Experiment 5

Student Name: Lakshay Verma **UID: 22BCS15481**
Branch: BE/CSE **Section/Group: 22BCS_IOT-618/A**
Semester: 6th **Date of Performance: 21/01/25**
Subject Name: Project Based Learning in JAVA with Lab **Subject Code: 22CSH-359**

1. **Aim:** This program aims to demonstrate the use of autoboxing and unboxing in Java while calculating the sum of a list of integers. It also ensures efficient parsing of string values into integers, handling invalid inputs gracefully.

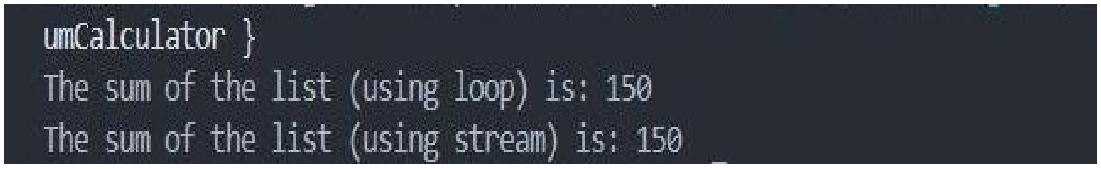
2. **Objective:** The objective is to create a program that takes a mix of integer values and numeric strings, converts them into Integer objects, and calculates their sum using unboxing. It also handles exceptions for invalid numeric strings without interrupting execution.

3. **Implementation/Code:**

```
import java.util.*;

public class IntegerSumCalculator
  { public static void main(String[] args)
  {
    List<Integer> numbers = new ArrayList<>();
    String[] inputs = {"10", "20", "30", "40", "50", "invalid"};

    for (String input : inputs) {
      Integer num = parseStringToInteger(input);
      if (num != null) {
        numbers.add(num); // Autoboxing: int -> Integer
      }
    }

    int sum = calculateSum(numbers);
    System.out.println("The sum of the list is: " + sum);
  }

  public static Integer parseStringToInteger(String str)
    { try {
      return Integer.parseInt(str); // Parsing string to Integer
```

```
        } catch (NumberFormatException e) {


            System.out.println("Invalid number format: " + str);
            return null;
        }
    }

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

4. Output:



```
umCalculator }
The sum of the list (using loop) is: 150
The sum of the list (using stream) is: 150
```

5.2 Aim: - This program aims to demonstrate object serialization and deserialization in Java by saving a Student object to a file and then reading it back. It ensures proper handling of exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

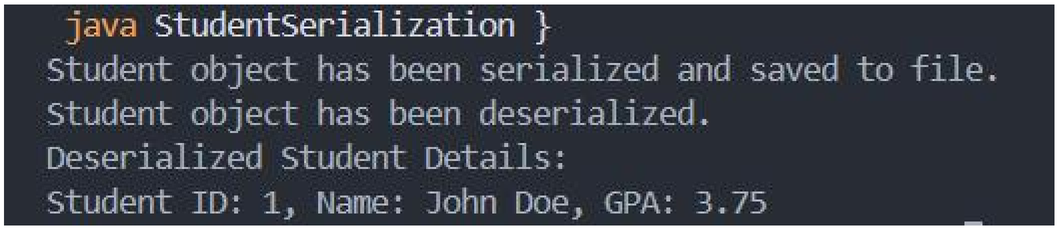
Objectives: - The objective is to implement a Student class that implements Serializable, serialize its instance to a file, and deserialize it back, displaying the student details. The program should also handle scenarios where the file is missing or an incompatible class version is encountered.

Code: -

```
import java.io.*;

// Student class implementing Serializable
class Student implements Serializable {
    private static final long serialVersionUID = 1L; // Ensures version consistency
    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("Student ID: " + id + ", Name: " + name + ", GPA: " + gpa);
    }
}

public class StudentSerialization {
    private static final String FILE_NAME = "student.ser"; // Serialized file name

    public static void main(String[] args) {
        // Test Case 1: Serialize and Deserialize a valid Student object
        Student student = new Student(1, "John Doe", 3.75);
        serializeStudent(student);
         Student deserializedStudent = deserializeStudent();

        if (deserializedStudent != null) {
            System.out.println("Student object has been deserialized.");
            System.out.println("Deserialized Student Details:");
            deserializedStudent.display();
            }
        }

        // Method to serialize the Student object
        public static void serializeStudent(Student student) {
            try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME)))
           {
                oos.writeObject(student);
                System.out.println("Student object has been serialized and saved to file.");
            } catch (IOException e) {
                System.out.println("Error during serialization: " + e.getMessage());
            }
        }

        // Method to deserialize the Student object
        public static Student deserializeStudent() {
            try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
                { return (Student) ois.readObject();
            } catch (FileNotFoundException e)
                { System.out.println("Error: File not found.");
            } catch (IOException e) {
                System.out.println("Error during deserialization: " + e.getMessage());
            } catch (ClassNotFoundException e)
                { System.out.println("Error: Class not found.");
            }
            return null;
        }
    }
```

Output:-

dkssdlkfjf



