OOPs?**

OOPs or Object Oriented Programming mainly comprises of the below four features, and make sure you don't miss any of these:

* Inheritance
* Encapsulation
* Polymorphism
* Data Abstraction

### **91. What are some advantages of using OOPs?**

* OOPs is very helpful in solving very complex level of problems.
* Highly complex programs can be created, handled, and maintained easily using object-oriented programming.
* OOPs, promote code reuse, thereby reducing redundancy.
* OOPs also helps to hide the unnecessary details with the help of Data Abstraction.
* OOPs, are based on a bottom-up approach, unlike the Structural programming paradigm, which uses a top-down approach.
* Polymorphism offers a lot of flexibility in OOPs.

### **92. Why is OOPs so popular?**

OOPs programming paradigm is considered as a better style of programming. Not only it helps in writing a complex piece of code easily, but it also allows users to handle and maintain them easily as well. Not only that, the main pillar of OOPs - Data Abstraction, Encapsulation, Inheritance, and Polymorphism, makes it easy for programmers to solve complex scenarios. As a result of these, OOPs is so popular.

## **Advanced OOPs Interview Questions**

### **93. What is a class?**

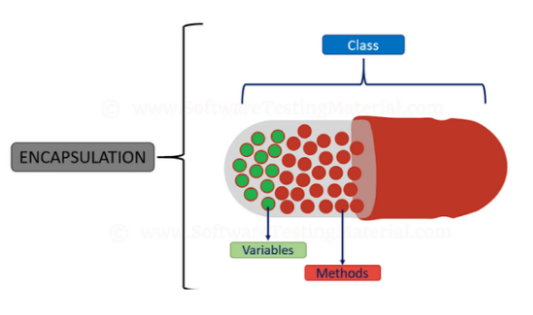
A class can be understood as a template or a blueprint, which contains some values, known as member data or member, and some set of rules, known as behaviors or functions. So when an object is created, it automatically takes the data and functions that are defined in the class.  
Therefore the class is basically a template or blueprint for objects. Also one can create as many objects as they want based on a class.

For example, first, a car’s template is created. Then multiple units of car are created based on that template.

### **94. What is an object?**

An object refers to the instance of the class, which contains the instance of the members and behaviors defined in the class template. In the real world, an object is an actual entity to which a user interacts, whereas class is just the blueprint for that object. So the objects consume space and have some characteristic behavior.  
For example, a specific car.

### **95. What is encapsulation?**

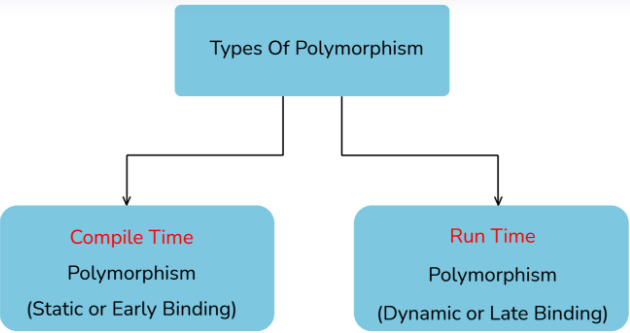


One can visualize Encapsulation as the method of putting everything that is required to do the job, inside a capsule and presenting that capsule to the user. What it means is that by Encapsulation, all the necessary data and methods are bind together and all the unnecessary details are hidden to the normal user. So Encapsulation is the process of binding data members and methods of a program together to do a specific job, without revealing unnecessary details.  
  
Encapsulation can also be defined in two different ways:  
  
1) ****Data hiding:**** Encapsulation is the process of hiding unwanted information, such as restricting access to any member of an object.  
  
2) ****Data binding:**** Encapsulation is the process of binding the data members and the methods together as a whole, as a class.

### **96. What is Polymorphism?**

Polymorphism is composed of two words - “poly” which means “many”, and “morph” which means “shapes”. Therefore Polymorphism refers to something that has many shapes.

### **97.What is Compile time Polymorphism and how is it different from Runtime Polymorphism?**



Compile Time Polymorphism: Compile time polymorphism, also known as Static Polymorphism, refers to the type of Polymorphism that happens at compile time. What it means is that the compiler decides what shape or value has to be taken by the entity in the picture.

Runtime Polymorphism: Runtime polymorphism, also known as Dynamic Polymorphism, refers to the type of Polymorphism that happens at the run time. What it means is it can't be decided by the compiler. Therefore what shape or value has to be taken depends upon the execution. Hence the name Runtime Polymorphism.

### **98. What is meant by Inheritance?**

The term “inheritance” means “receiving some quality or behavior from a parent to an offspring.” In object-oriented programming, inheritance is the mechanism by which an object or class (referred to as a child) is created using the definition of another object or class (referred to as a parent). Inheritance not only helps to keep the implementation simpler but also helps to facilitate code reuse.

### **99. What is Abstraction?**

If you are a user, and you have a problem statement, you don't want to know how the components of the software work, or how it's made. You only want to know how the software solves your problem. Abstraction is the method of hiding unnecessary details from the necessary ones. It is one of the main features of OOPs.   
For example, consider a car. You only need to know how to run a car, and not how the wires are connected inside it. This is obtained using Abstraction.

### **100. How much memory does a class occupy?**

Classes do not consume any memory. They are just a blueprint based on which objects are created. Now when objects are created, they actually initialize the class members and methods and therefore consume memory.

