OOAD with Java Self Learning – 1

Java Serialization

Name: Sahana Rao

SRN: PES1UG20CS588

Section: J

Introduction to Serialization and Deserialization

Serialization is a mechanism of converting the state of an object into a byte stream.

Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

In Java, serialization and deserialization are supported by the **Serializable interface**, which is a marker interface that indicates a class can be serialized. When a class implements Serializable, it allows its instances to be serialized and deserialized using Java's built-in serialization mechanism.

Applications

1. **Persistence:** Serialization is commonly used for saving the state of an object to a file or database, so that it can be later retrieved and restored.
2. **Remote Method Invocation (RMI):** Serialization is used in RMI to pass objects between client and server applications over a network. When a client invokes a remote method on a server, the arguments and return values are serialized and deserialized as they are passed over the network.
3. **Caching:** Serialization can be used to store frequently used objects in memory or on disk, so that they can be quickly retrieved and reused.
4. **Messaging:** Serialization can be used to send messages between distributed systems or microservices. When a message is sent, the data is serialized into a format that can be transmitted over a network, and then deserialized at the receiving end.

5. **Cloning:** Serialization can be used to create a deep copy of an object by serializing it and then deserializing the resulting byte stream into a new object.

Overall, serialization and deserialization provide a powerful and flexible mechanism for storing, transmitting, and manipulating objects in Java.

What is Serializable, ObjectOutputStream, FileOutputStream, ObjectInputStream and related methods?

In Java, serialization is implemented using the Serializable interface, ObjectOutputStream, FileOutputStream, ObjectInputStream and related methods.

1. **Serializable:** Serializable is a marker interface that indicates that a class can be serialized. When a class implements Serializable, it allows its instances to be serialized and deserialized using Java's built-in serialization mechanism.
2. **ObjectOutputStream:** The ObjectOutputStream class is used to write primitive data types, and Java objects to an OutputStream. Only objects that support the java.io.Serializable interface can be written to streams.ObjectOutputStream implements the DataOutput interface, which provides methods for writing primitive types and strings to a stream.
3. **FileOutputStream:** FileOutputStream is a class that provides the ability to write bytes to a file. This class is used to write serialized objects to a file. FileOutputStream implements the OutputStream interface, which provides methods for writing bytes to a stream.
4. **ObjectInputStream:** ObjectInputStream is a class that provides the ability to read objects from a stream in a

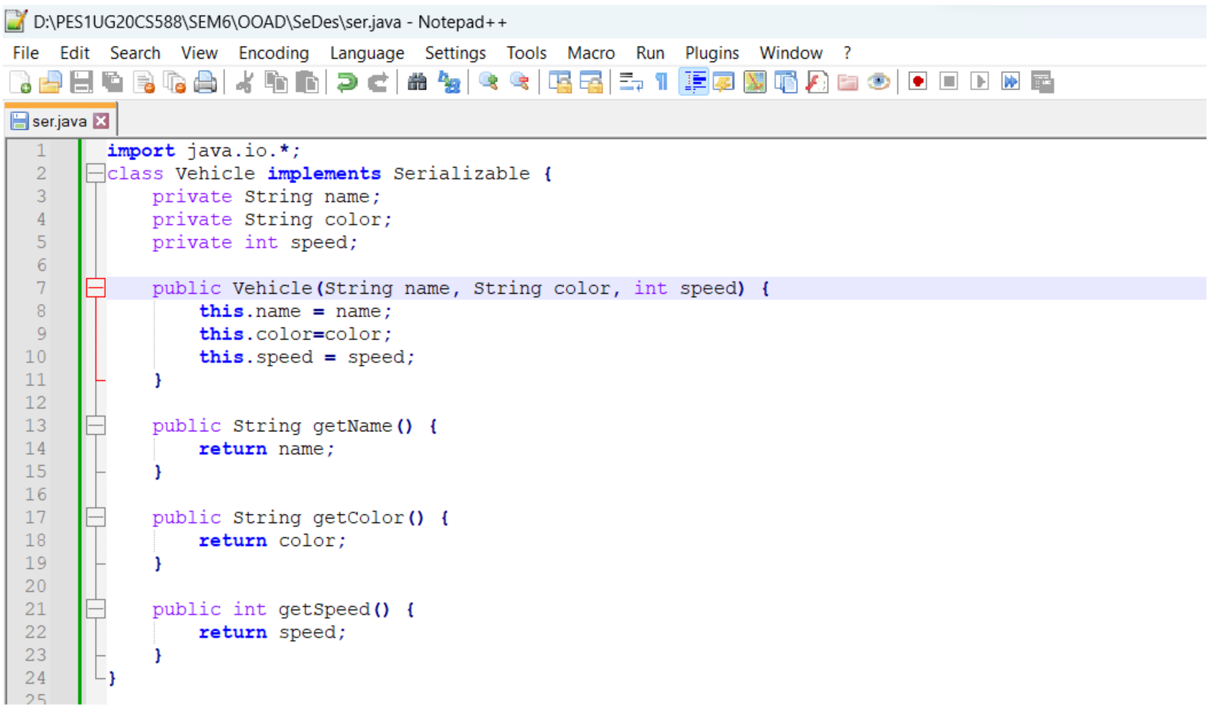
serialized form. This class is used to deserialize objects from a file or a network stream. ObjectInputStream implements the DataInput interface, which provides methods for reading primitive types and strings from a stream.

