FAQ's on OOPs Concepts (OO Thinking)

1. What is Object Oriented Programming?

OOP is a method of programming in which the programs are organized as cooperative collections of objects. Each object is an instance of a class and each class belong to a hierarchy.

Object oriented programming organizes a program around its data, i. e. the objects and a set of well-defined interfaces to that data. An object-oriented program can be characterized as data controlling access to code.

2. What are Objects?

Everything in Java is an object. An object is a collection of data and actions that make up a programming entity. A 'car' object, for example, might have some data (i.e. 'speed,' 'direction,' 'lights on,' 'current fuel,' etc.) and some actions it can take ('turn right,' 'turn lights on,' 'accelerate,' etc.)

3. What are the basic principles of OOPS?

The basic principles of Object Oriented Programming Structure (OOPS) are:

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

4. What is Encapsulation?

Encapsulation is the technique to wrap-up data and its associated methods into a single unit.

5. What is Inheritance?

Inheritance is a technique to make one object acquiring the properties of another object. This supports the hierarchical classification. An object can inherit its general attributes from its parent. A new sub-class inherits all of the attributes of all of its ancestors.

- A class that is inherited is called super class.
- The class that inherits properties from its super class is called a subclass.
- Inheritance is done by using the keyword *extends*.

- The two most common reasons to use inheritance are: To promote code reuse and to use polymorphism.

6. What is Polymorphism?

Polymorphism is a feature that allows a single interface to be used for the general class of actions. The specific action is determined by the exact nature of the situation. In general, polymorphism means one interface multiple methods. Polymorphism allows us to design a generic interface to a group of related activities. This helps in reducing complexity by allowing the same interface to be used to specify a general class of action. It is the compiler's job to select the specific action (that is the method) as it applies to each situation.

7. What is Abstraction?

Abstraction is an essential element of Object Oriented Programming which manages the complexity.
Definition: It Exposing the necessary / essential details and common characteristics, postponing other details.

8. How does Java implements polymorphism?

Polymorphism manifests itself in Java in the form of multiple methods having the same name:

- In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods).
- In other cases, multiple methods have the same name, same return type, and same formal argument list (overridden methods).

9. What are the different forms of Polymorphism?

There are two forms of polymorphism: **Compile-time polymorphism** and **Run-time polymorphism**. Compile-time polymorphism is also called method overloading. Run-time polymorphism is done by using inheritance and interface.

Note: From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:

- Method overloading,
- Method overriding through inheritance and
- Method overriding through the Java interface.

10. What is the difference between Abstraction and Encapsulation?

The following are the differences between Abstraction and Encapsulation:

- Abstraction focuses on the outside view of an object (i.e. the interface), whereas encapsulation (information hiding) prevents the client from seeing the view where the behavior of the abstraction is implemented.
- Abstraction solves the problem in the design side while Encapsulation is the implementation.
- Encapsulation is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs.

11. What is an Instance?

An instance is also called as an object. It has a state, behavior and identity. The structure and behavior of similar classes are defined in their common class.

12. What is a Base Class?

The base class is the most generalized class in a class structure. Most applications have such root classes. In Java, the object is the base class for all the classes.

13. What is a Subclass?

The Subclass is a class that inherits from one or more classes.

14. What is a Superclass?

The Superclass is a class from which another class inherits its properties.

15. What is a Constructor?

The constructor is an operation that creates an object and/or initializes its state.

Constructor declarations look like method declarations, except that they use the name of the class and have no return type. For example, the class *Bicycle* has one constructor:

```
public Bicycle(int startCadence, int startSpeed, int
startGear) {
   gear = startGear;
   cadence = startCadence;
```

```
            speed = startSpeed;
        }
```

16. Does Java support Destructor?

In Java, there is no concept of destructors. It is taken care by the JVM using Garbage Collection.

17. What is meant by Binding?

Binding denotes the association of a name with a class.

18. What is Static Binding?

The Static Binding is a binding in which the class association is made during compile time. This is also called as **Early Binding**.

19. What is Dynamic Binding?

The Dynamic Binding is a binding in which the class association is not made until the object is created at execution time. It is also called as **Late Binding**.

20. Define Modularity?

The Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.

21. What is Collaboration?

Collaboration is a process whereby several objects cooperate to provide some higher level behaviour.

22. In Java, How can objects are completely encapsulated?

In Java, to completely encapsulate an object completely, all the instance variables of its should be declared as private, and the public getter and setter methods should be provided for accessing the instance variables.

23. What is the difference between Aggregation and Composition?

The following are the differences between Aggregation and Composition:

- **Aggregation** is an association in which one class belongs to a collection. This is a part of a whole relationship where a part can exist without a whole. For example a line item is a whole and product is a part. If a line

item is deleted then corresponding product need not be deleted. So aggregation has a weaker relationship.