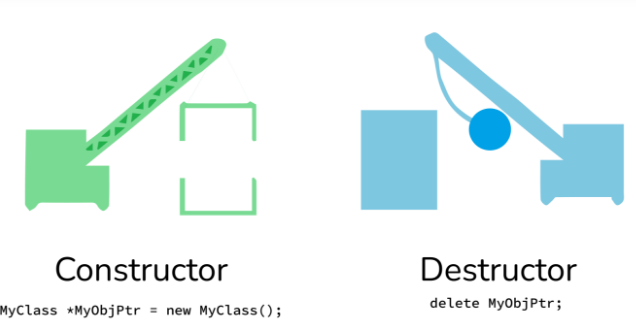
### **101. Is it always necessary to create objects from class?**

No. An object is necessary to be created if the base class has non-static methods. But if the class has static methods, then objects don’t need to be created. You can call the class method directly in this case, using the class name.

### **102. What is a constructor?**

Constructors are special methods whose name is the same as the class name. The constructors serve the special purpose of initializing the objects.  
For example, suppose there is a class with the name “MyClass”, then when you instantiate this class, you pass the syntax:  
MyClass myClassObject = new MyClass();

Now here, the method called after “new” keyword - MyClass(), is the constructor of this class. This will help to instantiate the member data and methods and assign them to the object myClassObject.



### **103. What are the various types of constructors in C++?**

The most common classification of constructors includes:

Default constructor: The default constructor is the constructor which doesn’t take any argument. It has no parameters.

**class** **ABC**

{

**int** x;

ABC()

{

x = 0;

}

}

Parameterized constructor: The constructors that take some arguments are known as parameterized constructors.

**class** **ABC**

{

**int** x;

ABC(**int** y)

{

x = y;

}

}

Copy constructor: A copy constructor is a member function that initializes an object using another object of the same class.

**class** **ABC**

{

**int** x;

ABC(**int** y)

{

x = y;

}

// Copy constructor

ABC(ABC abc)

{

x = abc.x;

}

}

### **104. What is a copy constructor?**

Copy Constructor is a type of constructor, whose purpose is to copy an object to another. What it means is that a copy constructor will clone an object and its values, into another object, is provided that both the objects are of the same class.

### **105. What is a destructor?**

Contrary to constructors, which initialize objects and specify space for them, Destructors are also special methods. But destructors free up the resources and memory occupied by an object. Destructors are automatically called when an object is being destroyed.

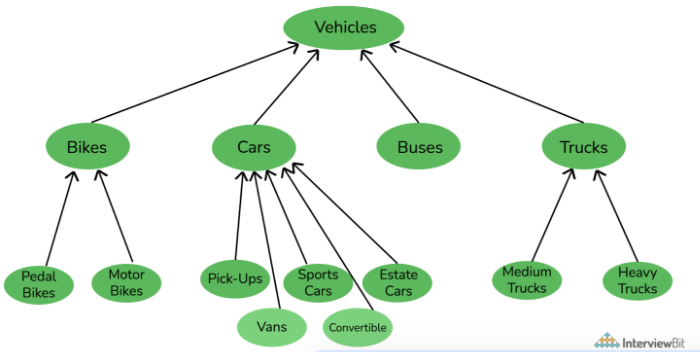
### **106. Are class and structure the same? If not, what's the difference between a class and a structure?**

No, class and structure are not the same. Though they appear to be similar, they have differences that make them apart. For example, the structure is saved in the stack memory, whereas the class is saved in the heap memory. Also, Data Abstraction cannot be achieved with the help of structure, but with class, Abstraction is majorly used.

### **107. Explain Inheritance with an example?**

Inheritance is one of the major features of object-oriented programming, by which an entity inherits some characteristics and behaviors of some other entity and makes them their own. Inheritance helps to improve and facilitate code reuse.

Let me explain to you with a common example. Let's take three different vehicles - a car, truck, or bus. These three are entirely different from one another with their own specific characteristics and behavior. But. in all three, you will find some common elements, like steering wheel, accelerator, clutch, brakes, etc. Though these elements are used in different vehicles, still they have their own features which are common among all vehicles. This is achieved with inheritance. The car, the truck, and the bus have all inherited the features like steering wheel, accelerator, clutch, brakes, etc, and used them as their own. Due to this, they did not have to create these components from scratch, thereby facilitating code reuse.



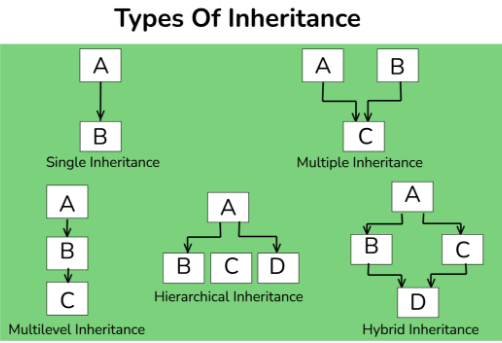
### **108. Are there any limitations of Inheritance?**

Yes, with more powers comes more complications. Inheritance is a very powerful feature in OOPs, but it has some limitations too. Inheritance needs more time to process, as it needs to navigate through multiple classes for its implementation. Also, the classes involved in Inheritance - the base class and the child class, are very tightly coupled together. So if one needs to make some changes, they might need to do nested changes in both classes. Inheritance might be complex for implementation, as well. So if not correctly implemented, this might lead to unexpected errors or incorrect outputs.