Some commonly used methods in these classes:

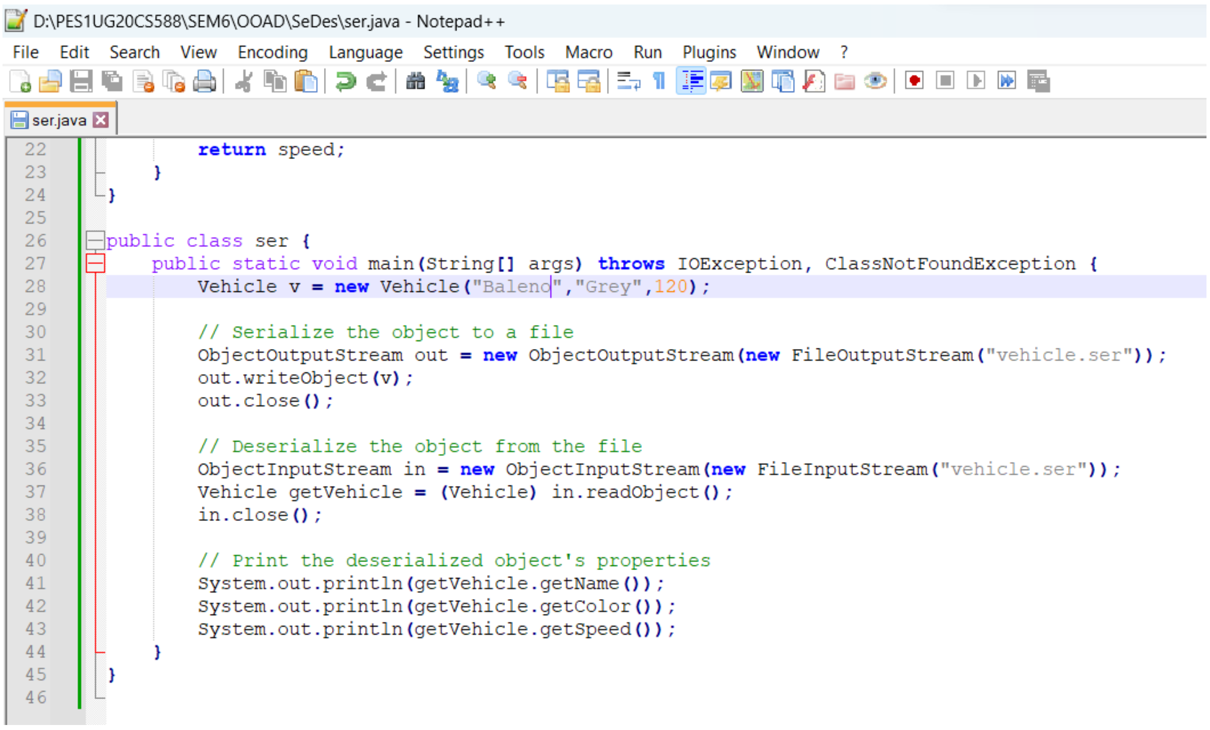
1. **ObjectOutputStream.writeObject(Object obj):** This method writes an object to the ObjectOutputStream. The object must be Serializable.
2. **ObjectOutputStream.flush():** This method flushes the output stream, meaning that any buffered data is written to the underlying stream.
3. **ObjectOutputStream.close():** This method closes the ObjectOutputStream, releasing any system resources associated with it.
4. **FileOutputStream(String fileName):** This constructor creates a new FileOutputStream with the specified file name.
5. **ObjectInputStream.readObject():** This method reads an object from the ObjectInputStream. The object must be Serializable.
6. **ObjectInputStream.close():** This method closes the ObjectInputStream, releasing any system resources associated with it.

