**CORE JAVA**

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

JVM (Java Virtual Machine): JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a Java code. JVM is a part of JRE(Java Runtime Environment).

JRE (Java Runtime Environment): JRE refers to a runtime environment in which Java bytecode can be executed. It implements the JVM (Java Virtual Machine) and provides all the class libraries and other support files that JVM uses at runtime. So JRE is a software package that contains what is required to run a Java program. Basically, it’s an implementation of the JVM which physically exists.

JDK(Java Development Kit): It is the tool necessary to compile, document and package Java programs. The JDK completely includes JRE which contains tools for Java programmers. The Java Development Kit is provided free of charge. Along with JRE, it includes an interpreter/loader, a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development. In short, it contains JRE + development tools.

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

Public: Public is an access modifier. Public means that this Method will be accessible by any Class.

static : It is a keyword in java which identifies it is class-based i.e it can be accessed without creating the instance of a Class. Since we want the main method to be executed without any instance also, we use static.

Void: It is the return type of the method. Void defines the method which will not return any value.

main: This is the first method executed by JVM. The signature of the method must be the same.

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

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

**Q4. Why is Java not pure Object-oriented?**

Java is not considered pure Object-oriented because it supports primitive data-types such as boolean, byte, char, int, float, double, long, short.

**Q5. Define class and object. Explain them with an example using java.**

Class: A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

Superclass: The name of the class’s parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.

Interfaces: A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.

Object: It is a basic unit of Object Oriented Programming and represents the real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

State : It is represented by attributes of an object. It also reflects the properties of an object.

Behavior : It is represented by methods of an object. It also reflects the response of an object with other objects.

Identity : It gives a unique name to an object and enables one object to interact with other objects.

For Example: Employee is an example of a class

**Q6.What is a method? Provide several signatures of the methods**   
A Java method is a set of statements to perform a task. A method is placed in a class.   
Signatures of methods: The name of the method, return type and the number of parameters comprise the method signature.   
A method can have the following elements in its signature:   
– Access specifier – public, private, protected, etc.   
– Access modifier – static, synchronized, etc.   
– Return type – void, int, String, etc.   
– Method name – show()  
– With or without parameters – (int number, String name);

**Q7.Explain the difference between instance variable and a class variable.**   
An instance variable is a variable which has one copy per object/instance. That means every object will have one copy of it.   
A class variable is a variable which has one copy per class. The class variables will not have a copy in the object.

**Q9.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.   
If a class does not explicitly declare any, the Java compiler automatically provides a no-argument constructor, also called the default constructor.   
This default constructor calls the class parent’s no-argument constructor (as it contains only one statement i.e. super();), or the Object class constructor if the class has no other parent (as Object class is a parent of all classes either directly or indirectly).   
There are two types of constructors:

1. Default constructor
2. Parameterized constructor

**Q10. What are the different ways to create objects in Java?**   
There are many different ways to create objects in Java.

1. Using new keyword
2. Using new instance
3. Using clone() method
4. Using deserialization
5. Using newInstance() method of Constructor class

**Q11. What’s the purpose of Static methods and static variables?**   
When there is a requirement to share a method or a variable between multiple objects of a class instead of creating separate copies for each object, we use static keyword to make a method or variable shared for all objects.   
**Static variable:** Static variables are also known as Class variables.   
These variables are declared similarly as instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.   
Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create.   
Static variables are created at the start of program execution and destroyed automatically when execution ends.   
To access static variables, we need not create an object of that class.   
Static methods: A static method can be accessed without creating objects. Just by using the Class name the method can be accessed. The static method can only access static variables and not local or global non-static variables.

**Q12. Why static methods cannot access non-static variables or methods?**   
Ans) A static method cannot access non-static variables or methods because static methods can be accessed without instantiating the class, so if the class is not instantiated the variables are not initialized and thus cannot be accessed from a static method.

**Q13.What is a static class?**   
A class can be said to be static class if all the variables and methods of the class are static and the constructor is private. Making the constructor private will prevent the class to be instantiated. So the only possibility to access is using the Class name only.

**Q14. How many types of Variable? Explain.**There are three types of variables in Java:

1. Local Variables
2. Instance Variables
3. Static Variables

**Local Variables:** A variable defined within a block or method or constructor is called local variable.   
These variable are created when the block is entered or the function is called and destroyed after exiting from the block or when the call returns from the function.   
The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variables only within that block. 

**Instance Variables:** Instance variables are non-static variables and are declared in a class outside any method, constructor or block.   
As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed.   
Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier then the default access specifier will be used. 

**Static variable:** Static variables are also known as Class variables.

* These variables are declared similarly as instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.
* Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create.
* Static variables are created at start of program execution and destroyed automatically when execution ends.

To access static variables, we need not create an object of that class, we can simply access the variable

**Q15. List the features of Java Programming language.**

There are the following features in Java Programming Language.

Simple: Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it

**Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior.

**Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.

**Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which needs a platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed.

**Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secured.

**Robust**: Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust.

**Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java.

Interpreted: Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution.

**Q17.As a language, Java is considered platform-independent. Why?**

Java is considered to be platform-independent because it does not depend on any hardware or software. When the Java programmer compiles Java code, the compiler converts the code into bytecode that can be run on different platforms. The only constraint is that the platform must have JRE (runtime environment) or Java Virtual Machine (JVM) installed on it.

**Q18.What is data-encapsulation in Java?**

Data-encapsulation in Java is the process of hiding the variables (data) and the code applied to the data in a single unit. It is a significant feature of object-oriented programming. This helps Java developers to isolate different objects from each other and to create modules for each. By this, all objects would have their own behaviors, attributes, and functions. It prevents data or object mixing, hides data, enhances security.

**Q19.What are wrapper classes in Java?**

Java, as a programming language, supports primitive data types. Wrapper classes help convert these primitive data types into objects or reference types and vice versa. Thus, the wrapper classes are so named because they ‘wrap’ these data types into identifiable objects of the specific data classes the primitive data types belong to.

**Q20.Does Java use pointers? If not, why?**

No, unlike C++, Java doesn’t use pointers. Java’s main focus is simplicity. The use of pointers complicates the process, especially for new java programmers. The use of pointers can also lead to possible errors. Moreover, pointers grant direct access to memory and thus, compromise security. By excluding pointers from Java, a certain amount of abstraction is achieved. Java has automatic Garbage Collectors, the use of pointers can slow down the garbage collection process.

**Q21. What is the difference between an Inner Class and a Sub-Class?**

Ans: An Inner class is a class which is nested within another class. An Inner class has access rights for the class which is nesting it and it can access all variables and methods defined in the outer class.

A sub-class is a class which inherits from another class called super class. Sub-class can access all public and protected methods and fields of its super class.

**Q22. What is the difference between continue and break statement?**

Ans: break and continue are two important keywords used in Loops. When a break keyword is used in a loop, loop is broken instantly while when continue keyword is used, current iteration is broken and loop continues with next iteration.

**Q23. What’s the difference between an Abstract Class and Interface in Java?**

Ans: The primary difference between an abstract class and interface is that an interface can only possess declaration of public static methods with no concrete implementation while an abstract class can have members with any access specifiers (public, private etc) with or without concrete implementation.

Another key difference in the use of abstract classes and interfaces is that a class which implements an interface must implement all the methods of the interface while a class which inherits from an abstract class doesn’t require implementation of all the methods of its super class.

A class can implement multiple interfaces but it can extend only one abstract class.