Experiment-5

Student Name: Anshika Goel **UID: 22BCS11678**
Branch: BE-CSE **Section/Group: 22BCS_IOT-643/B**
Semester: 6th **Date of Performance: 27/02/25**
Subject: Project-Based Learning with Java **Subject Code: 22CSH-359**

Easy -Level

1. **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 wrapper classes (e.g., Integer.parseInt()).

2. **Algorithm:**

- **Initialize an array of strings** containing the numbers as text (e.g., ["10", "20", "30", "40", "50", "1810", "1110"]).
- **Create an empty list** (integerList) to store the converted integer values.
- **Loop through each element** in the array of strings:
  - Convert the current string element to an integer using Integer.parseInt().
  - Add the converted integer to the list integerList.
- **Print the list of integers** to show the numbers that were converted.
- **Initialize a variable** sum to 0.
- **Loop through the integer list**: For each integer, add its value to the sum.
- **Print the total sum** calculated from the integers.

3. **Implementation/Code:**

```
import java.util.ArrayList;
import java.util.List;
public class WrapperClass {
   public static void main(String[] args) {
      String[] numbers = {"10", "20", "30", "40", "50", "1810", "1110"};
      List<Integer> integerList = new ArrayList<>();
      for (String str : numbers) {
         integerList.add(parseStringToInteger(str));
      }
      System.out.println("The numbers are: " + integerList);
      int sum = calculateSum(integerList);
      System.out.println("The sum of the numbers is: " + sum);
   }
   public static Integer parseStringToInteger(String str) {
      return Integer.parseInt(str);
   }
public static int calculateSum(List<Integer> integerList) {
      int sum = 0;
```

```
        for (Integer num : integerList) {
            sum += num;
        }
        return sum;
    }
}
```

4. **Output:**

```
The numbers are: [10, 20, 30, 40, 50, 1810, 1110]
The sum of the numbers is: 3070
```

Medium-Level

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. **Algorithm:**

- Define a student class implementing Serializable with id, name, and gpa variables. Create a constructor to initialize these variables and override the toString() method to return a string representation of the student.
- In main(), create a Student object with specific values for the variables.
- Serialize the Student object by opening a FileOutputStream for the file student.ser and wrapping it with an ObjectOutputStream. Use the writeObject() method to write the Student object to the file.
- Handle serialization exceptions (FileNotFoundException, IOException).
- Deserialize the Student object by opening a FileInputStream for student.ser and wrapping it with an ObjectInputStream. Use the readObject() method to read the object and cast it to Student.
- Handle exceptions during deserialization (e.g., FileNotFoundException, IOException, ClassNotFoundException).
- Print the details of the deserialized Student object using toString().
- End the program.

3. **Implementation/Code:**

```
import java.io.*;
class Student implements Serializable {
    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 String toString() {
        return "Student ID: " + id + "\nName: " + name + "\nGPA: " + gpa;
    }
}
public class Serialization {
    public static void main(String[] args) {
        Student student = new Student(11678, "Anshika Goel", 8.2);
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("student.ser"))) {
            out.writeObject(student);
            System.out.println("Student object has been serialized and saved to 'student.ser'.");
        }
        catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        }
       catch (IOException e) {
            System.out.println("I/O error during serialization: " + e.getMessage());
        }
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("student.ser"))) {
            Student deserializedStudent = (Student) in.readObject();
            System.out.println("\nDeserialized Student Details:");
            System.out.println(deserializedStudent);
        }
       catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        }
        catch (IOException e) {
            System.out.println("I/O error during deserialization: " + e.getMessage());
        }
        catch (ClassNotFoundException e) {
            System.out.println("Class not found: " + e.getMessage());
        }
    }
     }
```

4. **Output:**

```
Student object has been serialized and saved to 'student.ser'.

Deserialized Student Details:
Student ID: 11678
Name: Anshika Goel
GPA: 8.2
```

```
¬í··sr··StudentÙ·áTd··è···D··gpaI··idL··namet··Ljava/lang/String;xp@ ffffff··-·t··Anshika Goel
```

Hard-Level

1. **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.

2. **Algorithm:**

3. **Implementation/Code:**

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

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

    public Employee(String name, int id, String designation, double salary) {
        this.name = name;
        this.id = id;
        this.designation = designation;
        this.salary = salary;
    }
    public String getName() {
        return name;
    }
    public int getId() {
        return id;
    }
    public String getDesignation() {
        return designation;
    }
    public double getSalary() {
        return salary;
    }
    public String toString() {
        return "Employee ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;
    }
}
public class Main {
    private static final String FILE_NAME = "employees.dat";
    private static ArrayList<Employee> employees = new ArrayList<>();

    public static void main(String[] args) {
        loadEmployeesFromFile();
        Scanner scanner = new Scanner(System.in);
        int choice = 0;
        while (choice != 3) {
            System.out.println("\nEmployee Management System");
```

```
        System.out.println("1. Add Employee");
        System.out.println("2. Display All Employees");
        System.out.println("3. Exit");
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                addEmployee(scanner);
                break;
            case 2:
                displayAllEmployees();
                break;
            case 3:
                saveEmployeesToFile();
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
private static void addEmployee(Scanner scanner) {
    System.out.print("Enter Employee Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter Employee ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Enter Employee Designation: ");
    String designation = scanner.nextLine();
    System.out.print("Enter Employee Salary: ");
    double salary = scanner.nextDouble();

    Employee employee = new Employee(name, id, designation, salary);
    employees.add(employee);
    System.out.println("Employee added successfully.");
}
private static void displayAllEmployees() {
    if (employees.isEmpty()) {
        System.out.println("No employees found.");
    }
    else {
        System.out.println("Employee List:");
        for (Employee employee : employees) {
            System.out.println(employee);
        }
    }
}
private static void saveEmployeesToFile() {
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
        oos.writeObject(employees);
        System.out.println("Employees saved to file.");
    }
   catch (IOException e) {
```

```
            System.err.println("Error saving employees: " + e.getMessage());
        }
    }
    private static void loadEmployeesFromFile() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            employees = (ArrayList<Employee>) ois.readObject();
            System.out.println("Employees loaded from file.");
        }
        catch (FileNotFoundException e) {
            System.out.println("No previous data found, starting fresh.");
        }
        catch (IOException | ClassNotFoundException e) {
            System.err.println("Error loading employees: " + e.getMessage());
        }
    }
      }
```

4. **Output:**

```
Employee Management System
1. Add Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee Name: Anshika Goel
Enter Employee ID: 11678
Enter Employee Designation: Engineer
Enter Employee Salary: 500000
Employee added successfully.

Employee Management System
1. Add Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee Name: Pawan
Enter Employee ID: 280908
Enter Employee Designation: CA
Enter Employee Salary: 1000000
Employee added successfully.

Employee Management System
1. Add Employee
2. Display All Employees
3. Exit
Enter your choice: 2
Employee List:
Employee ID: 11678, Name: Anshika Goel, Designation: Engineer, Salary: 500000.0
Employee ID: 280908, Name: Pawan, Designation: CA, Salary: 1000000.0
```