**Phase I**

1. **What is OOPS?**

Object Oriented Programming System is the programming technique to write programs based on the real world objects. The states and behaviors of an object are represented as the member variables and methods. In OOPS programming programs are organized around objects and data rather than actions and logic.

1. **What are the advantages of OOPS concepts?**

Major advantages of OOPS programming are;

Simplicity: OOPS programming objects model real world objects, so the complexity is reduced and the program structure is clear.

Modularity: Each object forms a separate entity whose internal workings are decoupled from other parts of the system.

Modifiability: It is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.

Extensibility: Adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.

Maintainability: Objects can be maintained separately, making locating and fixing problems easier.

Reusability: Objects can be reused in different programs.

1. **What is the difference between Procedural programming and OOPS?**

Procedural language is based on functions but object oriented language is based on real world objects.

Procedural language gives importance on the sequence of function execution but object oriented language gives importance on states and behaviors of the objects.

Procedural language exposes the data to the entire program but object oriented language encapsulates the data.

Procedural language follows top down programming paradigm but object oriented language follows bottom up programming paradigm.

Procedural language is complex in nature so it is difficult to modify, extend and maintain but object oriented language is less complex in nature so it is easier to modify, extend and maintain.

Procedural language provides less scope of code reuse but object oriented language provides more scope of code reuse.

1. **What are the core concepts of OOPS?**

OOPS core concepts are;

* Abstraction
* Encapsulation
* Polymorphism
* Inheritance
* Composition
* Association
* Aggregation

1. **What is Abstraction?**

Abstraction is an OOPS concept to construct the structure of the real world objects. During this construction only the general states and behaviors are taken and more specific states and behaviors are left aside for the implementers.

1. **What is Encapsulation?**

Encapsulation is an OOPS concept to create and define the permissions and restrictions of an object and its member variables and methods. A very simple example to explain the concept is to make the member variables of a class private and providing public getter and setter methods. Java provides four types of access level modifiers: public, protected, no modifier and private.

1. **What is the difference between Abstraction and Encapsulation?**

“Program to interfaces, not implementations” is the principle for Abstraction and “Encapsulate what varies” is the OO principle for Encapsulation.

Abstraction provides a general structure of a class and leaves the details for the implementers. Encapsulation is to create and define the permissions and restrictions of an object and its member variables and methods.

Abstraction is implemented in Java using interface and abstract class while Encapsulation is implemented using four types of access level modifiers: public, protected, no modifier and private.

1. **What is Polymorphism?**

Polymorphism is the occurrence of something in various forms. Java supports various forms of polymorphism like polymorphic reference variables, polymorphic method, polymorphic return types and polymorphic argument types.

1. **What is Inheritance?**

A subclass can inherit the states and behaviors of it’s super class is known as inheritance.

1. **What is multiple inheritance?**

A child class inheriting states and behaviors from multiple parent classes is known as multiple inheritance.

1. **What is the diamond problem in inheritance?**

In case of multiple inheritance, suppose class A has two subclasses B and C, and a class D has two super classes B and C.If a method present in A is overridden by both B and C but not by D then from which class D will inherit that method B or C? This problem is known as diamond problem.

1. **Why Java does not support multiple inheritance?**

Java was designed to be a simple language and multiple inheritance introduces complexities like diamond problem. Inheriting states or behaviors from two different type of classes is a case which in reality very rare and it can be achieved easily through an object association.

1. **What is Static Binding and Dynamic Binding?**

Static or early binding is resolved at compile time. Method overloading is an example of static binding.

Dynamic or late or virtual binding is resolved at run time. Method overriding is an example of dynamic binding.

1. **What is the meaning of “IS-A” and “HAS-A” relationship?**

“IS-A” relationship implies inheritance. A sub class object is said to have “IS-A” relationship with the super class or interface. If class A extends B then A “IS-A” B. It is transitive, that is, if class A extends B and class B extends C then A “IS-A” C. The “instanceof” operator in java determines the “IS-A” relationship.

When a class A has a member reference variable of type B then A “HAS-A” B. It is also known as Aggregation.

1. **What is Association?**

Association is a relationship between two objects with multiplicity.

1. **What is Aggregation?**

Aggregation is also known as “HAS-A” relationship. When class Car has a member reference variable of type Wheel then the relationship between the classes Car and Wheel is known as Aggregation. Aggregation can be understood as “whole to its parts” relationship.

Car is the whole and Wheel is part. Wheel can exist without the Car. Aggregation is a weak association.

1. **What is Composition?**

Composition is a special form of Aggregation where the part cannot exist without the whole. Composition is a strong Association. Composition relationship is represented like aggregation with one difference that the diamond shape is filled.

1. **What is Dependency?**

When one class depends on another because it uses that at some point in time then this relationship is known as Dependency. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class. A Dependency is drawn as a dotted line from the dependent class to the independent class with an open arrowhead pointing to the independent class.

1. **What is the difference between Association and Dependency?**

The main difference between Association and Dependency is in case of Association one class has an attribute or member variable of the other class type but in case of Dependency a method takes an argument of the other class type or a method has a local variable of the other class type.

1. **What is a Class?**

A class is the specification or template of an object.

1. **What is an Object?**

Object is instance of class.

Example:-

object None extends Option[Nothing]

scala> Nil

res5: scala.collection.immutable.Nil.type = List()

scala> Nil.length

res6: Int = 0