| Experiment No. 10 |
| --- |
| To implement single inheritance in object-oriented programming, demonstrating how derived classes inherit attributes and behaviors from a single base class |
| Date of Performance:23/08/24 |
| Date of Submission:30/08/24 |

**Aim:-** To implement single inheritance in object-oriented programming, demonstrating how derived classes inherit attributes and behaviors from a single base class

**Objective:-** Develop a Java program to demonstrate single inheritance by creating a base class Vehicle with attributes such as make, model, and year, and methods for setting and retrieving these attributes. Derive a subclass Car from Vehicle to include additional attributes specific to cars, such as numDoors and isConvertible, along with appropriate methods for these attributes. Implement the program to instantiate objects of both classes, set values for their attributes, and display these values to illustrate the inheritance relationship and functionality of the classes.

**Theory:-**

In object-oriented programming, inheritance is a mechanism where a class (subclass or derived class) inherits properties and behaviors from another class (superclass or base class). Single inheritance specifically refers to the inheritance relationship where a subclass extends a single superclass.

Key Concepts:

Base Class (Superclass):

The base class is the class whose attributes and methods are inherited by another class. It serves as a blueprint for creating objects with shared characteristics.

Derived Class (Subclass):

The derived class is the class that inherits from a superclass. It extends the functionality of the superclass by adding its own attributes and methods or by overriding inherited methods.

Inheritance Mechanism:

Code Reusability: Inheritance allows subclasses to reuse code defined in their superclass. This promotes code reuse and reduces redundancy.

Subclass Specialization: Subclasses can specialize behavior inherited from the superclass by adding new functionality or modifying existing behavior through method overriding.

Is-a Relationship: Inheritance models the "is-a" relationship, where a subclass is considered to be a more specific type of its superclass.

Example Scenario:

Consider a scenario where you have a superclass Vehicle with attributes like make, model, and methods to set and retrieve these attributes. You then create a subclass Car that inherits from Vehicle and adds specific attributes like numDoors and isConvertible. By inheriting from Vehicle, Car automatically gains access to the attributes and methods defined in Vehicle, while also allowing you to define additional attributes specific to cars.

Benefits and Considerations:

Code Organization: Inheritance helps organize code by grouping related attributes and behaviors in a hierarchical structure.

Promotes Modularity: Subclasses can be independently tested and modified without affecting the superclass or other subclasses.

Single Inheritance Limitation: Java supports single inheritance where a class can extend only one superclass. This avoids complexities related to multiple inheritance but may limit flexibility in certain scenarios.

Conclusion:

Single inheritance in Java is a fundamental concept that facilitates code reuse, promotes modularity, and allows for hierarchical organization of classes. By understanding how to effectively use inheritance, developers can design and implement robust object-oriented solutions that are both maintainable and scalable.

**Code:-**

class Vehicle {

private String make;

private String model;

private int year;

public Vehicle(String make, String model, int year) {

this.make = make;

this.model = model;

this.year = year;

}

public void setMake(String make) {

this.make = make;

}

public void setModel(String model) {

this.model = model;

}

public void setYear(int year) {

this.year = year;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public int getYear() {

return year;

}

public void displayInfo() {

System.out.println("Make: " + make);

System.out.println("Model: " + model);

System.out.println("Year: " + year);

}

}

class Car extends Vehicle {

private int numDoors;

private boolean isConvertible;

public Car(String make, String model, int year, int numDoors, boolean isConvertible) {

super(make, model, year);

this.numDoors = numDoors;

this.isConvertible = isConvertible;

}

public void setNumDoors(int numDoors) {

this.numDoors = numDoors;

}

public void setConvertible(boolean isConvertible) {

this.isConvertible = isConvertible;

}

public int getNumDoors() {

return numDoors;

}

public boolean isConvertible() {

return isConvertible;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Number of Doors: " + numDoors);

System.out.println("Convertible: " + (isConvertible ? "Yes" : "No"));

}

}

public class Main {

public static void main(String[] args) {

Vehicle myVehicle = new Vehicle("Toyota", "Corolla", 2020);

System.out.println("Vehicle Information:");

myVehicle.displayInfo();

Car myCar = new Car("Ferrari", "488", 2024, 2, true);

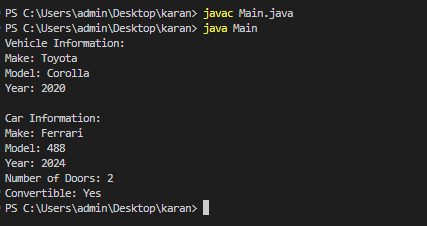
System.out.println("\nCar Information:");

myCar.displayInfo();

}

}

**Output:-**

****

**Conclusion:-**

Single Inheritance: A class inherits from one superclass.

Multilevel Inheritance: A class inherits from another derived class.

Hierarchical Inheritance: Multiple classes inherit from a single superclass.

Multiple Inheritance (via interfaces): A class implements multiple interfaces, avoiding ambiguity.