Experiment-2.2

Student Name: KARAN SINGH RANA
UID: 22BCS16520
Branch: BE-CSE
Section/Group: KPIT-902/B
Semester: 6th
Date of Performance: 27/02/25
Subject Name: PBLJ-Lab
Subject Code: 22CSH-359

1. Aim:

Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

a.) 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()).

b.) 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.

c.) 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. Objective:

a.) Autoboxing and Unboxing in Java:
This program demonstrates autoboxing (automatic conversion of primitive types to their wrapper classes) and unboxing (conversion of wrapper class objects back to primitive types). It reads a list of integers as strings, converts them into their wrapper class (Integer), sums them up, and prints the result.

b.) Serialization and Deserialization of a Student Object:
This program demonstrates how to serialize (convert an object into a byte stream and save it to a file) and deserialize (retrieve the object from the file and reconstruct it). It

ensures that a student object with ID, name, and GPA can be saved and retrieved while handling file-related exceptions.

c.) Menu-Based Employee Management System:
This program provides a simple file-based employee management system. It allows users to:
Add Employee – Gather employee details and store them in a file.
Display All Employees – Retrieve and display all stored employee records.
Exit – Close the application safely.

3. Implementation/Code:

a.)

```
import java.util.*;

public class AutoboxingExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter integers separated by spaces:");
        String input = scanner.nextLine();

        String[] numbers = input.split(" ");
        List<Integer> intList = new ArrayList<>();

        for (String num : numbers) {
            intList.add(Integer.parseInt(num));
        }

        int sum = 0;

        for (Integer num : intList) {
            sum += num;
        }

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

b.)

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

class Student implements Serializable {
    private static final long serialVersionUID = 1L; // Ensures compatibility
    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("\nDeserialized Student Details:");
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("GPA: " + gpa);
    }
}

public class StudentSerialization {
    private static final String FILE_NAME = "student.ser";

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

        System.out.print("Enter Student ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Student Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Student GPA: ");
        double gpa = scanner.nextDouble();
        Student student = new Student(id, name, gpa);

        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            out.writeObject(student);
            System.out.println("\nStudent object serialized successfully.");
        } catch (IOException e) {
            System.err.println("Error during serialization: " + e.getMessage());
        }

        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            Student deserializedStudent = (Student) in.readObject();
            deserializedStudent.display();
        } catch (FileNotFoundException e) {
            System.err.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("IO error: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Class not found: " + e.getMessage());
        }

        scanner.close();
    }
}
```

c.)

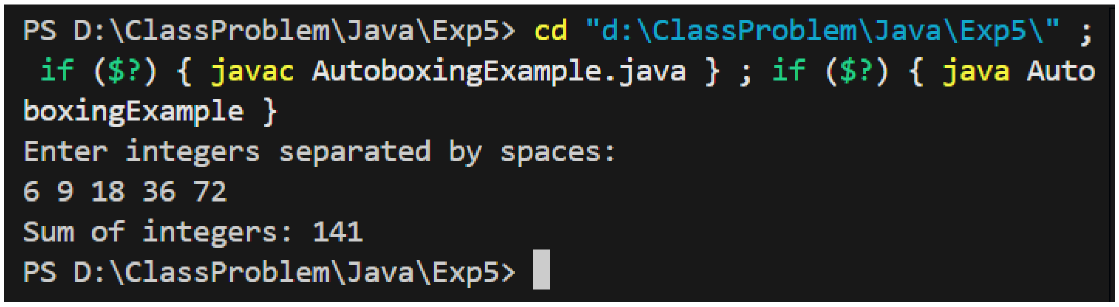
```
import java.io.*;
import java.util.*;

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

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

public class EmployeeManagementSystem {
    private static final String FILE_NAME = "employees.dat";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> employees = loadEmployees();

        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:
                    addEmployee(scanner, employees);
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
```

```
                System.out.println("Exiting program...");
                saveEmployees(employees);
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
}

private static void addEmployee(Scanner scanner, List<Employee> employees) {
    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();

    employees.add(new Employee(id, name, designation, salary));
    saveEmployees(employees);
    System.out.println("Employee added successfully.");
}

private static void displayEmployees() {
    List<Employee> employees = loadEmployees();
    if (employees.isEmpty()) {
        System.out.println("No employees found.");
    } else {
        System.out.println("Employee List:");
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
```

```
    private static void saveEmployees(List<Employee> employees) {
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            out.writeObject(employees);
        } catch (IOException e) {
            System.err.println("Error saving employees: " + e.getMessage());
        }
    }

    private static List<Employee> loadEmployees() {
        File file = new File(FILE_NAME);
        if (!file.exists())
            return new ArrayList<>();

        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            return (List<Employee>) in.readObject();
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Error loading employees: " + e.getMessage());
            return new ArrayList<>();
        }
    }
}
```

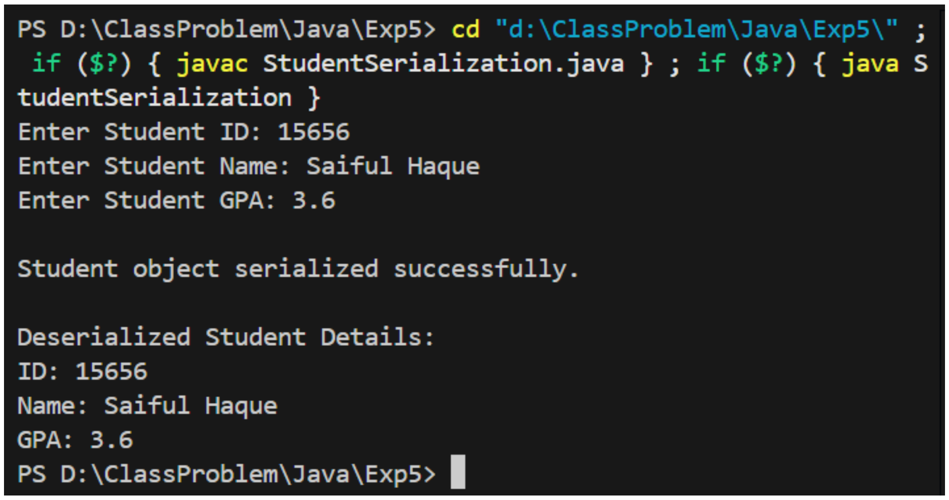
4. Output:

a.)