### **109. What are the various types of inheritance?**

The various types of inheritance include:

* Single inheritance
* Multiple inheritances
* Multi-level inheritance
* Hierarchical inheritance
* Hybrid inheritance



### **110. What is a subclass?**

The subclass is a part of Inheritance. The subclass is an entity, which inherits from another class. It is also known as the child class.

### **111. Define a superclass?**

Superclass is also a part of Inheritance. The superclass is an entity, which allows subclasses or child classes to inherit from itself.

### **112. What is an interface?**

An interface refers to a special type of class, which contains methods, but not their definition. Only the declaration of methods is allowed inside an interface. To use an interface, you cannot create objects. Instead, you need to implement that interface and define the methods for their implementation.

### **113. What is meant by static polymorphism?**

Static Polymorphism is commonly known as the Compile time polymorphism. Static polymorphism is the feature by which an object is linked with the respective function or operator based on the values during the compile time. Static or Compile time Polymorphism can be achieved through Method overloading or operator overloading.

### **114. What is meant by dynamic polymorphism?**

Dynamic Polymorphism or Runtime polymorphism refers to the type of Polymorphism in OOPs, by which the actual implementation of the function is decided during the runtime or execution. The dynamic or runtime polymorphism can be achieved with the help of method overriding.

### **115. What is the difference between overloading and overriding?**

Overloading is a compile-time polymorphism feature in which an entity has multiple implementations with the same name. For example, Method overloading and Operator overloading.

Whereas Overriding is a runtime polymorphism feature in which an entity has the same name, but its implementation changes during execution. For example, Method overriding.  
****Image****

### **116. How is data abstraction accomplished?**

Data abstraction is accomplished with the help of abstract methods or abstract classes.

### **117. What is an abstract class?**

An abstract class is a special class containing abstract methods. The significance of abstract class is that the abstract methods inside it are not implemented and only declared. So as a result, when a subclass inherits the abstract class and needs to use its abstract methods, they need to define and implement them.

### **118. How is an abstract class different from an interface?**

Interface and abstract class both are special types of classes that contain only the methods declaration and not their implementation. But the interface is entirely different from an abstract class. The main difference between the two is that, when an interface is implemented, the subclass must define all its methods and provide its implementation. Whereas when an abstract class is inherited, the subclass does not need to provide the definition of its abstract method, until and unless the subclass is using it.

Also, an abstract class can contain abstract methods as well as non-abstract methods.

### **119. What are access specifiers and what is their significance?**

Access specifiers, as the name suggests, are a special type of keywords, which are used to control or specify the accessibility of entities like classes, methods, etc. Some of the access specifiers or access modifiers include “private”, “public”, etc. These access specifiers also play a very vital role in achieving Encapsulation - one of the major features of OOPs.

### **120. What is an exception?**

An exception can be considered as a special event, which is raised during the execution of a program at runtime, that brings the execution to a halt. The reason for the exception is mainly due to a position in the program, where the user wants to do something for which the program is not specified, like undesirable input.

### **121. What is meant by exception handling?**

No one wants its software to fail or crash. Exceptions are the major reason for software failure. The exceptions can be handled in the program beforehand and prevent the execution from stopping. This is known as exception handling.  
So exception handling is the mechanism for identifying the undesirable states that the program can reach and specifying the desirable outcomes of such states.  
Try-catch is the most common method used for handling exceptions in the program.

### **122. What is meant by Garbage Collection in OOPs world?**

Object-oriented programming revolves around entities like objects. Each object consumes memory and there can be multiple objects of a class. So if these objects and their memories are not handled properly, then it might lead to certain memory-related errors and the system might fail.

Garbage collection refers to this mechanism of handling the memory in the program. Through garbage collection, the unwanted memory is freed up by removing the objects that are no longer needed.

### **123. Can we run a Java application without implementing the OOPs concept?**

No. Java applications are based on Object-oriented programming models or OOPs concept, and hence they cannot be implemented without it.

However, on the other hand, C++ can be implemented without OOPs, as it also supports the C-like structural programming model.

### **Most Popular Java Programming Interview Questions**

****Q #1) Write a Java Program to reverse a string without using String inbuilt function.****

****Answer:****Here, we are initializing a string variable str and making use of the string builder class.

The object of the string builder class str2 will be further used to append the value stored in the string variable str.  
Thereafter, we are using the inbuilt function of the string builder (reverse()) and storing the new reversed string in str2. Finally, we are printing str2.

****Following program code explains this:****

|  |
| --- |
| **public** **class** FinalReverseWithoutUsingStringMethods {    **public** **static** **void** main(String[] args) {            // TODO Auto-generated method stub            String str = "Automation";            StringBuilder str2 = **new** StringBuilder();            str2.append(str);            str2 = str2.reverse();     // used string builder to reverse           System.out.println(str2);            }    } |

****Output:****

noitamotuA

****Q #2) Write a Java Program to reverse a string without using String inbuilt function reverse().****

****Answer:****There are several ways with which you can reverse your string if you are allowed to use the other string inbuilt functions.

