# Experiment 5.1

**Student Name: Priyanshu Kumar UID: 22BET10235**

**Branch: BE-IT Section: 22BET\_IOT-703/A**

**Semester: 6th DOP:21/02/25**

**Subject: PBLJ Subject Code:22ITH-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:** Demonstrate **autoboxing** and **unboxing** in Java by converting string numbers into Integer objects, storing them in a list, and computing their sum.

**Code:**

import java.util.ArrayList;

import java.util.List;

public class AutoboxingExample { public static void main(String[] args) {

String[] numberStrings = {"100", "200", "300", "400", "500"};

List<Integer> numbers = parseStringArrayToIntegers(numberStrings);

int sum = calculateSum(numbers);

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

}

public static List<Integer> parseStringArrayToIntegers(String[] strings) {

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

for (String str : strings) {

integerList.add(Integer.parseInt(str));

}

return integerList;

}

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

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

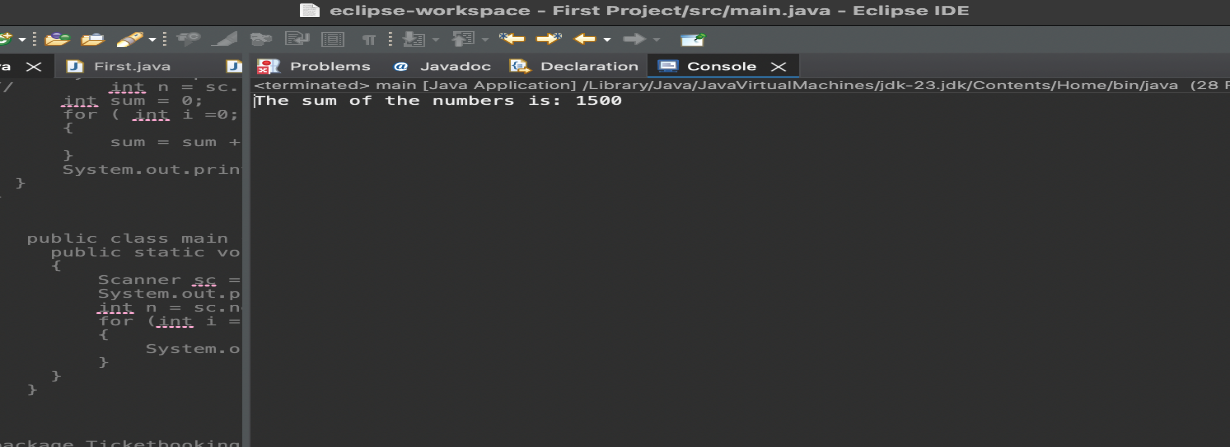
}

return sum;

}

}

**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.

**2.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.

**3. Implementation Code:**

import java.io.\*;

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;

}

@Override

public String toString() {

return "Student{id=" + id + ", name='" + name + "', gpa=" + gpa + "}";

}

}

public class StudentSerialization {

private static final String FILE\_NAME = "student.ser";

public static void main(String[] args) {

Student student = new Student(1, "Anwar", 7.8);

serializeStudent(student);

deserializeStudent();

}

public static void serializeStudent(Student student) {

try (ObjectOutputStream oos = new ObjectOutputStream(new

FileOutputStream(FILE\_NAME))) {

oos.writeObject(student);

System.out.println("Student object serialized successfully.");

} catch (FileNotFoundException e) {

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

} catch (IOException e) {

System.err.println("IOException occurred: " + e.getMessage());

}

}

public static void deserializeStudent() {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE\_NAME))) {

Student student = (Student) ois.readObject();

System.out.println("Deserialized Student: " + student);

} catch (FileNotFoundException e) {

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

} catch (IOException e) {

System.err.println("IOException occurred: " + e.getMessage());

} catch (ClassNotFoundException e) {

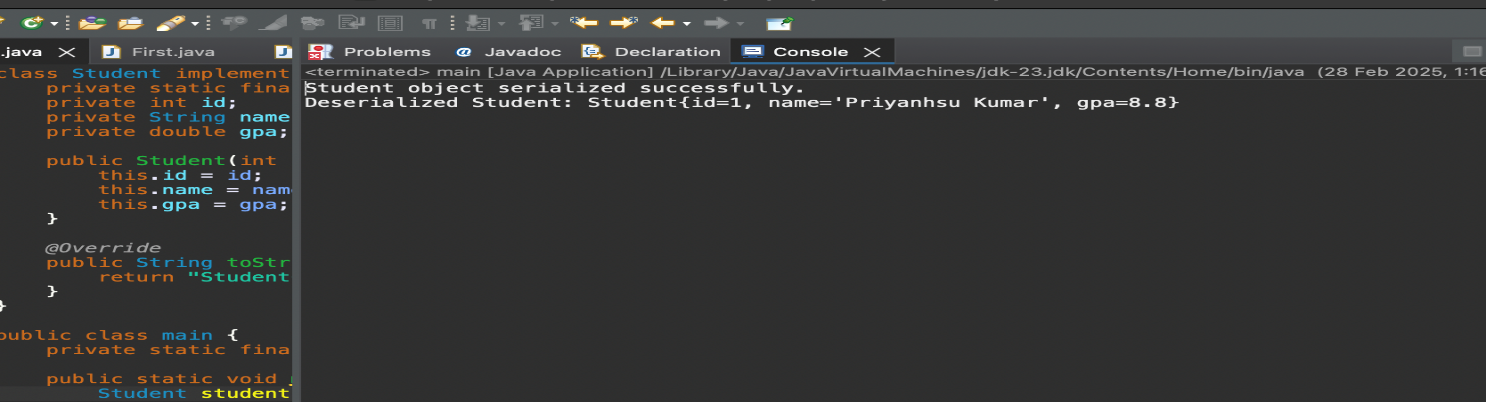
System.err.println("Class not found: " + e.getMessage());

}

}

}

**4.Output**



**6.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

1. Display All
2. 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.
3. **Objective:** The objective is to develop a menu-based Java application that allows users to **add employee details**, **store them in a file**, and **display all stored employee records**, with an option to exit the program.

