Experiment-5

Student Name: Gautam Thakur
UID: 22BCS10628
Branch: BE-CSE
Section/Group: 22BCS-IOT-640-A
Semester: 6th
Date of Performance: 24/02/2025
Subject Name: PBLJ with Lab
Subject Code: 22CSH-359

1. **Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

2. **Problem Statements:**
   Problem 1.1: 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()).
   Problem 1.2: 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.
   Problem 1.3: 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.

3. **Implementation/Code:**

Problem 1.1

```
import java.util.ArrayList;

import java.util.List;
```

```
public class AutoboxingUnboxingSum {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();

        numbers.add(10);
        numbers.add(20);
        numbers.add(30);

        numbers.add(Integer.parseInt("40"));
        numbers.add(Integer.valueOf("50"));

        int sum = calculateSum(numbers);

        System.out.println("List of numbers: " + numbers);
        System.out.println("Sum of numbers: " + sum);
    }

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

```
    }
}
```

Problem 1.2:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
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("Student ID: " + id);
        System.out.println("Name: " + name);
```

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

public class StudentSerialization {
    private static final String FILE_NAME = "studentData.txt";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        serializeStudents();

        System.out.print("Enter the file name to load student data: ");
        String userInput = scanner.nextLine();

        try {
            List<Student> students = deserializeStudents(userInput);
            if (students != null) {
                System.out.println("\nDeserialized Student Data:");
                for (Student student : students) {
                    student.display();
                }
            }
```

```
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found! Please check the filename and try
again.");
        } catch (IOException e) {
            System.out.println("Error: Issue reading the file.");
        } catch (ClassNotFoundException e) {
            System.out.println("Error: Student class not found.");
        } finally {
            scanner.close();
        }
    }

    private static void serializeStudents() {
        List<Student> students = new ArrayList<>();
        students.add(new Student(101, "Gautam Thakur", 9.8));
        students.add(new Student(102, "Harsh Kumar", 9.5));
        students.add(new Student(103, "Rohit Kumar", 9.2));

        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            out.writeObject(students);
            System.out.println("Student objects successfully serialized into " +
FILE_NAME);
        } catch (IOException e) {
            System.out.println("Error: Unable to serialize student data.");
```

```
        }
    }

    private static List<Student> deserializeStudents(String fileName) throws
IOException, ClassNotFoundException {
        File file = new File(fileName);

        if (!file.exists()) {
            throw new FileNotFoundException();
        }

        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(file))) {
            return (List<Student>) in.readObject();
        }
    }
}
```

Problem 1.3:

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

class Employee implements Serializable {
    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;
    }

    @Override
    public String toString() {
        return "Employee ID: " + id + ", Name: " + name + ", Designation: " +
designation + ", Salary: $" + salary;
    }
}

public class EmployeeManagementSystem {
    private static final String FILE_NAME = "employees.dat";
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
```

```
        System.out.println("\nEmployee Management System");
        System.out.println("1. Add an 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 the application. Goodbye!");
                System.exit(0);
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
```

```
    private static void addEmployee() {
        System.out.print("Enter employee name: ");
        String name = scanner.nextLine();

        System.out.print("Enter employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        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);

        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME, true))) {
            oos.writeObject(employee);
            System.out.println("Employee added successfully!");
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
```

```
    }


    private static void displayAllEmployees() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            while (true) {
                try {
                    Employee employee = (Employee) ois.readObject();
                    System.out.println(employee);
                } catch (EOFException e) {
                    break; // End of file reached
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("No employees found. The file is empty or doesn't
exist.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error reading from file: " + e.getMessage());
        }
    }
}
```

4. Output:

```
PS D:\Semester-6\PROJECT BASED LEARNING IN JAVA WITH LAB\PBLJ With Lab-Code\Exp-5> cd "d:\Semester-6\PROJECT BASED LEARNING
h Lab-Code\Exp-5\" ; if ($?) { javac AutoboxingUnboxingSum.java } ; if ($?) { java AutoboxingUnboxingSum }
List of numbers: [10, 20, 30, 40, 50]
Sum of numbers: 150
PS D:\Semester-6\PROJECT BASED LEARNING IN JAVA WITH LAB\PBLJ With Lab-Code\Exp-5>
```

(Fig. 1- Problem 1.1 Output)

```
PS D:\Semester-6\PROJECT BASED LEARNING IN JAVA WITH LAB\PBLJ With Lab-Code\Exp-5> cd "d:\Semester-6\PROJECT BASED LE
h Lab-Code\Exp-5\" ; if ($?) { javac StudentSerialization.java } ; if ($?) { java StudentSerialization }
Note: StudentSerialization.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Student objects successfully serialized into studentData.txt
Enter the file name to load student data: student.txt
Error: File not found! Please check the filename and try again.
PS D:\Semester-6\PROJECT BASED LEARNING IN JAVA WITH LAB\PBLJ With Lab-Code\Exp-5>
```

(Fig. 2- Problem 1.2 Output)

```
Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter employee name: Gautam Thakur
Enter employee ID: 101
Enter employee designation: Director
Enter employee salary: 200000
Employee added successfully!

Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2
Employee ID: 101, Name: Gautam Thakur, Designation: Director, Salary: $200000.0
Error reading from file: invalid type code: AC

Employee Management System
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 3
Exiting the application. Goodbye!
PS D:\Semester-6\PROJECT BASED LEARNING IN JAVA WITH LAB\PBLJ With Lab-Code\Exp-5>
```

(Fig. 3- Problem 1.3 Output)

5. Learning Outcome:

1. Learned how Java automatically converts primitive data types into their corresponding wrapper classes and vice versa, improving efficiency in data handling.
2. Gained hands-on experience in storing and retrieving objects using Java's Serializable interface, handling exceptions like FileNotFoundException and ClassNotFoundException.
3. Developed skills in reading and writing data to files using FileInputStream, FileOutputStream, ObjectInputStream, and ObjectOutputStream.
4. Learned how to create interactive Java applications using a Scanner for user input, managing object persistence, and handling multiple operations through a structured menu.