****Method 1:****

In this method, we are initializing a string variable called str with the value of your given string. Then, we are converting that string into a character array with the toCharArray() function. Thereafter, we are using for loop to iterate between each character in reverse order and printing each character.

|  |
| --- |
| **public** **class** FinalReverseWithoutUsingInbuiltFunction {  **public** **static** **void** main(String[] args) {          String str = "Saket Saurav";  **char** chars[] = str.toCharArray();  // converted to character array and printed in reverse order  **for**(**int** i= chars.length-1; i&gt;=0; i--) {              System.out.print(chars[i]);          }      }  } |

****Output:****

varuaS tekaS

****Method 2:****

This is another method in which you are declaring your string variable str and then using Scanner class to declare an object with a predefined standard input object.

This program will accept the string value through the command line (when executed).

We have used nextLine() which will read the input with the spaces between the words of a string. Thereafter, we have used a split() method to split the string into its substrings(no delimiter given here). Finally, we have printed the string in reverse order using for loop.

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** ReverseSplit {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub          String str;          Scanner in = **new** Scanner(System.in);          System.out.println("Enter your String");          str = in.nextLine();          String[] token = str.split("");    //used split method to print in reverse order  **for**(**int** i=token.length-1; i&gt;=0; i--)          {              System.out.print(token[i] + "");          }        }    } |

****Output:****

Enter your String  
Softwaretestinghelp  
plehgnitseterawtfoS

****Method 3:****

This is almost like method 2, but here we did not use the split() method. We have used the scanner class and nextLine() for reading the input string. Then, we have declared an integer length which has the length of the input string.

Thereafter, we have printed the string in the reverse order using for loop. However, we have used the charAt(index) method which will return the character at any specific index. After each iteration, the character will be concatenated to reverse the string variable.

Finally, we have printed the reverse string variable.

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** Reverse {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub          String original, reverse = "";          System.out.println("Enter the string to be reversed");          Scanner in = **new** Scanner(System.in);          original = in.nextLine();  **int** length = original.length();  **for**(**int** i=length-1; i&gt;=0; i--) {              reverse = reverse + original.charAt(i);   //used inbuilt method charAt() to reverse the string          }          System.out.println(reverse);      }    } |

****Output:****

Enter the string to be reversed  
automation testing  
gnitset noitamotua

****Q #3) Write a Java Program to swap two numbers using the third variable.****

****Answer:****In this example, we have made use of the Scanner class to declare an object with a predefined standard input object. This program will accept the values of x and y through the command line (when executed).

We have used nextInt() which will input the value of an integer variable ‘x’ and ‘y’ from the user. A temp variable is also declared.

Now, the logic of the program goes like this – we are assigning temp or third variable with the value of x, and then assigning x with the value of y and again assigning y with the value of temp. So, after the first complete iteration, the temp will have a value of x, x will have a value of y and y will have a value of temp (which is x).

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** SwapTwoNumbers {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub  **int** x, y, temp;          System.out.println("Enter x and y");          Scanner in = **new** Scanner(System.in);          x = in.nextInt();          y = in.nextInt();          System.out.println("Before Swapping" + x + y);          temp = x;          x = y;          y = temp;          System.out.println("After Swapping" + x + y);        }    } |

****Output:****

Enter x and y  
45  
98  
Before Swapping4598  
After Swapping9845

****Q #4) Write a Java Program to swap two numbers without using the third variable.****

****Answer:****Rest all things will be the same as the above program. Only the logic will change. Here, we are assigning x with the value x + y which means x will have a sum of both x and y.

Then, we are assigning y with the value x – y which means we are subtracting the value of y from the sum of (x + y). Till here, x still has the sum of both x and y. But y has the value of x.

Finally, in the third step, we are assigning x with the value x – y which means we are subtracting y (which has the value of x) from the total (x + y). This will assign x with the value of y and vice versa.

|  |
| --- |
| **import** java.util.Scanner;    **class** SwapTwoNumberWithoutThirdVariable  {  **public** **static** **void** main(String args[])     {  **int** x, y;        System.out.println("Enter x and y");        Scanner in = **new** Scanner(System.in);          x = in.nextInt();        y = in.nextInt();          System.out.println("Before Swapping\nx = "+x+"\ny = "+y);          x = x + y;        y = x - y;        x = x - y;          System.out.println("After Swapping without third variable\nx = "+x+"\ny = "+y);     }  } |

****Output:****

Enter x and y  
45  
98  
Before Swapping  
x = 45  
y = 98  
After Swapping without a third variable  
x = 98  
y = 45

****Q #5) Write a Java Program to count the number of words in a string using HashMap.****

****Answer:**** This is a collection class program where we have used HashMap for storing the string.

First of all, we have declared our string variable called str. Then we have used split() function delimited by single space so that we can split multiple words in a string.

Thereafter, we have declared HashMap and iterated using for loop. Inside for loop, we have an if-else statement in which wherever at a particular position, the map contains a key, we set the counter at that position and add the object to the map.

Each time, the counter is incremented by 1. Else, the counter is set to 1.

Finally, we are printing the HashMap.

