Experiment-5

Student Name: Anshuman Raj **UID:** 22BET10081
Branch: BE-IT **Section:** 22BET_IOT-702 'A'
Semester: 6th **Date of Performance:** 19/02/25
Subject Name: PBLJ Lab **Subject Code:** 22ITH-359

Problem 1

1. Aim:

To implement a Java program that calculates the sum of a list of integers using autoboxing and unboxing.

2. Objective:

1 To understand the concept of autoboxing and unboxing in Java.
2 To convert a list of strings into integer values using Integer.parseInt().
3 To perform arithmetic operations using autoboxed values.
4 To enhance user interaction by accepting input dynamically.

3. Code:

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

public class raj13 {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
ArrayList<Integer> numbers = new ArrayList<>();

System.out.print("Enter the number of elements: ");
int n = scanner.nextInt();
```

```
System.out.println("Enter " + n + " integers:");
for (int i = 0; i < n; i++) {
numbers.add(scanner.nextInt());
}

int sum = 0;
for (Integer num : numbers) {
sum += num;
}

System.out.println("Sum of the integers: " + sum);
scanner.close();
}
}
```

4. Output:

```
<terminated> raj13 [Java Application] C:\Users\ANSHUMAN\.p2\p
Enter the number of elements: 6
Enter 6 integers:
2
5
8
9
10
4
Sum of the integers: 38
```

Fig. shows output for sum

5. Learning Outcomes:

1. Understand autoboxing (automatic conversion of int to Integer) and unboxing (Integer to int) in Java.
2. Learn how wrapper classes (Integer) work with collections like ArrayList<Integer>.
3. Gain experience in handling dynamic user input and storing values in a list.
4. Develop skills in using enhanced for-loops for efficient iteration and sum calculation.

Problem 2

1.Aim:

Create a program to use lambda expressions and stream operations to filter students scoring above 75%, sort them by marks, and display their names.

2.Objective:

1 To understand the process of serializing and deserializing objects in Java.

2 To implement the Serializable interface in a Java class.

3 To write and read objects from a file using ObjectOutputStream and ObjectInputStream.

4 To handle exceptions that may arise during serialization and deserialization.

3.Code:

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

class Person implements Serializable {
private static final long serialVersionUID = 1L;
private String name;
private int age;

public Person(String name, int age) {
this.name = name;
this.age = age;
}
public void display() {
System.out.println("Name: " + name + ", Age: " + age);
}
}

public class raj13 {
private static final String FILE_NAME = "person_data.ser";

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

```
System.out.print("Enter name: ");
String name = scanner.nextLine();

System.out.print("Enter age: ");
int age = scanner.nextInt();

Person person = new Person(name, age);
serializeObject(person);
Person deserializedPerson = deserializeObject();
if (deserializedPerson != null) {
System.out.println("Deserialized Object Data:");
deserializedPerson.display();
}
scanner.close();
}
private static void serializeObject(Person person) {
try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
oos.writeObject(person);
System.out.println("Object serialized successfully!");
} catch (IOException e) {
e.printStackTrace();
}
}
private static Person deserializeObject() {
try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
return (Person) ois.readObject();
} catch (IOException | ClassNotFoundException e) {
e.printStackTrace();
}
return null;
}
}
```

4.Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> raj13 [Java Application] C:\Users\ANSHUMAN\.p2\p
Enter name: Anshuman
Enter age: 22
Object serialized successfully!
Deserialized Object Data:
Name: Anshuman, Age: 22
```

Fig. shows serialization & deserialisation

5.Learning Outcomes:

1 Understood the concept of object serialization and deserialization.

2 Implemented the Serializable interface in Java.

3 Learned how to write and read objects using file streams.

4 Explored exception handling related to file operations.

5 Developed an understanding of how objects can be stored persistently with user input.

Problem 3

1.Aim:

To develop a menu-based Java application for managing employee details using file handling and serialization.