```
  java StudentSerialization }
 Student object has been serialized and saved to file.
 Student object has been deserialized.
 Deserialized Student Details:
 Student ID: 1, Name: John Doe, GPA: 3.75
```

5.3 Aim : - This program aims to create a menu-driven Java application that allows users to add employee details, store them in a file, and display all stored employee records. It ensures proper file handling and exception management for a smooth user experience.

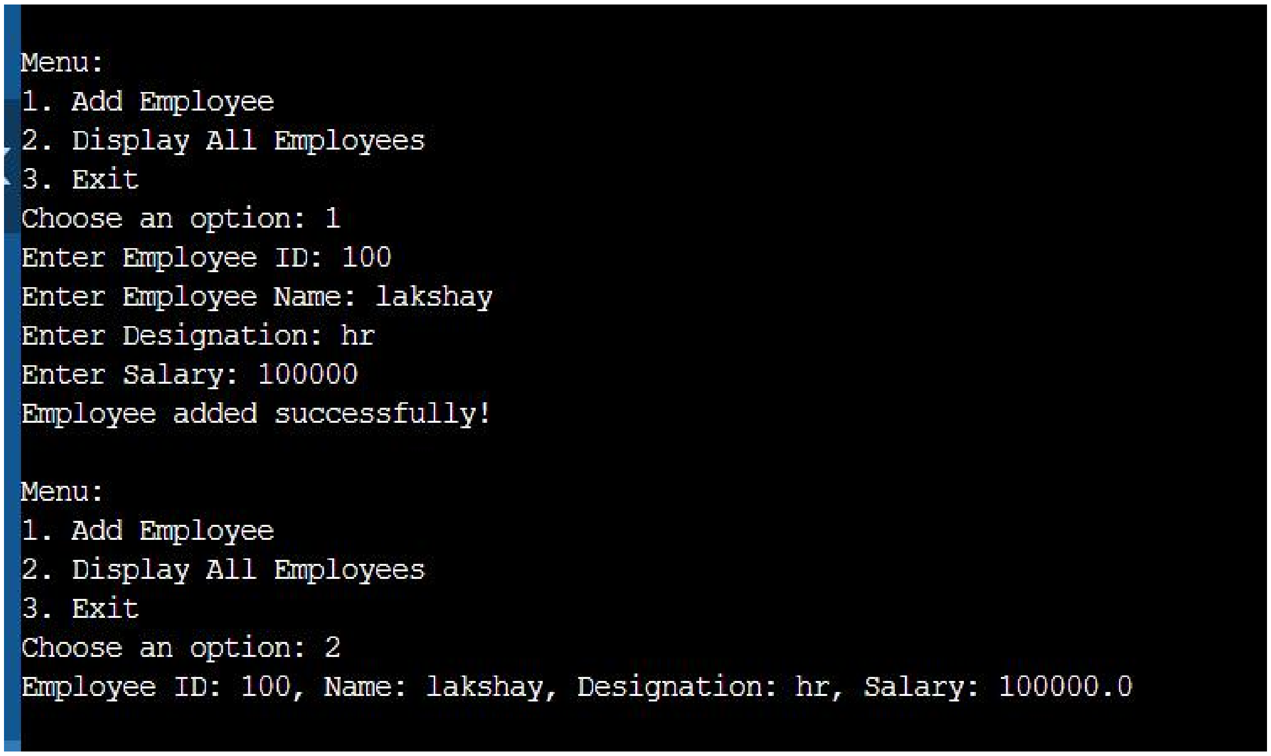
Objectives:- The objective is to implement an Employee class that supports serialization, enabling the storage and retrieval of employee data from a file. The program provides menu options to add employees, display all employees.

Code:-

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

// Employee class implementing Serializable
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("Employee ID: " + id + ", Name: " + name + ", Designation: " + designation + ",
      Salary: " + salary);
    }
}

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser"; // Serialized file name
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args)
        { while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employee");
            System.out.println("2. Display All Employees");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice)
          { case 1:
              addEmployee();
              break;
          case 2:
              displayAllEmployees();
              break;
          case 3:
              System.out.println("Exiting program...");
              return;
          default:
              System.out.println("Invalid choice! Please try again.");
        }
    }
}

// Method to add an employee and serialize to file
public static void addEmployee() {
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME,
  true))) {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        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);
        oos.writeObject(employee);
        System.out.println("Employee added successfully!");

    } catch (IOException e) {
        System.out.println("Error during file writing: " + e.getMessage());
    }
}
```

```
// Method to display all employees from file
public static void displayAllEmployees() {
    List<Employee> employees = readEmployeesFromFile();
    if (employees.isEmpty()) {
        System.out.println("No employee records found.");
    } else {
        System.out.println("\nEmployee Records:");
        for (Employee emp : employees) {
            emp.display();
        }
    }
}

// Method to read employees from file
public static List<Employee> readEmployeesFromFile()
    { List<Employee> employees = new ArrayList<>();
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
        { while (true) {
            try {
                Employee employee = (Employee) ois.readObject();
                employees.add(employee);
            } catch (EOFException e)
                { break; // End of file reached
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("File not found. No employee records available.");
    } catch (IOException | ClassNotFoundException e)
        { System.out.println("Error reading employee data: " + e.getMessage());
    }
    return employees;
}
 }
```

Output :-



```
Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID: 100
Enter Employee Name: lakshay
Enter Designation: hr
Enter Salary: 100000
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 2
Employee ID: 100, Name: lakshay, Designation: hr, Salary: 100000.0
```