****Note:**** The same program can be used to count the number of characters in a string. All you need to do is to remove one space (remove space delimited in split method) in String[] split = str.split(“”);

|  |
| --- |
| **import** java.util.HashMap;    **public** **class** FinalCountWords {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub          String str = "This this is is done by Saket Saket";          String[] split = str.split(" ");                    HashMap<String,Integer> map = **new** HashMap<String,Integer>();  **for** (**int** i=0; i<split.length; i++) {  **if** (map.containsKey(split[i])) {  **int** count = map.get(split[i]);                  map.put(split[i], count+1);              }  **else** {                  map.put(split[i], 1);              }          }          System.out.println(map);      }    } |

****Output:****

{Saket=2, by=1, this=1, This=1, is=2, done=1}

****Q #6) Write a Java Program to iterate HashMap using While and advance for loop.****

****Answer:****Here we have inserted three elements in HashMap using put() function.

The size of the map can get using the size() method. Thereafter, we have used a While loop for iterating through the map which contains one key-value pair for each element. Keys and Values can be retrieved through getKey() and getValue().

Likewise, we have used advanced for loop where we have a “me2” object for the HashMap.

|  |
| --- |
| **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.Map;    **public** **class** HashMapIteration {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub          HashMap&lt;Integer,String&gt; map = **new** HashMap&lt;Integer,String&gt;();          map.put(2, "Saket");          map.put(25, "Saurav");          map.put(12, "HashMap");          System.out.println(map.size());          System.out.println("While Loop:");          Iterator itr = map.entrySet().iterator();  **while**(itr.hasNext()) {              Map.Entry me = (Map.Entry) itr.next();              System.out.println("Key is " + me.getKey() + " Value is " + me.getValue());          }          System.out.println("For Loop:");  **for**(Map.Entry me2: map.entrySet()) {              System.out.println("Key is: " + me2.getKey() + " Value is: " + me2.getValue());          }      }    } |

****Output:****

3  
While Loop:  
Key is 2 Value is Saket  
Key is 25 Value is Saurav  
Key is 12 Value is HashMap  
For Loop:  
Key is: 2 Value is: Saket  
Key is: 25 Value is: Saurav  
Key is: 12 Value is: HashMap

****Q #7) Write a Java Program to find whether a number is prime or not.****

****Answer:****Here, we have declared two integers temp and num and used Scanner class with nextInt(as we have integer only).

One boolean variable isPrime is set to true. Thereafter, we have used for loop starting from 2, less than half of the number is entered and incremented by 1 for each iteration. Temp will have the remainder for every iteration. If the remainder is 0, then isPrime will be set to False.

Based on isPrime value, we are coming to the conclusion that whether our number is prime or not.

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** Prime {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub  **int** temp, num;  **boolean** isPrime = **true**;          Scanner in = **new** Scanner(System.in);          num = in.nextInt();          in.close();  **for** (**int** i = 2; i&lt;= num/2; i++) {              temp = num%i;  **if** (temp == 0) {                  isPrime = **false**;  **break**;              }          }  **if**(isPrime)              System.out.println(num + "number is prime");  **else**                  System.out.println(num + "number is not a prime");          }    } |

****Output:****

445  
445number is not a prime

****Q #8) Write a Java Program to find whether a string or number is palindrome or not.****

****Answer:****You can use any of the reverse string program explained above to check whether the number or string is palindrome or not.

What you need to do is to include one if-else statement. If the original string is equal to a reversed string then the number is a palindrome, otherwise not.

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** Palindrome {  **public** **static** **void** main (String[] args) {          String original, reverse = "";          Scanner in = **new** Scanner(System.in);  **int** length;          System.out.println("Enter the number or String");          original = in.nextLine();          length = original.length();  **for** (**int** i =length -1; i>;=0; i--) {              reverse = reverse + original.charAt(i);          }          System.out.println("reverse is:" +reverse);    **if**(original.equals(reverse))              System.out.println("The number is palindrome");  **else**              System.out.println("The number is not a palindrome");        }  } |

****Output:****

****For String-****

Enter the number or String  
vijay  
reverse is:yajiv  
The number is not a palindrome

****For Number-****

Enter the number or String  
99  
reverse is:99  
The number is palindrome

****Q #9) Write a Java Program for the Fibonacci series.****

****Answer:****Fibonacci series is a series of numbers where after the initial two numbers, every occurring number is the sum of two preceding numbers.

****For Example**** 0,1,1,2,3,5,8,13,21………

In this program, we have used Scanner class again with nextInt (discussed above). Initially, we are entering (through command line) the number of times the Fibonacci has to iterate. We have declared integer num and initialized a,b with zero and c with one. Then, we have used for loop to iterate.

The logic goes like a is set with the value of b which is 0, then b is set with the value of c which is 1. Then, c is set with the sum of both a and b.

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** Fibonacci {  **public** **static** **void** main(String[] args) {  **int** num, a = 0,b=0, c =1;          Scanner in = **new** Scanner(System.in);          System.out.println("Enter the number of times");          num = in.nextInt();          System.out.println("Fibonacci Series of the number is:");  **for** (**int** i=0; i<num; i++) {              a = b;              b = c;              c = a+b;              System.out.println(a + "");    //if you want to print on the same line, use print()          }      }  } |

****Output:****

Enter the number of times  
10  
Fibonacci Series of the number is:  
0  
1  
1  
2  
3  
5  
8  
13  
21  
34

****Q #10) Write a Java Program to iterate ArrayList using for-loop, while-loop, and advance for-loop.****

****Answer:**** In this program, we have inserted three elements and printed the size of the ArrayList.

Then, we have used While Loop with an iterator. Whenever the iterator has (next) element, it will display that element until we reach the end of the list. So it will iterate three times.

