Week1

Object-oriented programming System (OOPs) is a programming paradigm based on the concept of “objects” that contain data and methods. The primary purpose of object-oriented programming is to increase the flexibility and maintainability of programs. Object oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works. Java is mature OOP language.

1. Abstraction:
   * Abstraction is the process of hiding the implementation details and exposing only the functionality to the user.
   * In Java, abstraction is achieved through abstract classes and interfaces.
   * Users know what an object does rather than how it achieves its functionality.
2. Encapsulation:
   * Encapsulation is a mechanism in Java that bundles the data (variables) and the code (methods) that operate on the data into a single unit.
   * Variables of a class are declared as private to hide them from other classes.
   * Public setter and getter methods are provided to modify and access the variable values, ensuring controlled access.
3. Inheritance:
   * Inheritance is the process where one class (subclass) acquires the properties (methods and fields) of another class (superclass).
   * The subclass is also known as the derived class or child class, and the superclass is known as the base class or parent class.
   * The extends keyword is used for a class to inherit properties from another class.
4. Polymorphism:
   * Polymorphism involves processing objects differently based on their data types.
   * One method can have multiple implementations for a certain class of action.
   * Polymorphism can be achieved through method overloading and method overriding.
5. Polymorphism Implementation:
   * Overloading:
     + More than one method with the same name but different parameters.
     + Resolution is done at compile-time.
   * Overriding:
     + A derived class provides a specific implementation for a method defined in its superclass.
     + The method to be invoked is determined at runtime.

Decision control in Java refers to the structures that allow you to make decisions in your code based on certain conditions. These are typically implemented using if statements, switch statements, and ternary operators.

The **if** statement is the most basic decision-making statement. It allows your program to execute a certain section of code only if a specified condition is true.

The **switch** statement allows you to select one of many code blocks to be executed.

You can use nested if statements to create more complex decision structures.

In Java, modifiers are keywords that define the scope, accessibility, and behavior of classes, methods, and variables. There are several types of modifiers in Java, and they can be categorized into two main groups: access modifiers and non-access modifiers.

Access modifiers control the visibility and accessibility of classes, methods, and variables.

There are four main access modifiers in Java:

**public:**

* Public members are accessible from any other class.

private:

* Private members are only accessible within the same class.

Protected members are accessible within the same package and by subclasses (even if they are in different packages)

default (Package-Private):

If no access modifier is specified (i.e., default), it is visible only within the same package.

Members with default access are not visible to classes outside the package.

**Non-Access Modifiers:**

Non-access modifiers provide additional information about classes, methods, and variables. Here are some commonly used non-access modifiers:

**final:**

* The **final** modifier is used with classes, methods, and variables.
* A **final** class cannot be subclassed, a **final** method cannot be overridden, and a **final** variable cannot be reassigned.

**static:**

* The **static** modifier is used with variables and methods.
* A **static** variable is shared among all instances of a class, and a **static** method belongs to the class rather than an instance.

**abstract:**

* The **abstract** modifier is used with classes and methods.
* An **abstract** class cannot be instantiated, and an **abstract** method must be implemented by any concrete (non-abstract) subclass.

**synchronized:**

* The **synchronized** modifier is used with methods and blocks.
* It ensures that only one thread can access the synchronized method or block at a time, preventing data corruption due to concurrent access