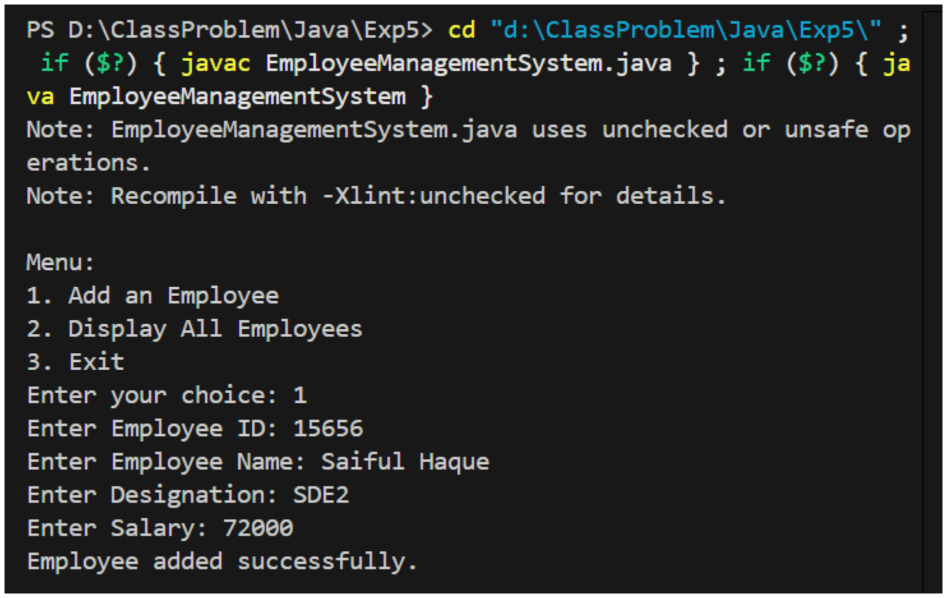
```
PS D:\ClassProblem\Java\Exp5> cd "d:\ClassProblem\Java\Exp5\" ;
 if ($?) { javac AutoboxingExample.java } ; if ($?) { java Auto
boxingExample }
Enter integers separated by spaces:
6 9 18 36 72
Sum of integers: 141
PS D:\ClassProblem\Java\Exp5>
```

b.)

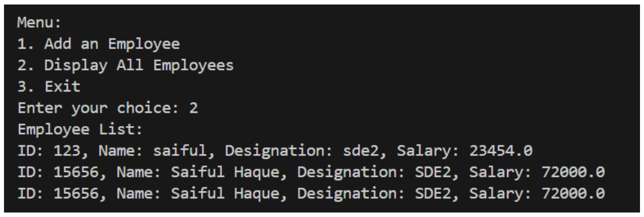


```
PS D:\ClassProblem\Java\Exp5> cd "d:\ClassProblem\Java\Exp5\" ;
 if ($?) { javac StudentSerialization.java } ; if ($?) { java S
tudentSerialization }
Enter Student ID: 15656
Enter Student Name: Saiful Haque
Enter Student GPA: 3.6

Student object serialized successfully.

Deserialized Student Details:
ID: 15656
Name: Saiful Haque
GPA: 3.6
PS D:\ClassProblem\Java\Exp5>
```

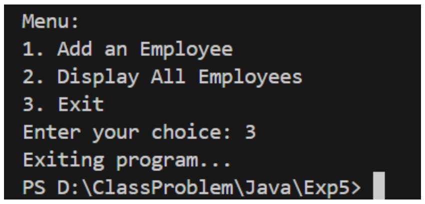
c.)



```
PS D:\ClassProblem\Java\Exp5> cd "d:\ClassProblem\Java\Exp5\" ;
 if ($?) { javac EmployeeManagementSystem.java } ; if ($?) { ja
va EmployeeManagementSystem }
Note: EmployeeManagementSystem.java uses unchecked or unsafe op
erations.
Note: Recompile with -Xlint:unchecked for details.

Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 15656
Enter Employee Name: Saiful Haque
Enter Designation: SDE2
Enter Salary: 72000
Employee added successfully.
```



```
Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2
Employee List:
ID: 123, Name: saiful, Designation: sde2, Salary: 23454.0
ID: 15656, Name: Saiful Haque, Designation: SDE2, Salary: 72000.0
ID: 15656, Name: Saiful Haque, Designation: SDE2, Salary: 72000.0
```



```
Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 3
Exiting program...
PS D:\ClassProblem\Java\Exp5>
```

5. Learning Outcome:

1. Learned autoboxing and unboxing to automatically convert between primitive types and wrapper classes.
2. Used Integer.parseInt() to convert strings into numeric values.
3. Practiced Java serialization to save and retrieve objects from a file.
4. Handled exceptions like IOException, ClassNotFoundException, and FileNotFoundException.
5. Implemented file handling to store and manage employee records.
6. Created a menu-driven system for adding, displaying, and managing employee data.
7. Used object-oriented programming concepts like classes, objects, and encapsulation.
8. Improved data processing skills by efficiently reading and writing structured data.