Likewise, we have done for Advanced For Loop where we have created an object called obj for the ArrayList called list. Then printed the object.

Thereafter, we have put the condition of For Loop where the iterator i is set to 0 index, then it is incremented by 1 until the ArrayList limit or size is reached. Finally, we have printed each element using a get(index) method for each iteration of For Loop.

|  |
| --- |
| **import** java.util.\*;    **public** **class** arrayList {  **public** **static** **void** main(String[] args) {          ArrayList list = **new** ArrayList();          list.add("20");          list.add("30");          list.add("40");          System.out.println(list.size());          System.out.println("While Loop:");          Iterator itr = list.iterator();  **while**(itr.hasNext()) {              System.out.println(itr.next());          }          System.out.println("Advanced For Loop:");  **for**(Object obj : list) {              System.out.println(obj);      }          System.out.println("For Loop:");  **for**(**int** i=0; i&lt;list.size(); i++) {              System.out.println(list.get(i));          }  }  } |

****Output:****

3  
While Loop:  
20  
30  
40  
Advanced For Loop:  
20  
30  
40  
For Loop:  
20  
30  
40

****Q #11) Write a Java Program to demonstrate an explicit wait condition check.****

****Answer:****There are two main types of wait – implicit and explicit. (We are not considering Fluent wait in this program)

|  |
| --- |
| WebDriverWait wait = **new** WebDriverWait(driver, 20);  WebElement element2 = wait.until(ExpectedConditions.visibilityOfElementLocated(By.partialLinkText("Software testing - Wikipedia")));  element2.click(); |

In the above piece of code, you can see that we have created an object wait for WebDriverWait and then we have searched for WebElement called element2.

The condition is set in such a way that the webdriver will have to wait until we see the link “Software testing – Wikipedia” on a web page. It won’t execute if it does not find this link. If it does, then it will do a mouse click on that link.

|  |
| --- |
| **package** Codes;  **import** java.util.concurrent.TimeUnit;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.openqa.selenium.chrome.ChromeOptions;  **import** org.openqa.selenium.support.ui.ExpectedConditions;  **import** org.openqa.selenium.support.ui.WebDriverWait;    **public** **class** explicitWaitConditionCheck {    **public** **static** **void** main(String[] args) {                      // TODO Auto-generated method stub                      System.setProperty("webdriver.chrome.driver", "C:\\webdriver\\chromedriver.exe");                     ChromeOptions options = **new** ChromeOptions();                     options.addArguments("--disable-arguments");                     WebDriver driver = **new** ChromeDriver();                     driver.manage().window().maximize();                     driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);                     driver.navigate().to("https://www.google.com");                     WebElement element = driver.findElement(By.name("q"));                      element.sendKeys("Testing");                      element.submit();                      WebDriverWait wait = **new** WebDriverWait(driver, 20);                         WebElement element2 = wait.until(ExpectedConditions.visibilityOfElementLocated(By.partialLinkText("Software testing - Wikipedia")));  element2.click();  }} |

****Q #12) Write a Java Program to demonstrate Scroll up/ Scroll down.****

****Answer:**** All the lines of codes are easily relatable as we have discussed in our previous example.

However, in this program, we have included our JavascriptExecutor js which will do the scrolling. If you see the last line of the code, we have passed window.scrollBy(arg1,arg2).

If you want to scroll up then pass some value in arg1 if you want to scroll down then pass some value in arg2.

|  |
| --- |
| **package** Codes;    **import** java.util.concurrent.TimeUnit;    **import** org.openqa.selenium.By;  **import** org.openqa.selenium.JavascriptExecutor;  **import** org.openqa.selenium.Keys;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;    **public** **class** ScrollDown {    **public** **static** **void** main(String[] args) {                     // TODO Auto-generated method stub                     System.setProperty("webdriver.chrome.driver", "C:\\webdriver\\chromedriver.exe");                     WebDriver driver = **new** ChromeDriver();                     JavascriptExecutor js = (JavascriptExecutor) driver;                     driver.manage().window().maximize();                     driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);                     driver.get("https://www.google.com");                     WebElement element = driver.findElement(By.name("q"));                     element.sendKeys("SoftwareTestingHelp");                     element.sendKeys(Keys.ENTER);                      js.executeScript("window.scrollBy(0,1000)");    }    } |

****Q #13) Write a Java Program to open all links of gmail.com.****

****Answer:**** It is a typical example of advanced for loop which we have seen in our previous programs.

Once you have opened a website such as Gmail using get() or navigate().to(), you can use a tagName locator to find the tag name of a website that will return all the tags.

