Class, Objects, Setters, Getters, and Constructors in Java

# Introduction

In object-oriented programming (OOP), a class is a template or blueprint for creating objects, which are instances of that class. Objects in Java encapsulate data (attributes) and methods that act on that data.

# 1. Class

A class in Java is a user-defined data type that defines the attributes (data fields) and methods (functions) that its objects will have. A class is defined using the 'class' keyword.

## Example:

class Car {  
 String brand;  
 String model;  
  
 void displayInfo() {  
 System.out.println("Car: " + brand + " " + model);  
 }  
}  
  
// Creating an object of Car class  
Car myCar = new Car();  
myCar.brand = "Toyota";  
myCar.model = "Corolla";  
myCar.displayInfo();

# 2. Object

An object is an instance of a class. When a class is defined, no memory is allocated until an object of that class is created. Objects store the actual values of the attributes and can access the methods defined in the class.

## Example:

// Object Creation  
Car myCar = new Car();  
myCar.brand = "Toyota";  
myCar.model = "Corolla";  
myCar.displayInfo(); // Outputs: Car: Toyota Corolla

# 3. Constructors

A constructor in Java is a special method that is used to initialize objects. It has the same name as the class and is called automatically when an object of the class is created. Constructors can be parameterized to initialize object attributes with specific values.

## Example:

class Car {  
 String brand;  
 String model;  
  
 // Constructor  
 Car(String brand, String model) {  
 this.brand = brand;  
 this.model = model;  
 }  
  
 void displayInfo() {  
 System.out.println("Car: " + brand + " " + model);  
 }  
}  
  
// Constructor is called when creating an object  
Car myCar = new Car("Honda", "Civic");  
myCar.displayInfo();

# 4. Getters and Setters

Getters and setters are used to access and update private attributes of a class. These methods allow controlled access to an object's attributes. Typically, attributes are marked as private, and getters and setters provide the interface to read and modify them.

• Getters are used to retrieve the value of an attribute.

• Setters are used to modify or set the value of an attribute.

## Example:

class Car {  
 private String brand;  
 private String model;  
  
 // Constructor  
 Car(String brand, String model) {  
 this.brand = brand;  
 this.model = model;  
 }  
  
 // Getter for brand  
 public String getBrand() {  
 return brand;  
 }  
  
 // Setter for brand  
 public void setBrand(String brand) {  
 this.brand = brand;  
 }  
  
 // Getter for model  
 public String getModel() {  
 return model;  
 }  
  
 // Setter for model  
 public void setModel(String model) {  
 this.model = model;  
 }  
}  
  
// Example usage  
Car myCar = new Car("Ford", "Mustang");  
System.out.println(myCar.getBrand()); // Outputs: Ford  
myCar.setBrand("Chevrolet");  
System.out.println(myCar.getBrand()); // Outputs: Chevrolet