Java Questions and Answers

What are Class and Objects in Java?

* We often find many individual objects all of the same kind. There may be thousands of other bicycles in existence, all of the same make and model. Each bicycle was built from the same set of blueprints and therefore contains the same components. In object-oriented terms, we say that your bicycle is an *instance* of the *class of objects* known as bicycles. A *class* is the blueprint from which individual objects are created.

Why java is platform Independent?

* Java is platform-independent **because it uses a virtual machine**. The Java programming language and all APIs are compiled into bytecodes. Bytecodes are effectively platform-independent. The virtual machine takes care of the differences between the bytecodes for the different platforms.

Define basic principle of OOP?

* Abstraction
* Encapsulation
* Inheritance
* Polymorphism

What is JDK, JRE, JIT, JVM?

* **JDK** (Java Development Kit) is a Kit that provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) that includes two things

1. Development Tools(to provide an environment to develop your java programs)
2. JRE

* **2. JRE** (Java Runtime Environment) is an installation package that provides an environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.
* **JVM** (**Java Virtual Machine)**is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an **i*nterpreter*.**
* **JIT** is the part of the Java Virtual Machine (JVM) that is used to speed up the execution time. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.

What is an Interface?

* An *interface* is a reference type, similar to a class, that can contain *only* constants, method signatures, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods. Interfaces cannot be instantiated—they can only be *implemented* by classes or *extended* by other interfaces.

What is an abstract class?

* An abstract class is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.
* An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon).

What are the different types of memory in Java?

* Class(Method) Area : The class method area is the memory block that stores the class code, variable code(static variable, runtime constant), method code, and the constructor of a Java program.
* Heap : The Heap area is the memory block where objects are created or objects are stored.
* Stack : Each thread has a private JVM stack, created at the same time as the thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.
* Program Counter Register : Each JVM thread that carries out the task of a specific method has a program counter register associated with it.
* Native Method Stack : Also called C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when it’s created and it can be of a fixed or dynamic nature.

What do you mean by Constructor?

* A *constructor* is used in the creation of an object that is an instance of a class. Typically, it performs operations required to initialize the class before methods are invoked or fields are accessed. Constructors are never inherited.

What is Inheritance?

* A class that is derived from another class is called a subclass (also a derived class, extended class, or child class). The class from which the subclass is derived is called a superclass (also a base class or a parent class).
* Excepting Object, which has no superclass, every class has one and only one direct superclass (single inheritance). In the absence of any other explicit superclass, every class is implicitly a subclass of Object.
* Classes can be derived from classes that are derived from classes that are derived from classes, and so on, and ultimately derived from the topmost class, Object. Such a class is said to be descended from all the classes in the inheritance chain stretching back to Object.

What is Abstraction?

* Data Abstraction is defined as **the process of reducing the object to its essence so that only the necessary characteristics are exposed to the users**. Abstraction defines an object in terms of its properties (attributes), behavior (methods), and interfaces (means of communicating with other objects).

What is Encapsulation?

* Encapsulation is all about hiding implementation details of a given type. The narrow understanding of this principle is making instance variables private and providing public methods to operate on the variables. But there is also a wider understanding that any implementation details such as internal mechanics and logic implemented within a class is encapsulated, behind a "facade" of its public methods. Which means that an invoker of a method does not need to know this methods implementation details. Eg: Getters and Setters.

Relation between Abstraction and Encapsulation?

* the objects that result in encapsulation need not be abstracted. The objects that help to perform abstraction are encapsulated.

What is Polymorphism?

* Polymorphism means "**many forms**", and it occurs when we have many classes that are related to each other by inheritance.

What are the types of Polymorphism?

* Compile time Polymorphism: Method Overloading:- Same method name but different method signature.
* Runtime Polymorphism: Method Overriding :- Same method of the parent class with a different method body.