We have advanced for loop where we have created a new WebElement link2 for a link(which already has located all the tags), then we have got all the links through getAttribute(“href”) and got all the texts through getText().

|  |
| --- |
| **package** Codes;    **import** java.util.concurrent.TimeUnit;    **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;    **public** **class** openAllLinks {    **public** **static** **void** main(String[] args) {          // TODO Auto-generated method stub          System.setProperty("webdriver.chrome.drive", "C:\\webdriver\\chromedriver.exe");          WebDriver driver = **new** ChromeDriver();          driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);          driver.manage().window().maximize();          driver.get("https://www.gmail.com/");          java.util.List&lt;WebElement&gt; link = driver.findElements(By.tagName("a"));         System.out.println(link.size());    **for** (WebElement link2: link) {           //print the links i.e. http://google.com or https://www.gmail.com        System.out.println(link2.getAttribute("href"));          //print the links text       System.out.println(link2.getText());  }  }  } |

****Output:****

Starting ChromeDriver 2.38.551601 (edb21f07fc70e9027c746edd3201443e011a61ed) on port 16163  
Only local connections are allowed.  
4  
https://support.google.com/chrome/answer/6130773?hl=en-GB  
Learn more  
https://support.google.com/accounts?hl=en-GB  
Help  
https://accounts.google.com/TOS?loc=IN&hl=en-GB&privacy=true  
Privacy  
https://accounts.google.com/TOS?loc=IN&hl=en-GB  
Terms

****Q #14) Write a Selenium code to switch to the previous tab.****

****Answer:**** We have demonstrated the use of the Robot class. We see this as an important third party because we can achieve the different navigation within a browser and its tabs if you know the shortcut keys.

****For example****, if you have three tabs open in your chrome and you want to go to the middle tab, then you have to press control + 2 from your keyboard. The same thing can be achieved through the code as well.

Observe the following code (just after we see the instantiation of Robot class). we have used the Robot class object called a robot with two inbuilt methods keyPress(KeyEvenet.VK\_\*) and keyRelease(KeyEvenet.VK\_\*).

|  |
| --- |
| **package** Codes;    **import** java.awt.AWTException;  **import** java.awt.Robot;  **import** java.awt.event.KeyEvent;  **import** java.util.concurrent.TimeUnit;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.Keys;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **public** **class** PreviousTab {  **public** **static** **void** main(String[] args) **throws** AWTException {                 // TODO Auto-generated method stub                System.setProperty("webdriver.chrome.driver", "C:\\webdriver\\chromedriver.exe");               WebDriver driver = **new** ChromeDriver();               driver.manage().window().maximize();               driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);               driver.get("https://www.google.com");               WebElement element1 = driver.findElement(By.name("q"));               element1.sendKeys("software testing help");               element1.sendKeys(Keys.ENTER);               String a = Keys.chord(Keys.CONTROL,Keys.RETURN);               driver.findElement(By.partialLinkText("Software Testing Help - A Must Visit Software Testing Portal")).sendKeys(a);               Robot robot = **new** Robot(); // instantiated robot class               robot.keyPress(KeyEvent.VK\_CONTROL); // with robot class you can easily achieve anything if you know the shortcut keys               robot.keyPress(KeyEvent.VK\_2); // here, we have just pressed ctrl+2               robot.keyRelease(KeyEvent.VK\_CONTROL); // once we press and release ctrl+2, it will go to the second tab.               robot.keyRelease(KeyEvent.VK\_2); //if you again want to go back to first tab press and release vk\_1               }  } |

****Q #15) Write a Java Program to find the duplicate characters in a string.****

****Answer:****In this program, we have created a string variable str and initialized an integer count with zero.

Then, we have created a character array to convert our string variable to the character. With the help of for loop, we are performing a comparison between different characters at different indexes.

If two characters of consecutive index match, then it will print that character and the counter will be incremented by 1 after each iteration.

|  |
| --- |
| **public** **class** DuplicateCharacters {    **public** **static** **void** main(String[] args) {                     // TODO Auto-generated method stub                    String str = **new** String("Sakkett");  **int** count = 0;  **char**[] chars = str.toCharArray();                    System.out.println("Duplicate characters are:");  **for** (**int** i=0; i&lt;str.length();i++) {  **for**(**int** j=i+1; j&lt;str.length();j++) {  **if** (chars[i] == chars[j]) {                                                      System.out.println(chars[j]);                                                      count++;  **break**;                                            }                                 }                     }             }    } |

****Output:****

Duplicate characters are:  
k  
t

****Q #16) Write a Java Program to find the second-highest number in an array.****

****Answer:**** In this program, we have initialized an array with 10 random elements out of which we are going to find the second-highest number. Here, we have two integers- the largest and second-largest. Both set to the first index of the element. Then, we have printed all the elements using for loop.

Now, the logic is when the element at the 0th index is greater than the largest, then assign arr[0] to largest and secondLargest to largest. Again, if the element at the 0th index is greater than the secondLargest, then assign secondLargest to arr[0].

This will be repeated for each iteration and ultimately after comparing or completing iterations up to array length will give you the secondLargest element.

|  |
| --- |
| **package** codes;  **public** **class** SecondHighestNumberInArray {  **public** **static** **void** main(String[] args)      {  **int** arr[] = { 100,14, 46, 47, 94, 94, 52, 86, 36, 94, 89 };  **int** largest = 0;  **int** secondLargest = 0;          System.out.println("The given array is:");  **for** (**int** i = 0; i < arr.length; i++)          {              System.out.print(arr[i] + "\t");          }  **for** (**int** i = 0; i < arr.length; i++)          {  **if** (arr[i] > largest)              {                  secondLargest = largest;                  largest = arr[i];              }  **else** **if** (arr[i] > secondLargest)              {                  secondLargest = arr[i];              }          }          System.out.println("\nSecond largest number is:" + secondLargest);          System.out.println("Largest Number is: "  +largest);      }  } |

****Output:****

The given array is:  
100 14 46 47 94 94 52 86 36 94 89  
Second largest number is:94  
Largest Number is: 100

****Q #17) Write a Java Program to check Armstrong number.****

****Answer:**** First of all we need to understand what Armstrong Number is. Armstrong number is the number which is the sum of the cubes of all its unit, tens and hundred digits for three-digit numbers.