Example Codes

1a. Serialization

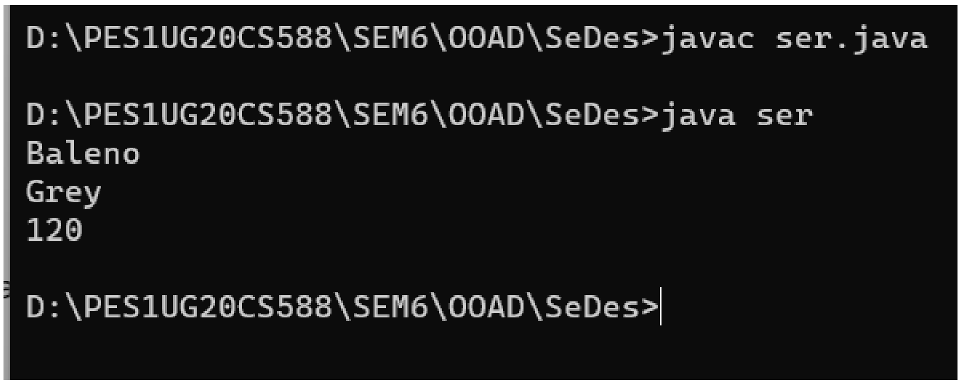

D:\PES1UG20CS588\SEM6\OOAD\SeDes\ser.java - Notepad++

```
import java.io.*;
class Vehicle implements Serializable {
    private String name;
    private String color;
    private int speed;

    public Vehicle(String name, String color, int speed) {
        this.name = name;
        this.color=color;
        this.speed = speed;
    }

    public String getName() {
        return name;
    }

    public String getColor() {
        return color;
    }

    public int getSpeed() {
        return speed;
    }
}
```


D:\PES1UG20CS588\SEM6\OOAD\SeDes\ser.java - Notepad++

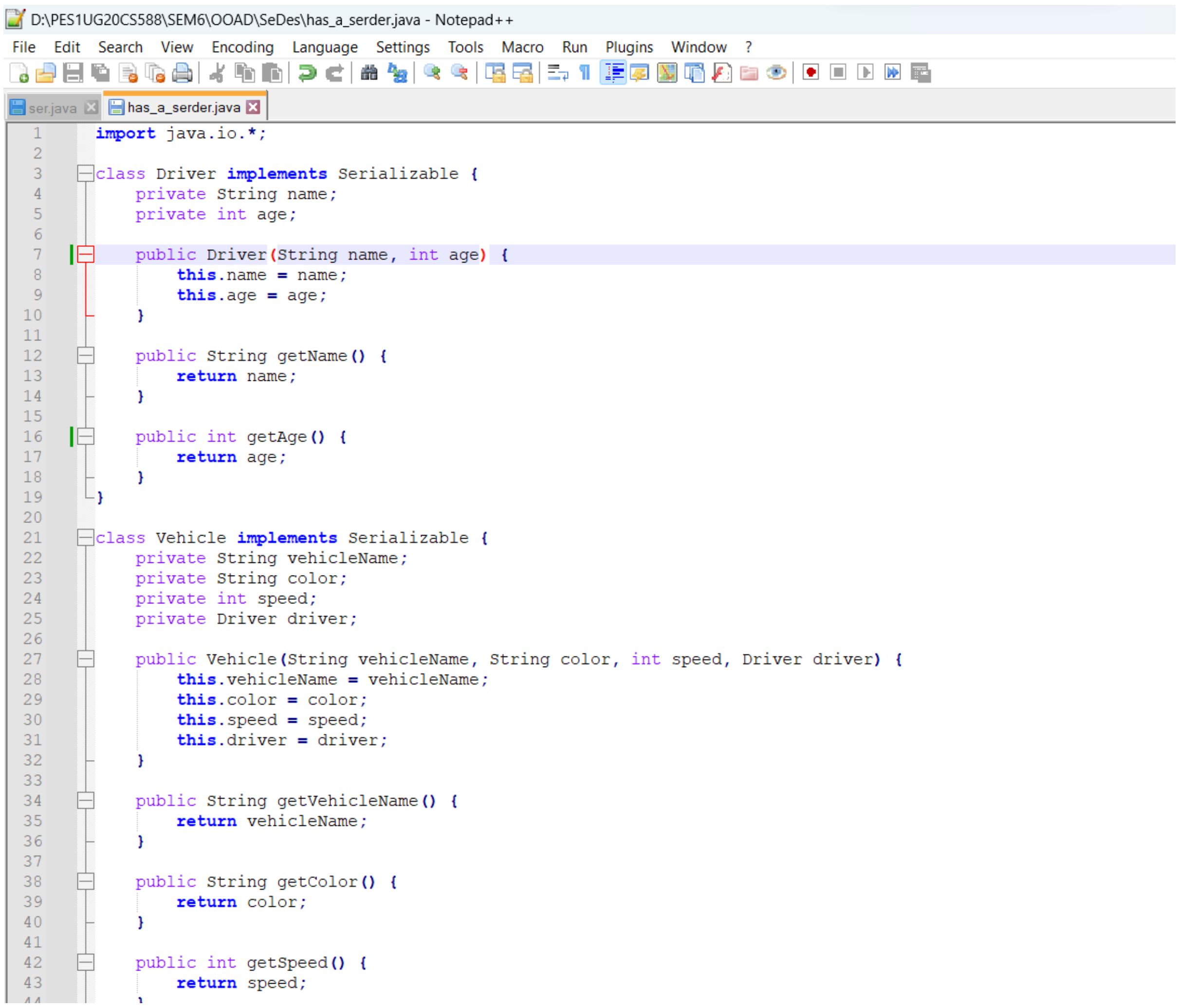
```
        return speed;
    }
}

public class ser {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        Vehicle v = new Vehicle("Baleno","Grey",120);

        // Serialize the object to a file
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("vehicle.ser"));
        out.writeObject(v);
        out.close();

        // Deserialize the object from the file
        ObjectInputStream in = new ObjectInputStream(new FileInputStream("vehicle.ser"));
        Vehicle getVehicle = (Vehicle) in.readObject();
        in.close();

        // Print the deserialized object's properties
        System.out.println(getVehicle.getName());
        System.out.println(getVehicle.getColor());
        System.out.println(getVehicle.getSpeed());
    }
}
```

