Experiment-5

Student Name: Pratham
UID: 22BCS14521
Branch: BE-CSE
Section/Group: KPIT-902/B
Semester: 6th
Date of Performance: 28/02/25
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 handlingCreate a program to collect and store all the cards to assist the users in finding all the cards in a given symbol using Collection interface.

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:

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

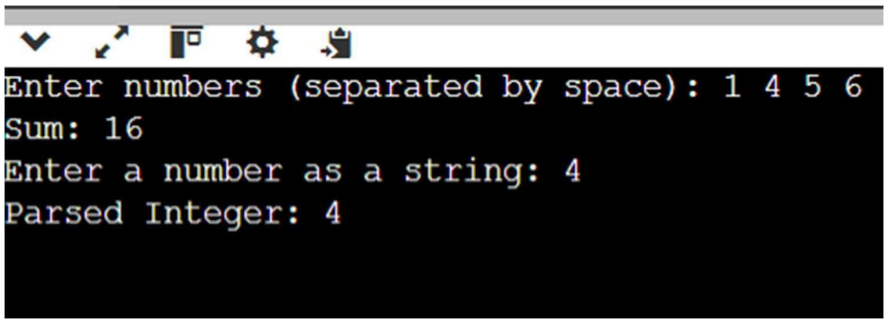
1. Implement autoboxing/unboxing to compute the sum of integers from a list and parse strings to wrapper classes.
2. Serialize a Student object (id, name, GPA) to a file and deserialize it back, handling exceptions properly.
3. Create a menu-driven employee management system that adds, stores, and retrieves employee data using file handling.

3. Implementation/Code:

a) Sum of Integers

```
import java.util.*;

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

    public static void main(String[] args)
        { Scanner sc = new Scanner(System.in);
        List<Integer> nums = new ArrayList<>();
        System.out.print("Enter numbers (separated by space): ");
        String[] inputs = sc.nextLine().split(" ");
        for (String str : inputs) nums.add(Integer.parseInt(str));

        System.out.println("Sum: " + calculateSum(nums));

        System.out.print("Enter a number as a string: ");
        String strNum = sc.next();
        int parsedNum = Integer.parseInt(strNum);
        System.out.println("Parsed Integer: " + parsedNum);
    }
}
```

Output :

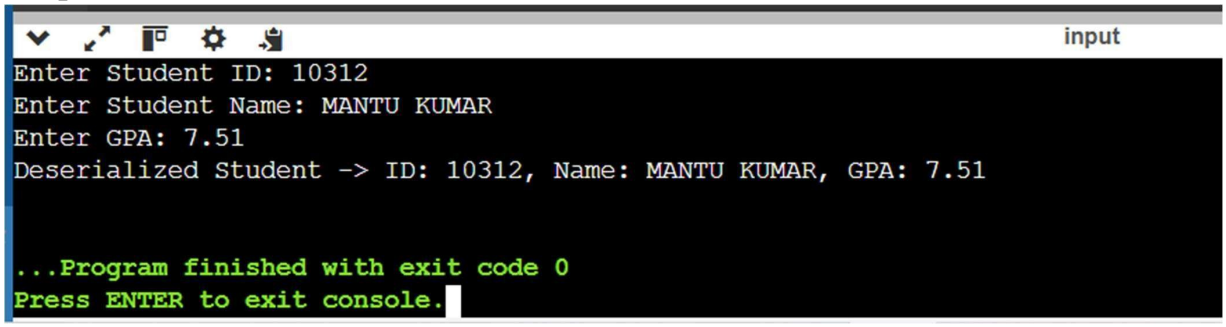


b) Student managenment:

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

class Student implements Serializable
   { int id;
   String name;
   double gpa;
   Student(int id, String name, double gpa) { this.id = id; this.name = name; this.gpa =
gpa; }
}

public class StudentSerialization {
   static final String FILE = "student.dat";

   public static void main(String[] args)
      { Scanner sc = new Scanner(System.in);
      System.out.print("Enter Student ID: "); int id = sc.nextInt();
      sc.nextLine();
      System.out.print("Enter Student Name: "); String name = sc.nextLine();
      System.out.print("Enter GPA: "); double gpa = sc.nextDouble();

      try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILE))) {
           out.writeObject(new Student(id, name, gpa));
       } catch (IOException e) { e.printStackTrace(); }

      try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE)))
         { Student s = (Student) in.readObject();
         System.out.println("Deserialized Student -> ID: " + s.id + ", Name: " + s.name +
```

