Experiment 5.2

1. **Aim:** Create a Java program to serialize and deserialize a Student object.
The program should:
- Serialize a Student object (containing id, name, and GPA) and save it to a file.
- Deserialize the object from the file and display the student details.
- Handle FileNotFoundException, IOException, and ClassNotFoundException using exception handling.

2. **Objective:** The objective is to serialize and deserialize a Student object, store and retrieve its id, name, and GPA from a file, and handle exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

3. Algorithm:

Step 1: Initialize the Program
1. Start the program.
2. Import the necessary classes (java.io.*).
3. Define a Student class implementing Serializable.
4. Declare attributes:
    - id (int)
    - name (String)
    - gpa (double)
5. Define a constructor to initialize Student objects.
6. Override toString() to display student details.

Step 2: Define the Serialization Method
1. Create serializeStudent(Student student).
2. Use a try-with-resources block to create an ObjectOutputStream:
    - Open a FileOutputStream to write to student.ser.
    - Write the Student object to the file using writeObject().
3. Handle exceptions:
    - FileNotFoundException → Print error message.
    - IOException → Print error message.
4. Print a success message if serialization is successful.

Step 3: Define the Deserialization Method
1. Create deserializeStudent().
2. Use a try-with-resources block to create an ObjectInputStream:
    - Open a FileInputStream to read student.ser.
    - Read the Student object using readObject().
3. Handle exceptions:
    - FileNotFoundException → Print error message.
    - IOException → Print error message.
    - ClassNotFoundException → Print error message.
4. Print the deserialized student details.

Step 4: Execute Main Function
1. Define main(String[] args).
2. Create a Student object with sample data.
3. Call serializeStudent() to save the object.
4. Call deserializeStudent() to read and display the object.

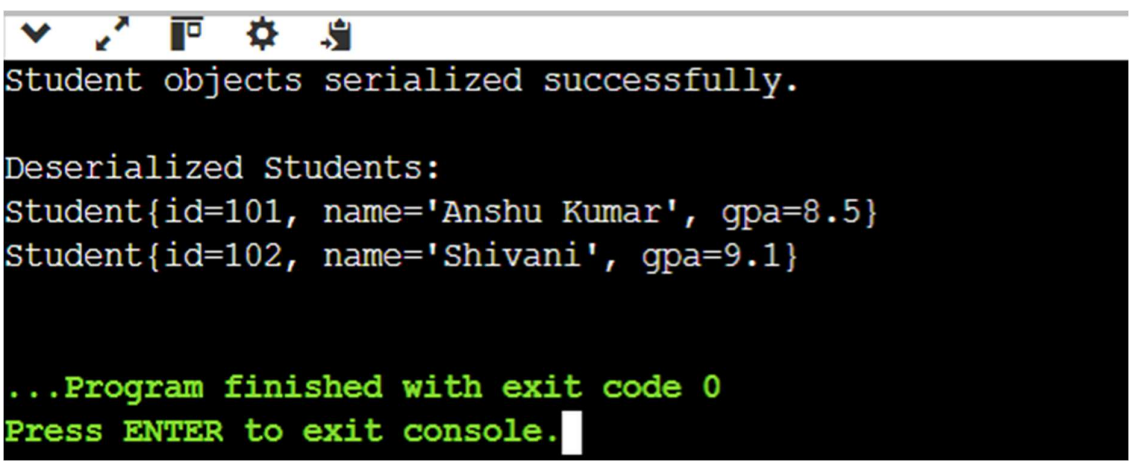
Step 5: Terminate the Program
1. End execution.

4. Implementation Code:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;

// Serializable Student class
class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private double gpa;

    public Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    @Override
    public String toString() {
        return "Student{id=" + id + ", name='" + name + "', gpa=" + gpa + "}";
    }
}

public class StudentSerialization {
    private static final String FILE_NAME = "students.ser";

    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student(101, "Ravi", 8.5));
        students.add(new Student(102, "Priya", 9.1));

        serializeStudents(students);
        deserializeStudents();
    }

    // Serializing students list
    public static void serializeStudents(List<Student> students) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            oos.writeObject(new ArrayList<>(students)); // Fixes unchecked cast issue
            System.out.println("Student objects serialized successfully.");
        } catch (IOException e) {
            System.err.println("IOException occurred: " + e.getMessage());
        }
    }

    // Deserializing students list
    public static void deserializeStudents() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            Object obj = ois.readObject();
            if (obj instanceof List<?>) {
                List<?> rawList = (List<?>) obj;
```

```
            List<Student> students = new ArrayList<>();
            for (Object item : rawList) {
                if (item instanceof Student) {
                    students.add((Student) item);
                }
            }
            System.out.println("\nDeserialized Students:");
            for (Student student : students) {
                System.out.println(student);
            }
        }
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Exception occurred: " + e.getMessage());
    }
  }
}
```

5. Output



6. Learning Outcomes:

- Understand object serialization and deserialization in Java.
- Learn how to use ObjectOutputStream and ObjectInputStream for file operations.
- Implement exception handling for FileNotFoundException, IOException, and ClassNotFoundException.
- Gain hands-on experience in storing and retrieving objects from a file.
- Develop skills in data persistence and file management using Java.