1b. Snapshot of outputs



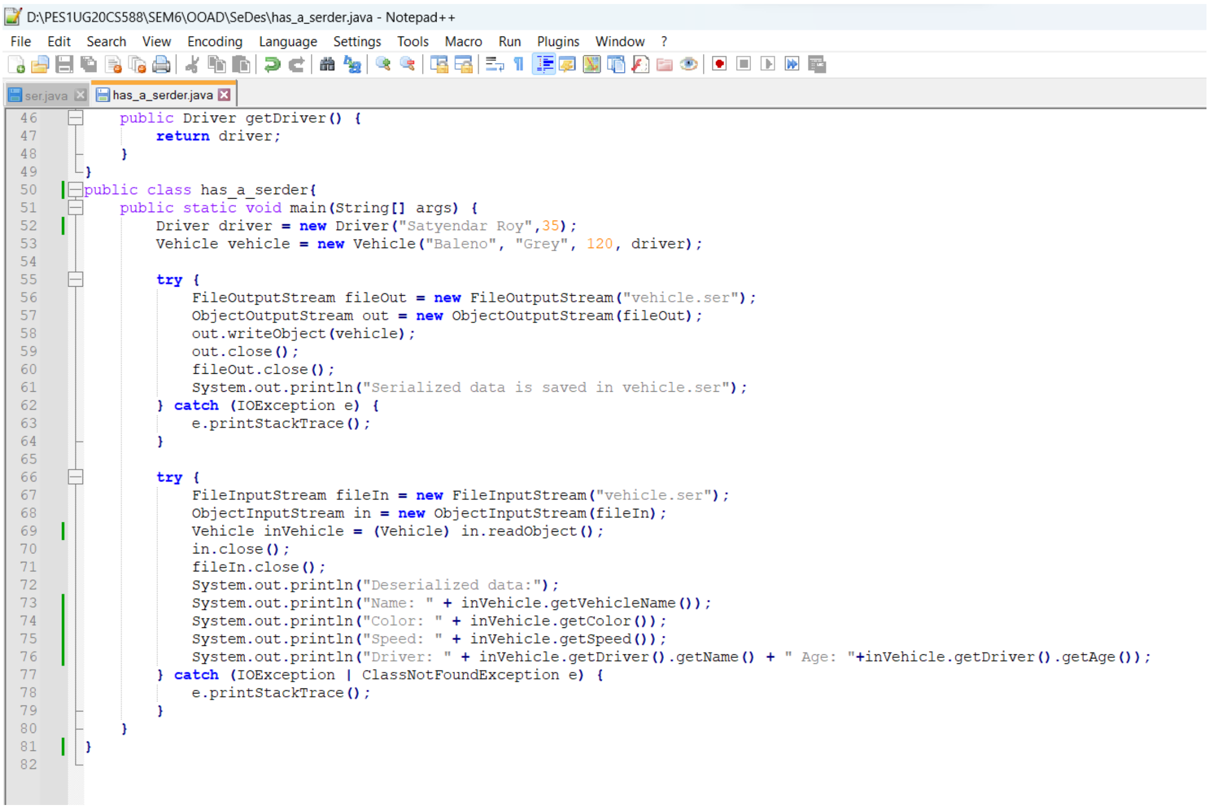
```
D:\PES1UG20CS588\SEM6\OOAD\SeDes>javac ser.java

D:\PES1UG20CS588\SEM6\OOAD\SeDes>java ser
Baleno
Grey
120

D:\PES1UG20CS588\SEM6\OOAD\SeDes>
```