```
", GPA: " + s.gpa);
        } catch (IOException | ClassNotFoundException e) { e.printStackTrace(); }
    }
}
```

Output:

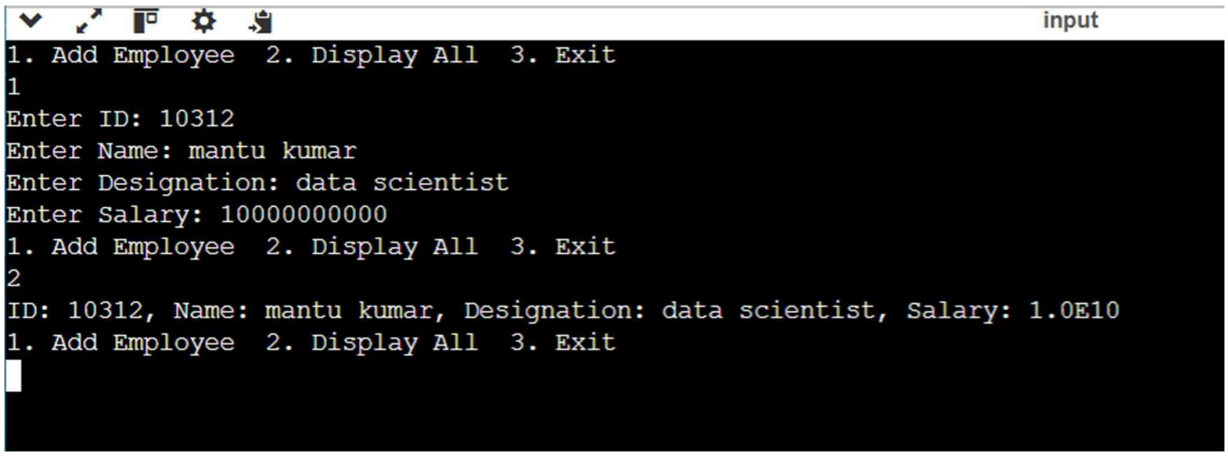


c) Employee Management System:

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

class Employee implements Serializable {
    int id; String name, designation; double salary;
    Employee(int id, String name, String designation, double salary) { this.id = id;
this.name = name; this.designation = designation; this.salary = salary; }
}

public class EmployeeManagement {
    static final String FILE = "employees.dat";

    public static void addEmployee() {
        try (FileOutputStream fos = new FileOutputStream(FILE, true);
            ObjectOutputStream out = new ObjectOutputStream(fos))
            { Scanner sc = new Scanner(System.in);
            System.out.print("Enter ID: "); int id = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter Name: "); String name = sc.nextLine();
            System.out.print("Enter Designation: "); String desig = sc.nextLine();
            System.out.print("Enter Salary: "); double salary = sc.nextDouble();
            out.writeObject(new Employee(id, name, desig, salary));
        } catch (IOException e) { e.printStackTrace(); }
```

```
    }

    public static void displayEmployees() {
        try (FileInputStream fis = new FileInputStream(FILE);
            ObjectInputStream in = new ObjectInputStream(fis))
            { while (true) {
                Employee e = (Employee) in.readObject();
                System.out.println("ID: " + e.id + ", Name: " + e.name + ", Designation: " +
e.designation + ", Salary: " + e.salary);
            }
        } catch (EOFException e) {} catch (IOException | ClassNotFoundException e)
{ e.printStackTrace(); }
    }

    public static void main(String[] args)
        { Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("1. Add Employee  2. Display All  3. Exit");
            int choice = sc.nextInt();
            if (choice == 1) addEmployee();
            else if (choice == 2) displayEmployees();
            else break;
        }
    }
}
```

Output :



4. Learning Outcome:

A. Understand autoboxing and unboxing for seamless conversion between primitive types and wrapper classes.

B. Implement serialization and deserialization to persist and retrieve Java objects efficiently.

C. Utilize file handling for storing and managing structured data in a menu-driven application.

D. Apply exception handling to manage runtime errors and ensure program stability.

E. Develop interactive Java applications with user input processing and data persistence.