- **Composition** is an association in which one class belongs to a collection. This is a part of a whole relationship where a part cannot exist without a whole. If a whole is deleted then all parts are deleted. For example, an order is a whole and line items are parts. If an order is deleted, then all its corresponding line items should also be deleted. So composition has a stronger relationship.

24. What is a Java Program?

A Java program is mostly a collection of objects talking to other objects by invoking each other's methods. Every object is of a certain type, and that type is defined by a class or an interface. Most Java programs use a collection of objects of many different types.

- **Class:** A template that describes the kinds of state and behavior that the objects of its type support.
- **Object:** At runtime, when the Java Virtual Machine (JVM) encounters the new keyword, it will use the appropriate class to make an object which is an instance of that class. That object will have its own state, and access to all of the behaviors defined by its class.
- **State (instance variables):** Each object (instance of a class) will have its own unique set of instance variables as defined in the class. Collectively, the values assigned to the object's instance variables make up the object's statement.
- **Behavior (methods):** When a programmer creates a class, she creates methods for that class. Methods are where the class' logic is stored. Methods are where the real work gets done. They are where algorithms get executed, and data gets manipulated.

25. What is the most important feature of Java?

Java is a platform independent language.

26. What do you mean by platform independence?

Platform independence means that we can write and compile the Java code in one platform (e.g. Windows) and can execute the class in any other supported platform (e.g. Linux, Solaris, etc.)

27. What is a JVM?

JVM is Java Virtual Machine which is a run time environment for the compiled Java class files.

28. Is JVM's platform independent?

JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.

29. What is the difference between a JDK and a JVM?

JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

30. What is a pointer and does Java support pointers?

A pointer is a reference handle to a memory location. Improper handling of pointers leads to memory leaks and reliability issues hence Java doesn't support the usage of pointers.

31. Is Java a pure object oriented language?

Java uses primitive data types and hence is not a pure object oriented language.

32. Are arrays primitive data types?

In Java, Arrays are objects.

33. What is the difference between Path and Classpath?

Path and Classpath are operating system level environment variables. The path is used to define where the system can find the executable (. exe) files and classpath is used to specify the location .class files.

34. Should a main() method be compulsorily declared in all Java classes?

No not required. The main() method should be defined only if the source class is a Java application.

35. What is the return type of the main() method?

The main() method doesn't return anything hence declared void.

36. Why is the main() method declared static?

The main() method is called by the JVM even before the instantiation of the class hence it is declared as static.

37. What is the argument of the main() method?

The main() method accepts an array of String objects as argument.

38. Can a main() method be overloaded?

Yes. You can have any number of main() methods with different method signature and implementation in the class.

39. Can a main() method be declared final?

Yes. Any inheriting class will not be able to have its own default main() method.

40. Does the order of public and static declaration matter in the main() method?

No. It doesn't matter but void should always come before main().

41. What is a native method?

A native method is a method that is implemented in a language other than Java.

42. What are order of precedence and associativity, and how are they used?

Order of precedence determines the order in which operators are evaluated in expressions. Associativity determines whether an expression is evaluated left-to-right or right-to-left.

43. Why isn't there operator overloading?

Because C++ has proven by example that operator overloading makes code almost impossible to maintain.

44. Is null a keyword?

The null value is not a keyword.

45. Which characters may be used as the second character of an identifier, but not as the first character of an identifier?

The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier.

46. How does the ternary operator be written; x: y? z or x ? y: z ?

The ternary is written x? y: z.

47. How is rounding performed under integer division?

Rounding is performed by truncating the fractional part of the result. This is also known as rounding toward zero.

48. What restrictions are placed on the values of each case of a switch statement?

During compilation, the values of each case of a switch statement must evaluate to a value that can be promoted to an int value.

49. What is the difference between a while statement and a do while statement?

A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do while statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do while statement will always execute the body of a loop at least once.

50. What is the difference between the prefix and postfix forms of the ++ operator?

The prefix form performs the increment operation and returns the value of the increment operation. The postfix form returns the current value all of the expression and then performs the increment operation on that value.

51. What is the difference between an 'if' statement and a switch statement?

The '**if'** statement uses a boolean expression to decide which alternative should be executed first. The switch statement is used to select among multiple alternatives. It uses an int expression to determine which alternative should be executed.

52. What is the difference between procedural and object-oriented programs?

The following are the difference between procedural and object-oriented programs:

- In procedural programming, the logic follows certain procedures and the instructions are executed one after another, whereas In OOP program, unit of program is an object, which is nothing but combination of data and code.

- In procedural programming, data is exposed to the whole program whereas in OOPs program, it is accessible within the object and which in turn assures the security of the code.