2a. Serialization with ‘has-a‘ relationship



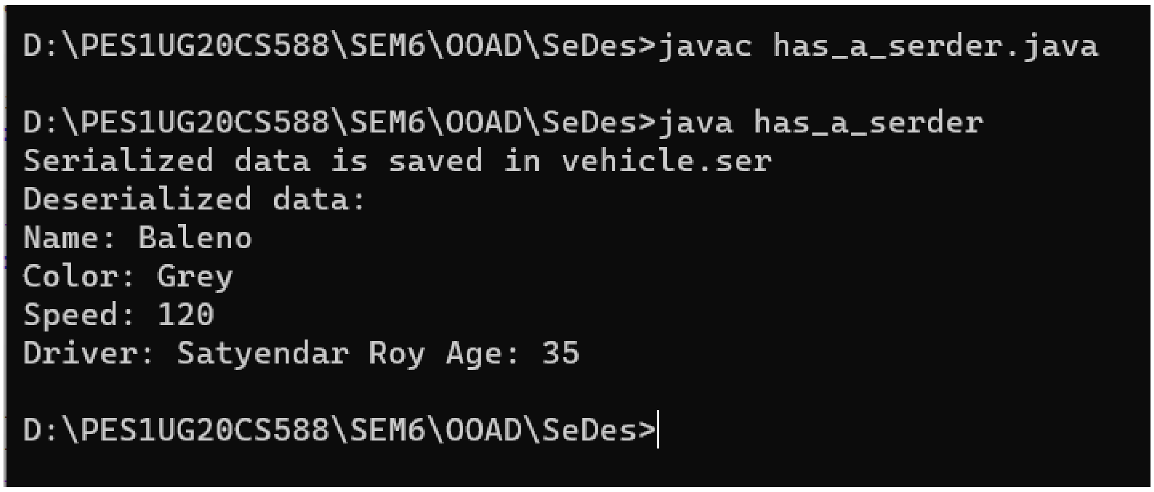
D:\PES1UG20CS588\SEM6\OOAD\SeDes\has_a_serder.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ser.java
has_a_serder.java

```java
import java.io.*;

class Driver implements Serializable {
    private String name;
    private int age;

    public Driver(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}

class Vehicle implements Serializable {
    private String vehicleName;
    private String color;
    private int speed;
    private Driver driver;

    public Vehicle(String vehicleName, String color, int speed, Driver driver) {
        this.vehicleName = vehicleName;
        this.color = color;
        this.speed = speed;
        this.driver = driver;
    }

    public String getVehicleName() {
        return vehicleName;
    }

    public String getColor() {
        return color;
    }

    public int getSpeed() {
        return speed;
```


D:\PES1UG20CS588\SEM6\OOAD\SeDes\has_a_serder.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ser.java
has_a_serder.java

