Abstraction:

Abstraction is a process to hide data and only to show the functionalities.

2 ways to achieve abstraction

1. Abstract class
2. Interface

Abstract class:

Abstract class contains both concrete and abstract method. Abstract methods are implemented without body. It is not mandatory to put an abstract method in abstract class. Abstract class can only be used by extend to another class.

They can’t be instantiated. The child class should override all the abstract methods of parent else the child class should be declared with abstract keyword.

Interface:

Interface contains only abstract methods but not concrete methods. A class can implement multiple interfaces. That class should implement all the abstract methods of interfaces. Multiple inheritances can be achieved only using interface.

Polymorphism:

Polymorphism is the ability to process objects differently on the basis of their class and data types. Polymorphism occurs when we have many classes that are interrelated to each other using inheritance. In this method overriding is used widely to use same method and attributes of classes.

2 types of polymorphism:

1) **Static polymorphism (compile time)->** known as method overloading and operator overloading. It works in same class with same method name but different number of parameter. Static Polymorphism in Java decides which method to execute during compile time. Due to the difference in the parameters, every method has a different signature. The Java compiler has an idea of which method is called.
2) **Dynamic polymorphism (run time) ->** known as method overriding. Dynamic polymorphism in Java refers to the process when a call to an overridden process is resolved at the run time. The reference variable of a super class calls the overridden method.

Inheritance:

a class can inherit attributes and methods from another class. The class that inherits the properties is known as the sub-class or the child class. The class from which the properties are inherited is known as the super class or the parent class.

Encapsulation :

Encapsulation is one of the fundamental principles of object-oriented programming (OOP) and is implemented in Java to help achieve data hiding, abstraction, and control over access to class members

In Java, encapsulation is achieved through the following mechanisms:

Access Modifiers: Java provides four access modifiers to control the visibility of class members:

public: The member is accessible from any other class.

protected: The member is accessible within its own class, subclasses, and classes within the same package.

default (no modifier): The member is accessible within its own class and classes within the same package.

private: The member is accessible only within its own class.

Getter and Setter Methods: Encapsulation involves providing public methods (getters and setters) to access and modify the private fields of a class. Getters retrieve the values of fields, and setters modify the values while enforcing any necessary validation or logic.