Experiment 5

Name: Neha Kumari **UID:22BCS10009**
Branch: BE-CSE **Section/Group:22BCS618-A**
Semester: 6th **DateofPerformance:21/02/2025**
Subject Name: Project Based Learning in Java with Lab **Subject Code: 22CSH-359**

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

2. Objective : The objective of developing Java programs that utilize autoboxing, serialization, file handling, and efficient data processing and management is to create robust applications that can effectively manage data in a structured manner.

3. Implementation/Code:

3.1 Writing a Java program to calculate the sum of a list of integers using autoboxing and unboxing, along with methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

```
import java.util.ArrayList;
import java.util.List;
public class Project1 {
    public static Integer parseStringToInteger(String str) {
        try {
            return Integer.parseInt(str);
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format: " + str);
            return null;
        }
    }
    public static int calculateSum(List<Integer> integers) {
        int sum = 0;
        for (Integer number : integers) {
            if (number != null) {
                sum += number;
            }
        }
        return sum;
```

```
    }
    public static void main(String[] args) {
        List<Integer> numbers1 = new ArrayList<>();
        numbers1.add(10);
        numbers1.add(20);
        numbers1.add(30);
        numbers1.add(parseStringToInteger("40"));
        numbers1.add(parseStringToInteger("50"));
        System.out.println("The sum of the list is: " + calculateSum(numbers1));
        List<Integer> numbers2 = new ArrayList<>();
        numbers2.add(parseStringToInteger("100"));
        numbers2.add(parseStringToInteger("200"));
        numbers2.add(parseStringToInteger("300"));
        System.out.println("The sum of the list is: " + calculateSum(numbers2));
        List<Integer> numbers3 = new ArrayList<>();
        numbers3.add(parseStringToInteger("50"));
        numbers3.add(parseStringToInteger("invalid"));
        numbers3.add(parseStringToInteger("70"));
        System.out.println("The sum of the list is: " + calculateSum(numbers3));
    }
}
```

Output:

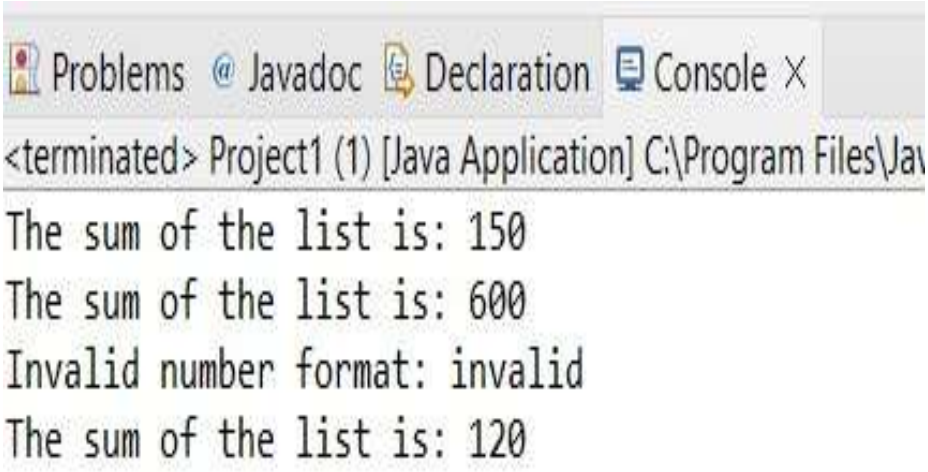

Fig. 1 (Output 3.1)

3.2 Java program that serializes and deserializes a Student object. It saves the Student object to a file and then reads it back, displaying the student details. The program handles exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

```
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 Project1 {
   public static void serializeStudent(Student student) {
      try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("student.ser")))
 {
         oos.writeObject(student);
         System.out.println("Student object has been serialized and saved to file.");
      } catch (FileNotFoundException e) {
         System.out.println("Error: File not found.");
      } catch (IOException e) {
         System.out.println("Error: IOException occurred during serialization.");
      }
   }
   public static Student deserializeStudent() {
      try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("student.ser"))) {
         Student student = (Student) ois.readObject();
         System.out.println("Student object has been deserialized.");
         return student;
      } catch (FileNotFoundException e) {
         System.out.println("Error: File not found.");
      } catch (IOException e) {
         System.out.println("Error: IOException occurred during deserialization.");
      } catch (ClassNotFoundException e) {
         System.out.println("Error: Class not found.");
      }
```

```
        return null;
    }
    public static void main(String[] args) {
        Student student1 = new Student(1, "John Doe", 3.75);
        serializeStudent(student1);
        Student deserializedStudent = deserializeStudent();
        if (deserializedStudent != null) {
            System.out.println("Deserialized Student Details:");
            System.out.println(deserializedStudent);
        }
        System.out.println("\nTest Case 2: Attempting to deserialize from a non-existent file.");
        new File("student.ser").delete();
        deserializeStudent();
        System.out.println("\nTest Case 3: Simulating ClassNotFoundException.");
        deserializeStudent();
    }
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> Project1 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe
Student object has been serialized and saved to file.
Student object has been deserialized.
Deserialized Student Details:
Student ID: 1, Name: John Doe, GPA: 3.75

Test Case 2: Attempting to deserialize from a non-existent file.
Error: File not found.

Test Case 3: Simulating ClassNotFoundException.
Error: File not found.
```

