EXPERIMENT 5

Student Name: Harshit Shandilya	**UID:** 22BET10274
Branch: BE-IT	**Section/Group:** BET-701-A
Semester: 6	**Date of Performance:**25.02.2025
Subject Name: Project Based Learning in Java	**Subject Code:** 22ITH-359

Aim:

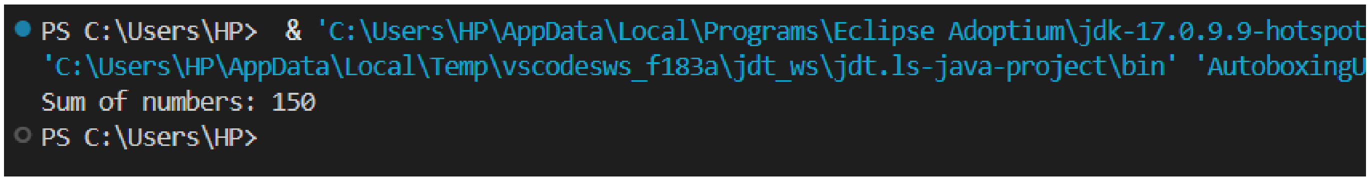
Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing. Include methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

Objective:

The objective of this Java program is to demonstrate object serialization and deserialization by creating a Student class with fields for id, name, and GPA. The program will serialize an instance of Student and store it in a file, then deserialize the object back into memory and display its details. It will also include proper exception handling for FileNotFoundException, IOException, and ClassNotFoundException to ensure robustness in file operations.

Source code:

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

public class AutoboxingUnboxingExample {

    public static List<Integer> parseStringList(String[] stringNumbers) {
        List<Integer> integerList = new ArrayList<>();
        for (String num : stringNumbers) {
            integerList.add(Integer.parseInt(num));
        }
        return integerList;
    }

    public static int calculateSum(List<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num;
        }
        return sum;
    }

    public static void main(String[] args) {
        String[] stringNumbers = {"10", "20", "30", "40", "50"};
        List<Integer> numbers = parseStringList(stringNumbers);
        int sum = calculateSum(numbers);
        System.out.println("Sum of numbers: " + sum);
    }
}
```



```
PS C:\Users\HP>  & 'C:\Users\HP\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.9.9-hotspot
 'C:\Users\HP\AppData\Local\Temp\vscodesws_f183a\jdt_ws\jdt.ls-java-project\bin' 'AutoboxingU
 Sum of numbers: 150
PS C:\Users\HP>
```

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. write a program and program name to save without comments write objective in paragrpah form.

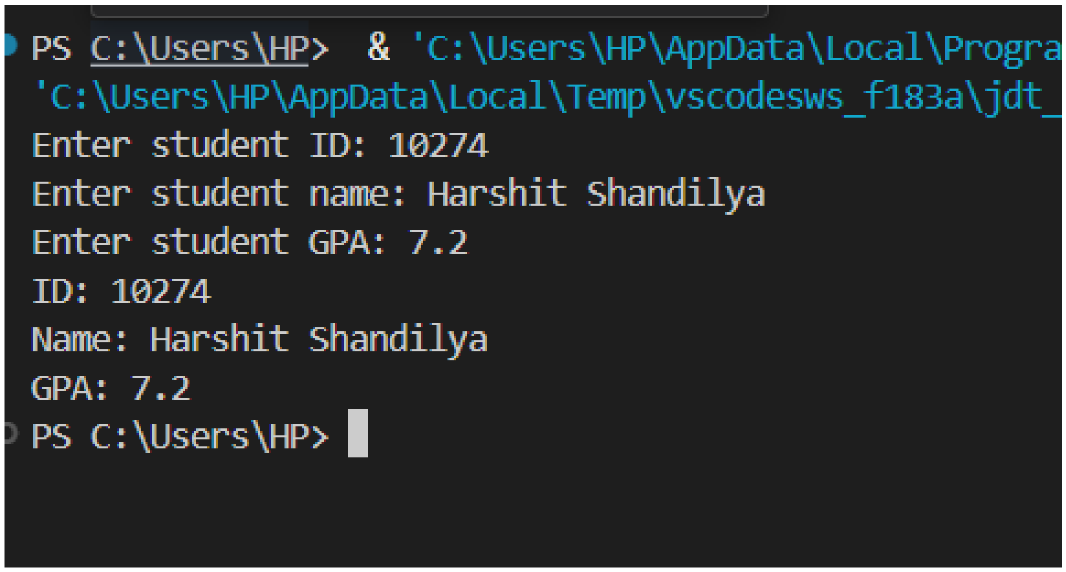
Objective:

The objective of this program is to implement serialization and deserialization of a Student object in Java. The program will create a Student class containing attributes such as id, name, and GPA. It will serialize an instance of this class and save it to a file, ensuring that the object state is preserved for future retrieval. Additionally, the program will deserialize the Student object from the file and display its details on the console. Proper exception handling mechanisms will be incorporated to manage potential errors such as FileNotFoundException, IOException, and ClassNotFoundException, ensuring the program executes smoothly and robustly.

aa

Source code:

```
import java.io.*;

import java.util.Scanner;



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;

    }



    public void display() {

        System.out.println("ID: " + id);

        System.out.println("Name: " + name);

        System.out.println("GPA: " + gpa);

    }

}



public class StudentSerialization {

    public static void serializeStudent(Student student, String filename) {

        try (ObjectOutputStream out = new ObjectOutputStream(new 
FileOutputStream(filename))) {

            out.writeObject(student);

        } catch (IOException e) {

            System.err.println("Error during serialization: " + e.getMessage());

        }

    }



    public static Student deserializeStudent(String filename) {

        try (ObjectInputStream in = new ObjectInputStream(new 
FileInputStream(filename))) {

            return (Student) in.readObject();
        } catch (FileNotFoundException e) {
            System.err.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("Error during deserialization: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Class not found: " + e.getMessage());
        }
        return null;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter student ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter student name: ");
        String name = scanner.nextLine();
        System.out.print("Enter student GPA: ");
        double gpa = scanner.nextDouble();

        String filename = "student.ser";
        Student student = new Student(id, name, gpa);
        serializeStudent(student, filename);
        Student deserializedStudent = deserializeStudent(filename);
        if (deserializedStudent != null) {
            deserializedStudent.display();
        }
    }
}
```



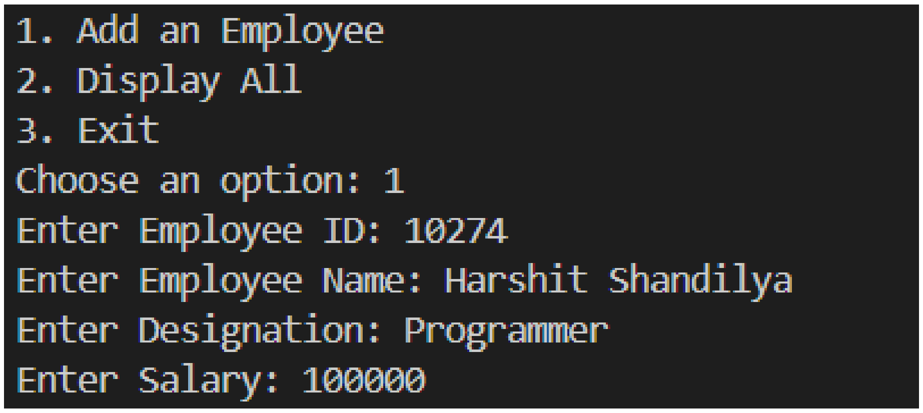
```
PS C:\Users\HP>  & 'C:\Users\HP\AppData\Local\Progra
'C:\Users\HP\AppData\Local\Temp\vscodesws_f183a\jdt_
Enter student ID: 10274
Enter student name: Harshit Shandilya
Enter student GPA: 7.2
ID: 10274
Name: Harshit Shandilya
GPA: 7.2
PS C:\Users\HP>
```

Aim:
Create a menu-based Java application with the following options. 1.Add an Employee 2. Display All 3. Exit If option 1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected, the application should display all the employee details. If option 3 is selected the application should exit write a program and program name to save without comments.

Objective:
The objective of this program is to develop a menu-based Java application for managing employee records using file handling. The application provides three options: adding an employee, displaying all employees, and exiting. When adding an employee, the program collects details such as employee ID, name, designation, and salary, and stores them in a file for persistent storage. When displaying employees, the program reads the stored employee records from the file and presents them on the console. The program ensures efficient data handling and retrieval while incorporating proper exception handling to manage potential file-related error.

Source code:

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

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private String designation;
    private double salary;

    public Employee(int id, String name, String designation, double salary) {
        this.id = id;
        this.name = name;
        this.designation = designation;
        this.salary = salary;
    }

    public void display() {
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Designation: " + designation);
        System.out.println("Salary: " + salary);
    }
}

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser";

    public static void addEmployee() {
        try (Scanner scanner = new Scanner(System.in);
             ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILE_NAME, true))) {
            System.out.print("Enter Employee ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Enter Employee Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter Designation: ");
            String designation = scanner.nextLine();
            System.out.print("Enter Salary: ");
            double salary = scanner.nextDouble();
            Employee employee = new Employee(id, name, designation, salary);
            out.writeObject(employee);
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    public static void displayEmployees() {
        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            while (true) {
                Employee employee = (Employee) in.readObject();
                employee.display();
                System.out.println();
            }
        } catch (EOFException e) {
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Error reading from file: " + e.getMessage());
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("1. Add an Employee");
            System.out.println("2. Display All");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```



```
1. Add an Employee
2. Display All
3. Exit
Choose an option: 1
Enter Employee ID: 10274
Enter Employee Name: Harshit Shandilya
Enter Designation: Programmer
Enter Salary: 100000
```