**Implementing the Singleton Pattern**

class Singleton{

    private static Singleton obj = null;

    private Singleton(){

    }

    public static Singleton getInstance(){

        if(obj == null)

        {

            synchronized(Singleton.class){

                if(obj == null)

                {

                     obj = new Singleton();

                }

            }

        }

        return obj;

    }

}

public class Main{

    public static void main(String args[])

    {

        Singleton obj1 = Singleton.getInstance();

        if(obj1 != null)

        System.out.println(obj1.hashCode());

        Singleton obj2 = Singleton.getInstance();

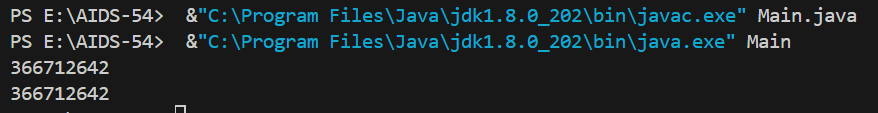
        if(obj2 != null)

        System.out.println(obj2.hashCode());

    }

}

**Output**



**Implementing the Factory method pattern**

interface Vehicle {

void drive();

}

class Car implements Vehicle {

public void drive() {

System.out.println("Driving a car...");

}

}

class Motorcycle implements Vehicle {

public void drive() {

System.out.println("Driving a motorcycle...");

}

}

class VehicleFactory {

public Vehicle createVehicle(String type) {

if (type.equalsIgnoreCase("car")) {

return new Car();

} else if (type.equalsIgnoreCase("motorcycle")) {

return new Motorcycle();

}

// Handle unknown vehicle types or return a default vehicle

return new Car();

}

}

public class Main{

public static void main(String args[])

{

//Vehicle car = new Car();

//Vehicle motor = new Motorcycle();

VehicleFactory obj = new VehicleFactory();

Vehicle car = obj.createVehicle("car");

Vehicle motorcycle = obj.createVehicle("Motorcycle");

car.drive();

motorcycle.drive();

}

}

**Output**