Fig. 2 (Output 3.2)

3.3 Menu-based Java application that allows you to add employee details, display all employees, and exit. The employee details will be stored in a file, and the program will read the file to display the stored employee information.

```
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 String name;
   private int id;
   private String designation;
   private double salary;
   public Employee(String name, int id, String designation, double salary) {
      this.name = name;
      this.id = id;
      this.designation = designation;
      this.salary = salary;
   }
   @Override
   public String toString() {
      return "Employee ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " +
salary;}}
public class Project1 {
   private static final String FILE_NAME = "employees.ser";
   public static void addEmployee() {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Employee Name: ");
      String name = scanner.nextLine();
      System.out.print("Employee ID: ");
      int id = scanner.nextInt();
      scanner.nextLine();
      System.out.print("Designation: ");
      String designation = scanner.nextLine();
      System.out.print("Salary: ");
      double salary = scanner.nextDouble();
      Employee employee = new Employee(name, id, designation, salary);
      saveEmployeeToFile(employee);
      System.out.println("Employee added successfully!");}
   private static void saveEmployeeToFile(Employee employee) {
      List<Employee> employees = readEmployeesFromFile();
      employees.add(employee);
      try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
```

```
            for (Employee emp : employees) {
                oos.writeObject(emp);
            }
        } catch (IOException e) {
            System.out.println("Error saving employee to file: " + e.getMessage());}}
    public static void displayAllEmployees() {
        List<Employee> employees = readEmployeesFromFile();
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            for (Employee employee : employees) {
                System.out.println(employee);}}}
    private static List<Employee> readEmployeesFromFile() {
        List<Employee> employees = new ArrayList<>();
        try {
            File file = new File(FILE_NAME);
            if (file.exists()) {
                try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {
                    while (true) {
                        Employee employee = (Employee) ois.readObject();
                        employees.add(employee);
                    }
                } catch (EOFException e) {
                } catch (IOException | ClassNotFoundException e) {
                    System.out.println("Error reading employees from file: " + e.getMessage());}}
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
        return employees;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employee");
            System.out.println("2. Display All Employees");
            System.out.println("3 . Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayAllEmployees();
```

```
                break;
            case 3:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");}
        } while (choice != 3);}}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
Project1 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (21-Feb-2025, 12:04:30 pm elapsed: 0:01:15) [pid:

Menu:
1. Add Employee
2. Display All Employees
3 . Exit
Enter your choice: 1
Employee Name: Neha
Employee ID: 101
Designation: Software Engineer
Salary: 50000
Error reading employees from file: invalid type code: AC
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3 . Exit
Enter your choice: 1
Employee Name: Tarun
Employee ID: 102
Designation: Software Testing
Salary: 60000
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3 . Exit
Enter your choice: 2
Employee ID: 101, Name: John Doe, Designation: Software Engineer, Salary: 50000.0
Employee ID: 101, Name: Neha , Designation: Software Engineer , Salary: 50000.0
Employee ID: 102, Name: Tarun, Designation: Software Testing, Salary: 60000.0

Menu:
1. Add Employee
2. Display All Employees
3 . Exit
```

Fig. 3 (Output 3.3)

4. Learning Outcomes:

- Understand automatic conversion between primitive types and their wrapper classes.
- Learn to implement the Serializable interface and use serialVersionUID.
- Gain skills in reading from and writing to files using Java I/O.
- Process data efficiently through searching, sorting, and filtering.
- Design user-friendly interfaces for data input and output.
- Enhance problem-solving abilities by implementing solutions in Java.
- Develop critical thinking through analyzing requirements and designing algorithms.