OOP Concepts

1. What is **Object-Oriented Programming**?

   “Creating a program that can use and support objects”

A type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions.

Object-oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic.

In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

2. **Object-Oriented Programming v/s Procedure Oriented**?

Programs are made up of modules, which are parts of a program that can be coded and tested separately, and then assembled to form a complete program. In procedural languages (i.e. C) these modules are procedures, where a procedure is a sequence of statements. In C for example, procedures are a sequence of imperative statements, such as assignments, tests, loops and invocations of sub procedures. These procedures are functions, which map arguments to return statements.

The design method used in procedural programming is called Top Down Design. This is where you start with a problem (procedure) and then systematically break the problem down into sub problems (sub procedures). This is called functional decomposition, which continues until a sub problem is straightforward enough to be solved by the corresponding sub procedure. The difficulties with this type of programming, is that software maintenance can be difficult and time consuming. When changes are made to the main procedure (top), those changes can cascade to the sub procedures of main, and the sub-sub procedures and so on, where the change may impact all procedures in the pyramid.
One alternative to procedural programming is object oriented programming. Object oriented programming is meant to address the difficulties with procedural programming. In object oriented programming, the main modules in a program are classes, rather than procedures. The object-oriented approach lets you create classes and objects that model real world objects.

“OOP enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.”

3. What is Polymorphism?

“Ability to appear in many forms”

In object-oriented programming, polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes.

For example, given a base class shape, polymorphism enables the programmer to define different area methods for any number of derived classes, such as circles, rectangles and triangles. No matter what shape an object is, applying the area method to it will return the correct results. Polymorphism is considered to be a requirement of any true object-oriented programming language (OOPL).

The three types of polymorphism are:

- Ad-hoc (overloading and overriding)

  Overloaded methods are methods with the same name signature but either a different number of parameters or different types in the parameter list.

  Overridden methods are methods that are redefined within an inherited or subclass. They have the same signature and the subclass definition is used.

- Parametric (generics)

  Parametric are generic typing procedures.

- Dynamic method binding
Dynamic (or late) method binding is the ability of a program to resolve references to subclass methods at runtime.

Compile Time - achieved when a derived class overrides a base class method or implements an interface method.

Runtime - achieved when a derived class provides a different implementation of a 'virtual' method of the base class.

4. What is Encapsulation?

Encapsulation is the ability of an object to hide its data and methods from the rest of the world.

Binding of data and methods that acts on data inside a single entity called as an Object is called Encapsulation.

Coupling – the degree to which a module (class) depends on other classes. In a good design, you should try to minimize coupling.

Cohesion – is the measure within a module (class) of how well the members work together to provide a specific piece of functionality.

“A well-designed system should maximize cohesion, and minimize coupling.”

5. What is Inheritance?

Inheritance is the capability of a class to use the properties and methods of another class while adding its own functionality.

Inheritance is a compile-time mechanism in Java that allows you to extend a class (called the base class or super class) with another class (called the derived class or subclass).
In Java, Inheritance is used for two purposes:

1. **Class inheritance** - create a new class as an extension of another class, primarily for the purpose of code reuse. That is, the derived class inherits the public methods and public data of the base class. Java only allows a class to have one immediate base class, i.e., single class inheritance.

2. **Interface inheritance** - create a new class to implement the methods defined as part of an interface for the purpose of subtyping. That is a class that implements an interface “conforms to” (or is constrained by the type of) the interface. Java supports multiple interface inheritance.

6. **What is Abstract class?**

   An abstract class is a class that leaves one or more method implementations unspecified by declaring one or more methods abstract. An abstract method has no body (i.e., no implementation). A subclass is required to override the abstract method and provide an implementation. Hence, an abstract class is incomplete and cannot be instantiated, but can be used as a base class.

7. **What is an Interface?**

   An abstract class mixes the idea of mutable data in the form of instance variables, non-abstract methods, and abstract methods. An abstract class with only static final instance variables and all abstract methods is called an interface. An interface is a specification, or contract, for a set of methods that a class that implements the interface must conform to in terms of the type signature of the methods. The class that implements the interface provides an implementation for each method, just as with an abstract method in an abstract class.

   So, you can think of an interface as an abstract class with all abstract methods. The interface itself can have either public, package, private or protected access defined. All methods declared in an interface are implicitly abstract and implicitly public. It is not necessary, and in fact considered redundant to declare a method in an interface to be abstract.
**Summary of Object-Oriented Concepts**

1. Everything is an object.
2. Computation is performed by objects communicating with each other, requesting that other objects perform actions. Objects communicate by sending and receiving messages. A message is a request for action, bundled with whatever arguments may be necessary to complete the tasks.
3. Each object has its own memory, which consists of other objects.
4. Every object is an instance of a class. A class simply represents a grouping of similar objects, such as Integers or lists.
5. The class is the repository for behaviour associated with an object. That is, that all objects that are instances of the same class can perform the same actions.
6. Classes are organized into a singly rooted tree structure, called the inheritance hierarchy. Memory and behaviour associated with instances of a class are automatically available to any class associated with a descendant in this tree structure.