**3.Implementation Code:**

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

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;

}

@Override

public String toString() {

return "Employee ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;

}

}

public class EmployeeManagementSystem { private static final String FILE\_NAME = "employees.ser"; private static List<Employee> employees = new ArrayList<>();

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);

employees.add(employee);

saveEmployees();

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

}

public static void displayAllEmployees() {

loadEmployees(); if (employees.isEmpty()) {

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

} else {

for (Employee employee : employees) {

System.out.println(employee);

}

}

}

private static void saveEmployees() {

try (ObjectOutputStream oos = new ObjectOutputStream(new

FileOutputStream(FILE\_NAME))) {

oos.writeObject(employees);

} catch (IOException e) {

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

}

}

@SuppressWarnings("unchecked") private static void loadEmployees() {

try (ObjectInputStream ois = new ObjectInputStream(new

FileInputStream(FILE\_NAME))) {

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

} catch (FileNotFoundException e) { employees = new ArrayList<>();

} catch (IOException | ClassNotFoundException e) {

System.err.println("Error loading employees: " + e.getMessage());

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\nEmployee Management System");

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(); break; case 2:

displayAllEmployees(); break; case 3:

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

return; default:

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

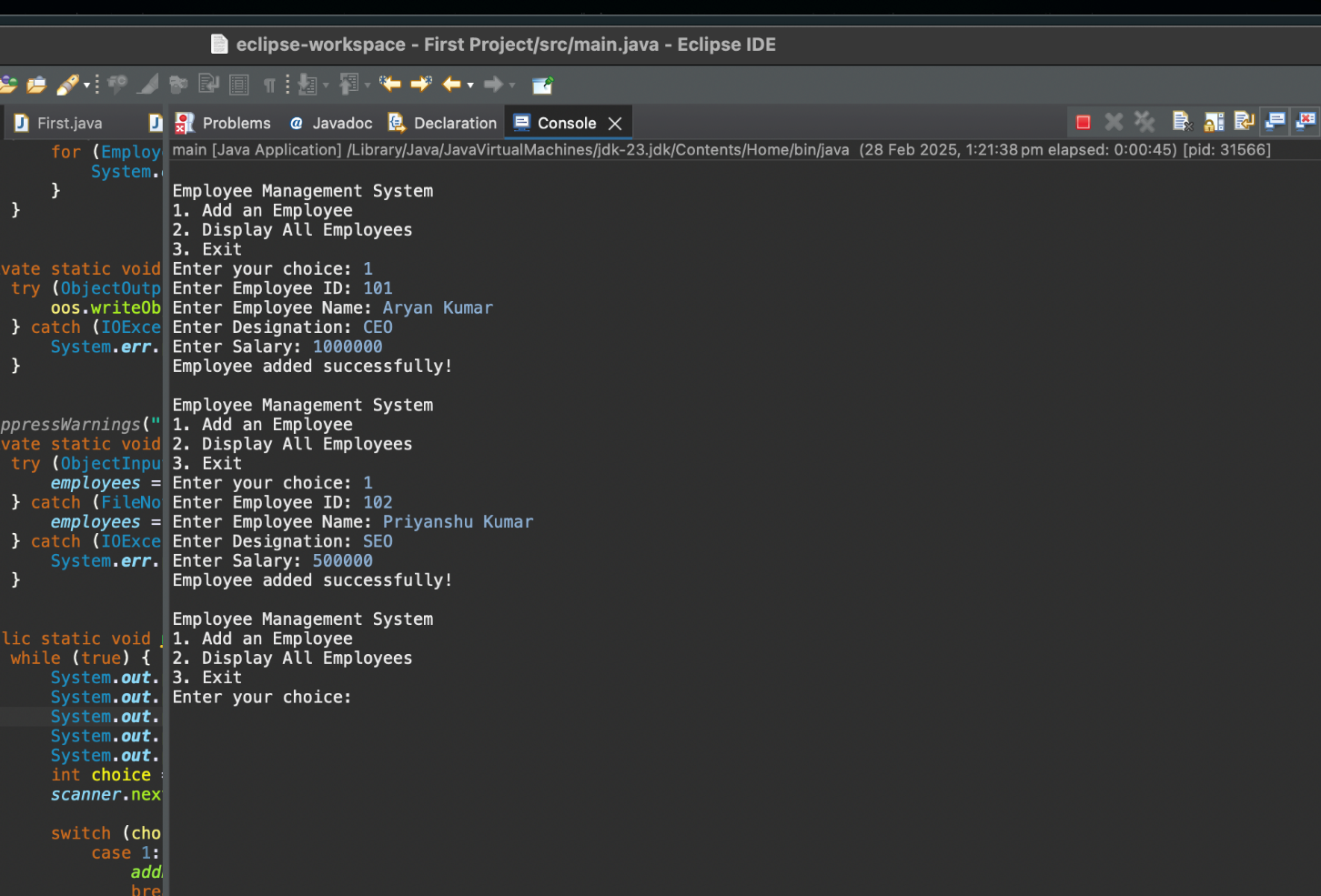
}

}

}

}

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.