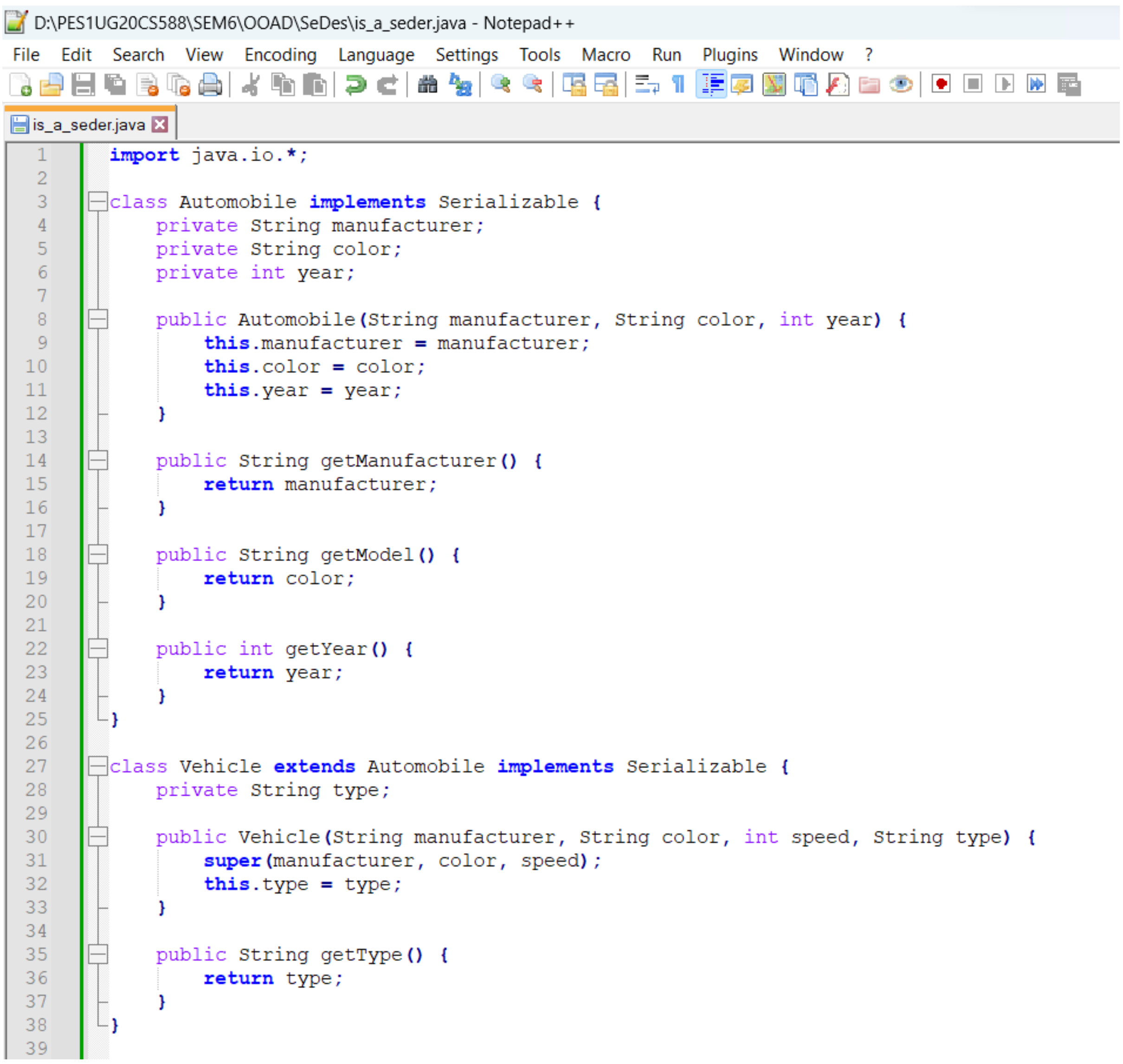
```java
    public Driver getDriver() {
        return driver;
    }
}
public class has_a_serder{
    public static void main(String[] args) {
        Driver driver = new Driver("Satyendar Roy",35);
        Vehicle vehicle = new Vehicle("Baleno", "Grey", 120, driver);

        try {
            FileOutputStream fileOut = new FileOutputStream("vehicle.ser");
            ObjectOutputStream out = new ObjectOutputStream(fileOut);
            out.writeObject(vehicle);
            out.close();
            fileOut.close();
            System.out.println("Serialized data is saved in vehicle.ser");
        } catch (IOException e) {
            e.printStackTrace();
        }

        try {
            FileInputStream fileIn = new FileInputStream("vehicle.ser");
