# Experiment 5.1

|  |  |
| --- | --- |
| **Student Name: Aditi R Sinha UID: 22BCS15130**  **Branch: CSE Semester: 6th**  **Subject: PBLJ** | **Section: KRG 2B**  **DOP:25/02/25**  **Subject Code:22CSH-359** |

**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:** The goal of this Java program is to demonstrate autoboxing and unboxing while calculating the sum of a list of integers.

**Code:**

import java.util.\*;

public class autoboxing { public static List<Integer> parseStringToIntegers(List<String> strNumbers) {

List<Integer> intNumbers = new ArrayList<>();

for (String num : strNumbers) {

intNumbers.add(Integer.parseInt(num));

}

return intNumbers;

}

public static int calculateSum(List<Integer> numbers) { int sum = 0;

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

}

return sum;

}

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

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

scanner.nextLine();

List<String> strNumbers = new ArrayList<>(); System.out.println("Enter " + n + " numbers:");

for (int i = 0; i < n; i++) {

strNumbers.add(scanner.nextLine());

}

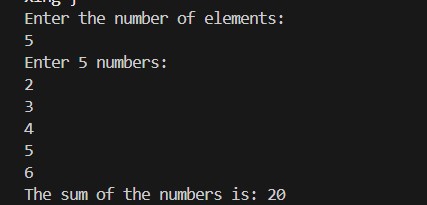
List<Integer> numbers = parseStringToIntegers(strNumbers); int sum = calculateSum(numbers);

System.out.println("The sum of the numbers is: " + sum);

scanner.close();

} }

**Output**:



**Learning Outcomes:**

* Understand the concept of autoboxing and unboxing in Java and how primitive types are automatically converted to their wrapper classes and vice versa.
* Learn how to convert string values into Integer objects using Integer.parseInt() and store them in a list.
* Gain experience in working with ArrayLists to store and manipulate a collection of numbers dynamically.
* Develop proficiency in iterating through collections and performing arithmetic operations like summation.

# Experiment 5.2

1.**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.

1. **Objective:** The objective is to serialize and deserialize a Student object, store and retrieve its id, name, and GPA from a file, and handle exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

1. **Implementation Code:**

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

class Student implements Serializable {

static final long serialVersionUID = 1L; int id;

String name;

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);

}

}

public class StudentSerialization {

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.err.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.err.println("File not found: " + e.getMessage());

} catch (IOException e) {

System.err.println("Error during deserialization: " + e.getMessage());

} catch (ClassNotFoundException e) {

System.err.println("Class not found: " + e.getMessage());

}

return null;

}

public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.println("Enter Student ID:"); int id = scanner.nextInt(); scanner.nextLine();

System.out.println("Enter Student Name:");

String name = scanner.nextLine(); System.out.println("Enter Student GPA:");

double gpa = scanner.nextDouble();

Student student = new Student(id, name, gpa); String filename = "student.ser";

serializeStudent(student, filename);

Student deserializedStudent = deserializeStudent(filename);

if (deserializedStudent != null) {

System.out.println("Deserialized Student:"); deserializedStudent.display();

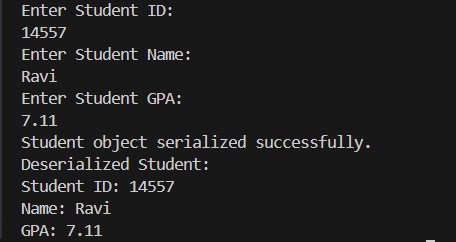
}

scanner.close();

}

}

1. **Output**



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

1. **Learning Outcomes:**

* + Understand object serialization and deserialization in Java.
  + Learn how to use ObjectOutputStream and ObjectInputStream for file operations.
  + Implement exception handling for FileNotFoundException, IOException, and ClassNotFoundException.
  + Gain hands-on experience in storing and retrieving objects from a file.
  + Develop skills in data persistence and file management using Java.

**Experiment 5.3**

* + 1. **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.

* + 1. Objective: The objective of this Java application is to create a **simple** menu-driven employee management **system** using file handling for data persistence.

* + 1. **Implementation Code:**

import java.io.\*;

import java.util.\*;

class Employee {

int id;

String name; String designation;

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 + "," + name + "," + designation + "," + salary;

}

public static Employee fromString(String line) { String[] parts = line.split(",");

return new Employee(Integer.parseInt(parts[0]), parts[1], parts[2],

Double.parseDouble(parts[3]));

}

}

public class EmployeeManagement {

static final String FILE\_NAME = "employees.txt";

public static void addEmployee() {

Scanner scanner = new Scanner(System.in); 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 employee = new Employee(id, name, designation, salary); try (FileWriter fw = new FileWriter(FILE\_NAME, true);

BufferedWriter bw = new BufferedWriter(fw); PrintWriter pw = new PrintWriter(bw)) { pw.println(employee); } catch (IOException e) {

System.err.println("Error saving employee data: " + e.getMessage());

}

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

}

public static void displayAllEmployees() { File file = new File(FILE\_NAME);

if (!file.exists()) {

System.out.println("No employee records found."); return;

}

try (BufferedReader br = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

while ((line = br.readLine()) != null) {

Employee emp = Employee.fromString(line);

System.out.println("ID: " + emp.id + ", Name: " + emp.name + ", Designation: " + emp.designation + ", Salary: " + emp.salary);

}

} catch (IOException e) {

System.err.println("Error reading employee data: " + e.getMessage());

}

}

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

System.out.println("\n1. Add Employee");

System.out.println("2. Display All Employees");

System.out.println("3. Exit");

System.out.print("Choose an option: "); int choice = scanner.nextInt();

switch (choice) { case 1:

addEmployee();

break; case 2:

displayAllEmployees();

break; case 3:

System.out.println("Exiting the application..."); scanner.close();

return; default:

System.out.println("Invalid option, try again.");

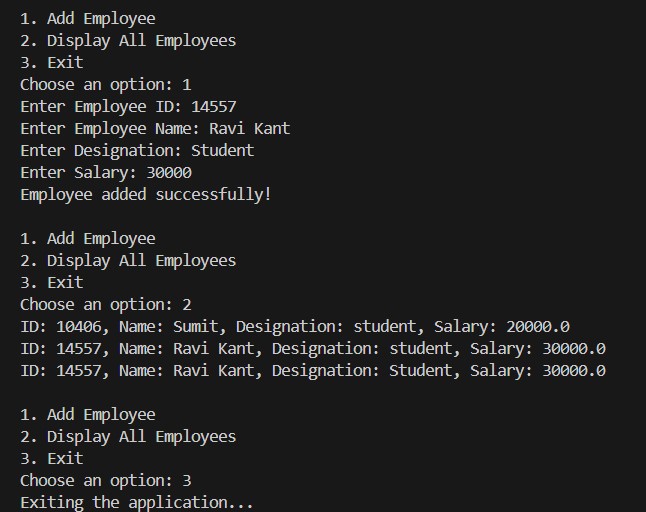
}

}

}

}

1. **Output:**



1. **Learning Outcomes:** 
   * Understand file handling and serialization in Java to store and retrieve objects persistently.
   * Learn how to implement a menu-driven console application using loops and conditional statements.
   * Gain experience in object-oriented programming (OOP) by defining and managing Employee objects.
   * Practice exception handling to manage file-related errors like FileNotFoundException and IOException.
   * Develop skills in list manipulation and user input handling using ArrayList and Scanner.