2.Objective:

1 To create a menu-driven program with options for adding and displaying employee details.

2 To use serialization to store and retrieve employee records from a file.

3 To implement file handling techniques in Java.

4 To improve user interaction through an efficient menu system.

3.Code:

```
import java.io.*;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Employee implements Serializable {

private static final long serialVersionUID = 1L;

private int empID;

private String name;

private double salary;

public Employee(int empID, String name, double salary) {

this.empID = empID;

this.name = name;

this.salary = salary;

}
```

```
public int getEmpID() { return empID; }

public String getName() { return name; }

public double getSalary() { return salary; }

@Override

public String toString() {

return "EmpID: " + empID + ", Name: " + name + ", Salary: " + salary;

}

}

public class EmployeeManagement {

private static final String FILE_NAME = "employees.ser";

private static List<Employee> employeeList = new ArrayList<>();

public static void main(String[] args) {

loadEmployees();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Add Employee");

System.out.println("2. View Employees");

System.out.println("3. Delete Employee");

System.out.println("4. Exit");

System.out.print("Enter choice: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:
```

```
addEmployee(scanner);

break;

case 2:

viewEmployees();

break;

case 3:

deleteEmployee(scanner);

break;

case 4:

saveEmployees();

System.out.println("Exiting program...");

scanner.close();

System.exit(0);

default:

System.out.println("Invalid choice! Try again.");

}

}

}

private static void addEmployee(Scanner scanner) {

System.out.print("Enter Employee ID: ");

int empID = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Employee Name: ");

String name = scanner.nextLine();

System.out.print("Enter Employee Salary: ");
```

```
double salary = scanner.nextDouble();

employeeList.add(new Employee(empID, name, salary));

saveEmployees();

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

private static void viewEmployees() {

if (employeeList.isEmpty()) {

System.out.println("No employees found.");

} else {

System.out.println("\nEmployee List:");

for (Employee emp : employeeList) {

System.out.println(emp);

}

}

}

private static void deleteEmployee(Scanner scanner) {

System.out.print("Enter Employee ID to delete: ");

int empID = scanner.nextInt();

boolean removed = employeeList.removeIf(emp -> emp.getEmpID() == empID);

if (removed) {

saveEmployees();

System.out.println("Employee deleted successfully.");

private static void saveEmployees() {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {

oos.writeObject(employeeList);

} catch (IOException e) {
```

```
e.printStackTrace();

}

private static void loadEmployees() {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {

employeeList = (List<Employee>) ois.readObject();

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}
```

4.Output:

```
Problems  Javadoc  Declaration  Console ×
exp3 (1) [Java Application] C:\Users\ANSHUMAN\.p2\pool\plugins\org.ed

Menu:
1. Add Employee
2. View Employees
3. Delete Employee
4. Exit
Enter choice: 1
Enter Employee ID: 10081
Enter Employee Name: Anshuman
Enter Employee Salary: 75000
Employee added successfully!

Menu:
1. Add Employee
2. View Employees
3. Delete Employee
4. Exit
Enter choice: 1
Enter Employee ID: 2003
Enter Employee Name: Raj
Enter Employee Salary: 39403
Employee added successfully!

Menu:
1. Add Employee
2. View Employees
3. Delete Employee
4. Exit
Enter choice: 2

Employee List:
EmpID: 10081, Name: Anshuman, Salary: 75000.0
EmpID: 2003, Name: Raj, Salary: 39403.0
```

Fig. shows employee management system

5.Learning Outcomes:

1. Understand File Handling & Persistence: Learn how to store and retrieve data using serialization and deserialization.
2. Implement Object-Oriented Programming (OOP): Work with classes, objects, and collections (List).
3. Use Serialization (Serializable Interface): Convert objects into a byte stream for saving and loading data.
4. Apply CRUD Operations: Gain hands-on experience in Create, Read, Update, and Delete (CRUD) functionalities.
5. Develop Menu-Driven Applications: Learn how to build interactive console-based programs for real-world applications.