            ObjectInputStream in = new ObjectInputStream(fileIn);
            Vehicle inVehicle = (Vehicle) in.readObject();
            in.close();
            fileIn.close();
            System.out.println("Deserialized data:");
            System.out.println("Name: " + inVehicle.getVehicleName());
            System.out.println("Color: " + inVehicle.getColor());
            System.out.println("Speed: " + inVehicle.getSpeed());
            System.out.println("Driver: " + inVehicle.getDriver().getName() + " Age: "+inVehicle.getDriver().getAge());
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

2b. Snapshots of outputs



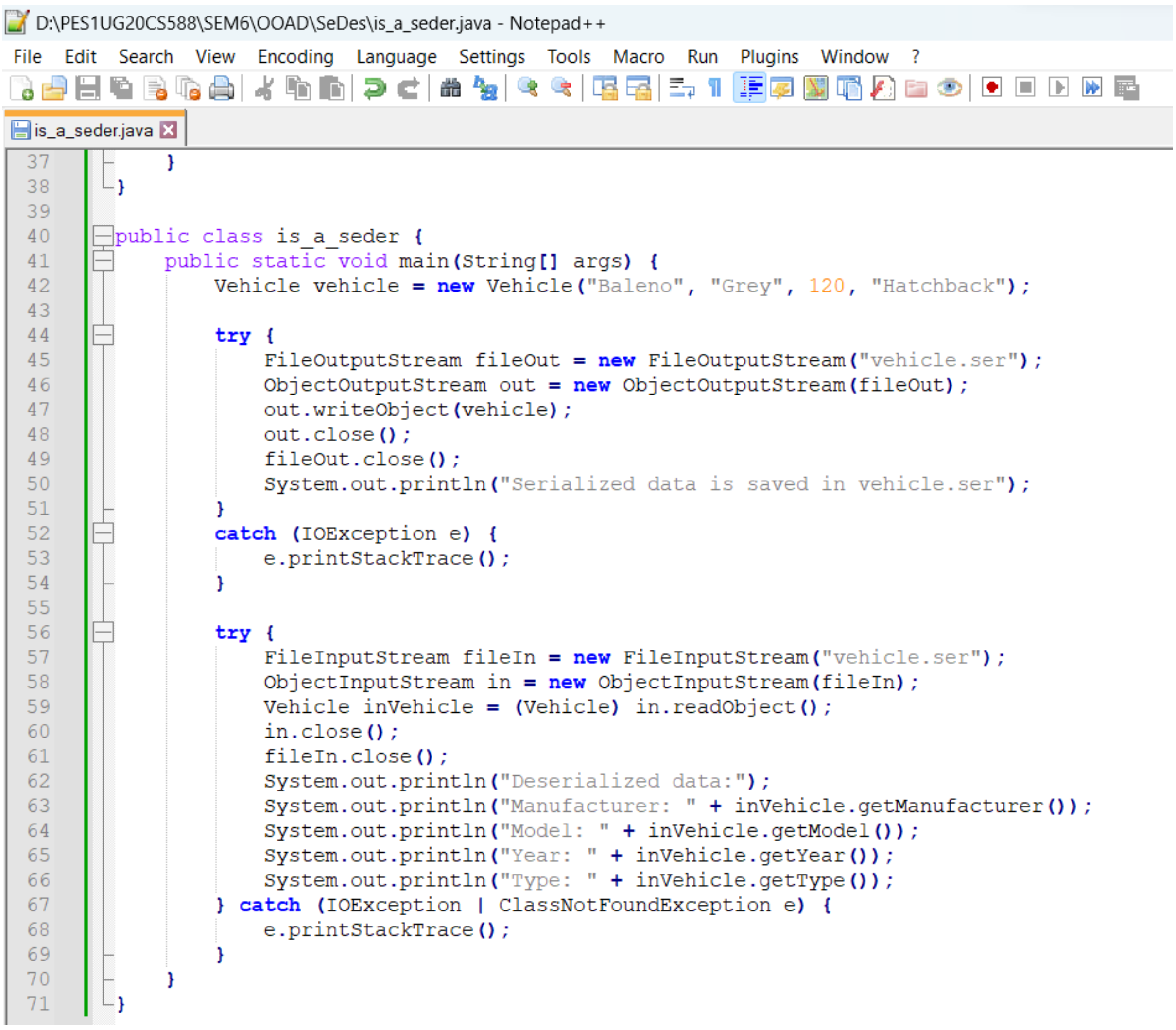
```
D:\PES1UG20CS588\SEM6\OOAD\SeDes>javac has_a_serder.java

