Introduction to Object-Oriented Programming

Why?

* Reuse code -> Write once, use many times
* Organize Code -> Code is easier to read and maintain
* Handles complexity -> Makes it easier to solve big problems by breaking them into smaller parts.

Core Concepts:

* Class
* Object
* Encapsulation
* Inheritance
* Polymorphism
* Abstraction

Objects and Classes:

What is class?

* A class is a blueprint.
* Assume that you are designing a car
* First We will create a blueprint which will describe what a car should have, like doors, engine, wheels.
* This blueprint is nothing but a class.

public class Car {  
 // Attributes  
 String color;  
 int speed;  
  
 void start(){  
 System.*out*.println("The car is starting....");  
 }  
  
 void stop(){  
 System.*out*.println("The car is stopping....");  
 }  
}

What is Object?

* Object is an actual car that is built using blueprint.
* It’s a real instance of the class.

public class CarObject {  
 public static void main(String[] args) {  
 Car myCar = new Car(); // created object  
 myCar.color="Black";  
 myCar.speed= 100;  
  
 Car yourCar = new Car();  
 yourCar.color= "Red";  
 yourCar.speed=120;  
  
 System.*out*.println("Your cars colour is "+ yourCar.color);  
  
 System.*out*.println("My Cars colour is "+ myCar.color);  
 myCar.start();  
 }  
}

Key principles of OOP:

Encapsulation:

* It combines data(attributes) and methods into a single unit, restricting the direct access to the data.

Why?

* Protects data integrity by allowing controlled access using getter and setter.

What are access modifiers?

* Public access modifier
* Scope: The wildest visibility. Members declared as public can be accessed from anywhere in the program.

package package1;  
  
public class PublicClass {  
 public String publicVariable = "I am public";  
  
 public void display(){  
 System.*out*.println(publicVariable);  
 }  
}

package package2;  
  
import package1.PublicClass;  
  
public class TestPublic {  
 public static void main(String[] args) {  
 PublicClass obj = new PublicClass();  
 obj.display(); // accessible because it is public  
 }  
}

What is Private Access Modifier?

Scope: Most restrictive, If we are declaring members as private then they are accessible within the same class only.

package privateExample;  
  
public class PrivateClassExample {  
 private String privateVariable = "I am private";  
  
 private void display(){  
 System.*out*.println(privateVariable);  
 }  
  
 public void accessPrivate(){  
 display(); //  
 }  
}

package privateExample;  
  
public class TestPrivate {  
 public static void main(String[] args) {  
 PrivateClassExample obj = new PrivateClassExample();  
 // obj.display(); ERROR: Not Accessible  
 obj.accessPrivate(); // Indirect access is allowed  
 }  
}

What are protected access modifiers?

What are default access modifiers?

Abstraction: