

EXPERIMENT 5

Student Name: Kritika Sharma

UID: 22Bcs14943

Branch: BE – cse

Section/Group: cse_KRG_IOT-3B

Semester: 6th

Date: 25/02/2025

Subject Name: PBLJ With Lab

Subject Code: 22CSH-359

PROBLEM 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 respective wrapper classes (e.g., Integer.parseInt()).

Objective:

- To Demonstrate Autoboxing and Unboxing.
- To calculate sum of integers.

Implementation/Code:

```
import java.util.ArrayList; import
java.util.List; import java.util.Scanner;
public class SumUsingAutoboxing {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter numbers separated by space:");
        String input = scanner.nextLine();
        NumberProcessor processor = new NumberProcessor(input);
        int sum = processor.calculateSum();
        System.out.println("Sum of numbers: " + sum);
        scanner.close();
    }
}

class NumberProcessor {    private
List<Integer> integerList;    public
NumberProcessor(String input) {
    integerList = new ArrayList<>();
    String[] numbers = input.split(" ");
```

```
for (String num : numbers) {  
    integerList.add(Integer.parseInt(num));  
}  
}  
public int calculateSum() {  
    int sum = 0;  
    for (Integer num : integerList) {  
sum += num;  
    }  
    return sum;  
}  
}
```

Output:

```
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> javac .\Exp5\SumUsingAutoboxing.java  
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> java Exp5.SumUsingAutoboxing  
Enter numbers separated by space:  
7 8 9 6 54 2  
Sum of numbers: 86
```

Learning Outcomes:

- Learn how Java automatically converts between primitive types and their wrapper classes when adding/removing elements from collections.
- Gain experience in reading user input, splitting strings, and converting them into numerical values using Integer.parseInt().
- Learn how to store user-provided integers in an ArrayList, iterate through the list, and perform calculations using loops.

PROBLEM 2

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 **Objective:**

- To Convert a Student object into a binary format and store it in a file.
- To Retrieve the object from the file and reconstruct it using deserialization.

- To implement the Serializable interface to allow objects to be written to and read from a file.

Implementation/Code:

```
import java.io.*; public class
StudentSerialization {    public static
void main(String[] args) {
    Student student = new Student(10239, "Ruchi Thakur", 8.0);
    String filename = "student.ser";    serializeStudent(student,
filename);
    Student deserializedStudent = deserializeStudent(filename);
    if (deserializedStudent != null) {
        System.out.println("\nDeserialized Student Details:");
        deserializedStudent.display();
    }
}
    public static void serializeStudent(Student student, String filename) {
    try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(filename))) {        oos.writeObject(student);
        System.out.println("Student object serialized successfully.");
    } catch (IOException e) {
        System.out.println("Error during serialization: " + e.getMessage());
    }
}
    public static Student deserializeStudent(String filename) {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
    return (Student) ois.readObject();
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + filename);
        } catch (IOException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Class not found error: " + e.getMessage());
        }
        return null;
    }
}
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);
    }
}
```

Output:

```
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> javac .\Exp5\StudentSerialization.java
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> java Exp5.StudentSerialization
Student object serialized successfully.

Deserialized Student Details:
Student ID: 10239
Name: Ruchi Thakur
GPA: 8.0
```

Learning Outcomes:

- Learn how to convert a Java object into a byte stream for storage or transmission (serialization).
- Understand how to retrieve the object back from the byte stream (deserialization).
- Understand the role of serialVersionUID in maintaining object compatibility during deserialization.
- Learn how to write and read objects to and from a file using ObjectOutputStream and ObjectInputStream.

PROBLEM 3

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

Objective:

- To read from and write to a file using FileWriter, BufferedWriter, and PrintWriter for storing and retrieving employee data.
- designing user-friendly menu-based applications using loops and switch cases for handling user choices.
- To take structured user input (such as integers, strings, and doubles) and process it correctly to avoid common input-related errors.

Implementation/Code:

```
import java.util.*;
public class EmployeeManagement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nMenu:");
            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();
            switch (choice) {
                case 1:
                    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 newEmployee = new Employee(id, name, designation, salary);
                    EmployeeManager.addEmployee(newEmployee);
                    break;
                case 2:
                    EmployeeManager.displayEmployees();
                    break;
                case 3:
```

```
        System.out.println("Exiting program...");
scanner.close();          System.exit(0);
default:
    System.out.println("Invalid choice! Please enter 1, 2, or 3.");
    }
    }
    }
}

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 int getId() { return id; }  public String
getName() { return name; }  public String
getDesignation() { return designation; }  public
double getSalary() { return salary; }  public String
toString() {
    return "ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " +
salary;
    }
}

class EmployeeManager {
    private static final String FILE_NAME = "employees.dat";
    public static void addEmployee(Employee employee) {
List<Employee> employees = getEmployees();
employees.add(employee);
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
oos.writeObject(employees);
            System.out.println("Employee added successfully!");
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }
}
```

```
}

    public static List<Employee> getEmployees() {
List<Employee> employees = new ArrayList<>();    try
(ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
    employees = (List<Employee>) ois.readObject();
} catch (FileNotFoundException e) {
    System.out.println("No employee records found.");
} catch (IOException | ClassNotFoundException e) {
    System.out.println("Error reading file: " + e.getMessage());
}
    return employees;
}

    public static void displayEmployees() {
List<Employee> employees = getEmployees();    if
(employees.isEmpty()) {
    System.out.println("No employees to display.");
return;
}
    System.out.println("\nEmployee Details:");
for (Employee emp : employees) {
    System.out.println(emp);
}
}
}
```

Output:

```
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> java Exp5.EmployeeManagement

Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 10239
Enter Employee Name: Ruchi Thakur
Enter Designation: SDE 3
Enter Salary: 175000
No employee records found.
Employee added successfully!

Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2

Employee Details:
ID: 10239, Name: Ruchi Thakur, Designation: SDE 3, Salary: 175000.0

Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 3
Exiting program...
```

Learning Outcomes:

- Understand Object-Oriented Programming (OOP) principles in Java.
- Learn file handling for storing and retrieving employee data.
- Implement exception handling to manage errors effectively.
- Use serialization and deserialization for object persistence.
- Develop a menu-driven program for user interaction.
- Apply BufferedReader and PrintWriter for efficient file operations.
- Gain hands-on experience in handling user input with Scanner.