D:\PES1UG20CS588\SEM6\OOAD\SeDes>java has_a_serder
Serialized data is saved in vehicle.ser
Deserialized data:
Name: Baleno
Color: Grey
Speed: 120
Driver: Satyendar Roy Age: 35

D:\PES1UG20CS588\SEM6\OOAD\SeDes>
```

3a. Serialization with ‘is-a‘ relationship



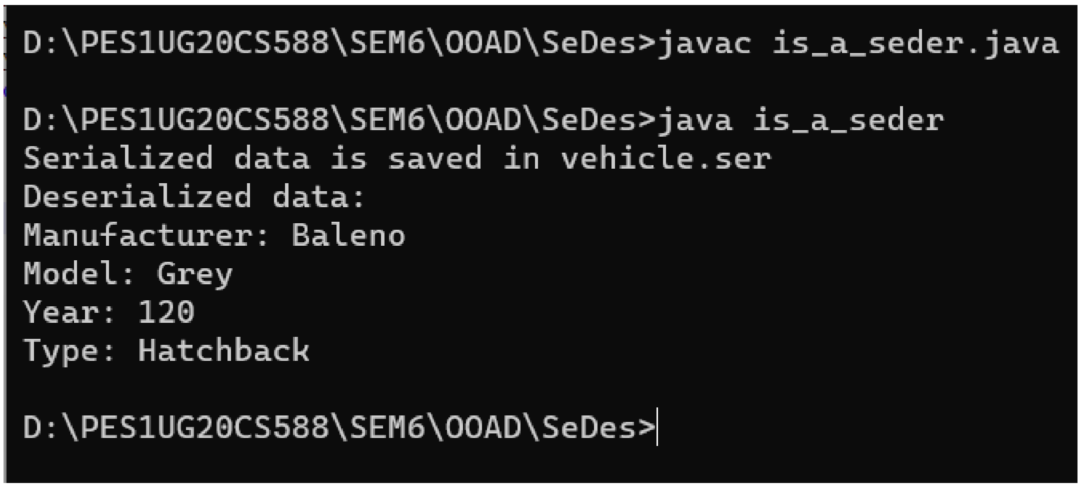
D:\PES1UG20CS588\SEM6\OOAD\SeDes\is_a_seder.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
is_a_seder.java

```java
import java.io.*;

class Automobile implements Serializable {
    private String manufacturer;
    private String color;
    private int year;

    public Automobile(String manufacturer, String color, int year) {
        this.manufacturer = manufacturer;
        this.color = color;
        this.year = year;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public String getModel() {
        return color;
    }

    public int getYear() {
        return year;
    }
}

class Vehicle extends Automobile implements Serializable {
    private String type;

    public Vehicle(String manufacturer, String color, int speed, String type) {
        super(manufacturer, color, speed);
        this.type = type;
    }

    public String getType() {
        return type;
    }
}

```


D:\PES1UG20CS588\SEM6\OOAD\SeDes\is_a_seder.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
is_a_seder.java

```java
    }
}

public class is_a_seder {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle("Baleno", "Grey", 120, "Hatchback");

        try {
            FileOutputStream fileOut = new FileOutputStream("vehicle.ser");
            ObjectOutputStream out = new ObjectOutputStream(fileOut);
            out.writeObject(vehicle);
            out.close();
            fileOut.close();
            System.out.println("Serialized data is saved in vehicle.ser");
        }
        catch (IOException e) {
            e.printStackTrace();
        }

        try {
            FileInputStream fileIn = new FileInputStream("vehicle.ser");
            ObjectInputStream in = new ObjectInputStream(fileIn);
            Vehicle inVehicle = (Vehicle) in.readObject();
            in.close();
            fileIn.close();
            System.out.println("Deserialized data:");
            System.out.println("Manufacturer: " + inVehicle.getManufacturer());
            System.out.println("Model: " + inVehicle.getModel());
            System.out.println("Year: " + inVehicle.getYear());
            System.out.println("Type: " + inVehicle.getType());
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

3b. Snapshots of outputs



```
D:\PES1UG20CS588\SEM6\OOAD\SeDes>javac is_a_seder.java

D:\PES1UG20CS588\SEM6\OOAD\SeDes>java is_a_seder
Serialized data is saved in vehicle.ser
Deserialized data:
Manufacturer: Baleno
Model: Grey
Year: 120
Type: Hatchback

D:\PES1UG20CS588\SEM6\OOAD\SeDes>
```