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3 = 1 + 125 + 27 = 153

If you have a four-digit number lets say

1634 = 1\*1\*1\*1 + 6\*6\*6\*6 + 3\*3\*3\*3 + 4\*4\*4\*4 = 1 + 1296 + 81 + 256 = 1634

Now, in this program, we have a temp and integers declared. We have initialized c with value 0. Then, we need to assign the integer value which we are going to check for Armstrong (in our case, let us say 153). Then we have assigned our temp variable with that number which we are going to check.

Thereafter, we have used while conditional check where the remainder is assigned to a and the number is divided by 10 and assigned to n. Now, our c variable which was set to zero initially is assigned with c+(a\*a\*a). Suppose we have to evaluate a four-digit number then c should be assigned with c + (a\*a\*a\*a).

Lastly, we have put an if-else statement for conditional checking where we have compared the value contained in c against temp(which has the actual number stored at this point). If it matches, then the number is Armstrong otherwise not.

|  |
| --- |
| **class** Armstrong{  **public** **static** **void** main(String[] args)  {  **int** c=0,a,temp;  **int** n=153;//It is the number to check Armstrong     temp=n;  **while**(n&gt;0)     {     a=n%10;     n=n/10;      c=c+(a\*a\*a);      }  **if**(temp==c)      System.out.println("armstrong number");  **else**          System.out.println("Not armstrong number");     }  } |

****Output:****  
armstrong number

****Q #18) Write a Java Program to remove all white spaces from a string with using replace().****

****Answer:**** This is a simple program where we have our string variable str1.

Another string variable str2 is initialized with the replaceAll option which is an inbuilt method to remove n number of whitespaces. Ultimately, we have printed str2 which has no whitespaces.

|  |
| --- |
| **class** RemoveWhiteSpaces  {  **public** **static** **void** main(String[] args)      {          String str1 = "Saket Saurav        is a QualityAna    list";            //1. Using replaceAll() Method            String str2 = str1.replaceAll("\\s", "");            System.out.println(str2);               }  }  } |

****Output:****

SaketSauravisaQualityAnalist

****Q #19) Write a Java Program to remove all white spaces from a string without using replace().****

****Answer:****This is another approach to removing all the white spaces. Again, we have one string variable str1 with some value. Then, we have converted that string into a character array using toCharArray().

Then, we have one StringBuffer object sb which will be used to append the value stored at chars[i] index after we have included for loop and one if condition.

If the condition is set such that then the element at i index of the character array should not be equal to space or tab. Finally, we have printed our StringBuffer object sb.

|  |
| --- |
| **class** RemoveWhiteSpaces  {  **public** **static** **void** main(String[] args)      {          String str1 = "Saket Saurav        is an Autom ation Engi ne      er";    **char**[] chars = str1.toCharArray();            StringBuffer sb = **new** StringBuffer();    **for** (**int** i = 0; i &lt; chars.length; i++)          {  **if**( (chars[i] != ' ') &amp;&amp; (chars[i] != '\t') )              {                  sb.append(chars[i]);              }          }          System.out.println(sb);           //Output : CoreJavajspservletsjdbcstrutshibernatespring      }  } |

****Output:****

SaketSauravisanAutomationEngineer

****Q #20)**** ****Write a Java Program to read an excel.****

****Answer:****These types of programs are generally used in Selenium framework. We have added detailed comments for every step to make the program more understandable.

The logic starts after we have loaded the sheet in which the data is stored. We are trying to import email and password. For this, we are retrieving the cell using getRow() and getCell() method. Let’s say we have email and passwords at the 1st and 2nd cell.

Then we are setting the type of cell to string. Thereafter we are carrying out a normal web element locator operation (By.id) where we have passed unique locator values such as “email” and “password” which will identify these elements.

Finally, we are sending keys using element.sendKeys where cell.getStringCellValue() is the key. This will return you the value stored at cell number 1 and 2 respectively.

|  |
| --- |
| @Test  **public** **void** ReadData() **throws** IOException   {       // Import excel sheet from a webdriver directory which is inside c drive.       //DataSource is the name of the excel       File src=**new** File("C:\\webdriver\\DataSource.xls");         //This step is for loading the file. We have used FileInputStream as       //we are reading the excel. In case you want to write into the file,       //you need to use FileOutputStream. The path of the file is passed as an argument to FileInputStream       FileInputStream finput = **new** FileInputStream(src);         //This step is to load the workbook of the excel which is done by global HSSFWorkbook in which we have       //passed finput as an argument.      workbook = **new** HSSFWorkbook(finput);         //This step is to load the sheet in which data is stored.       sheet= workbook.getSheetAt(0);    **for**(**int** i=1; i&lt;=sheet.getLastRowNum(); i++)       {           // Import data for Email.           cell = sheet.getRow(i).getCell(1);           cell.setCellType(Cell.CELL\_TYPE\_STRING);           driver.findElement(By.id("email")).sendKeys(cell.getStringCellValue());             // Import data for the password.           cell = sheet.getRow(i).getCell(2);           cell.setCellType(Cell.CELL\_TYPE\_STRING);           driver.findElement(By.id("password")).sendKeys(cell.getStringCellValue());            }    } |