Experiment-5

Name: Ahatsham ansari
Branch: BE-CSE
Semester: 6th
Subject Name: Project Based Learning in Java

UID:22BCS10017
Section/Group: 643/A
D.Performance:3/03/2025
Subject Code: 22CSH-359

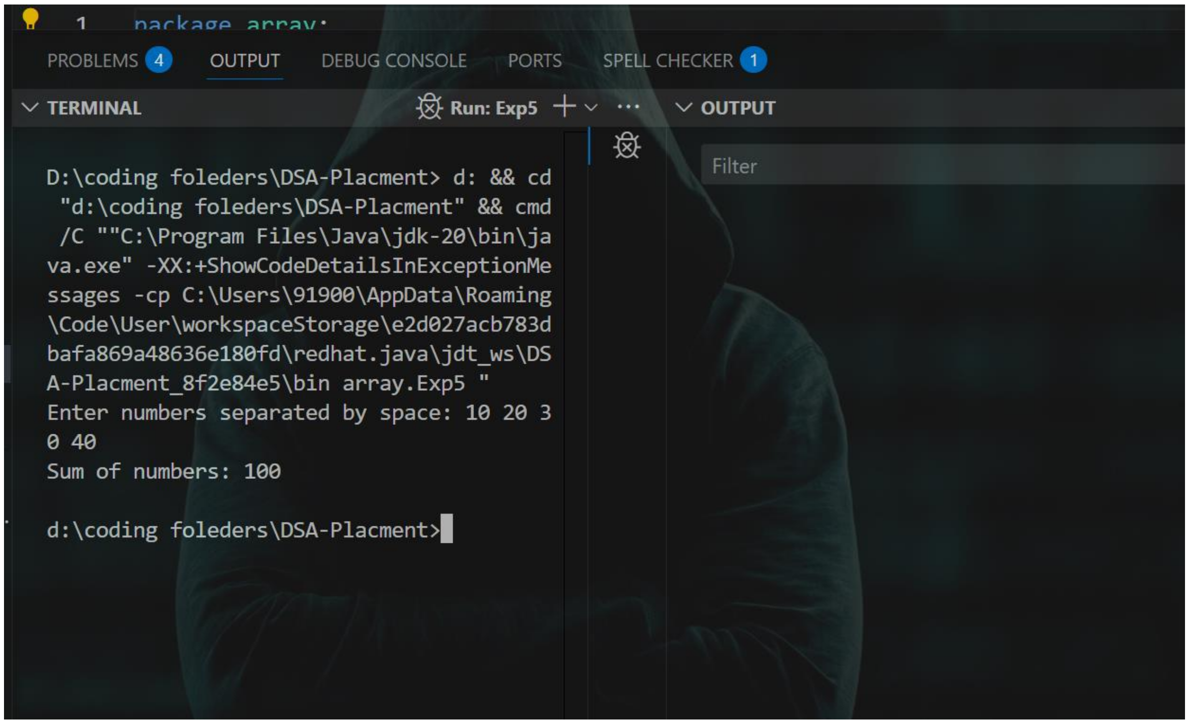
1. **Aim :** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

2. **Easy Level:**
   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()).

3. **Implementation/Code:**

```
package array;
import java.util.*;
import java.util.stream.Collectors;

public class Exp5{

    // Calculate sum using Streams (Alternative method)
    public static int calculateSum(List<Integer> numbers) {
        return numbers.stream().mapToInt(Integer::intValue).sum();
    }

    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter numbers separated by space: ");
        String inputLine = scanner.nextLine();

        // Converting input string to list of integers (Handling spaces properly)
        try {
            numbers = Arrays.stream(inputLine.trim().split("\\s+"))
                    .map(Integer::parseInt) // Autoboxing happens here
                    .collect(Collectors.toList());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter only numbers.");
            scanner.close();
            return;
        }
        System.out.println("Sum of numbers: " + calculateSum(numbers));
        scanner.close();
    }
}
```

1. OUTPUT



```
D:\coding foleders\DSA-Placment> d: && cd
 "d:\coding foleders\DSA-Placment" && cmd
 /C ""C:\Program Files\Java\jdk-20\bin\ja
va.exe" -XX:+ShowCodeDetailsInExceptionMe
ssages -cp C:\Users\91900\AppData\Roaming
\Code\User\workspaceStorage\e2d027acb783d
bafa869a48636e180fd\redhat.java\jdt_ws\DS
A-Placment_8f2e84e5\bin array.Exp5 "
Enter numbers separated by space: 10 20 3
0 40
Sum of numbers: 100

d:\coding foleders\DSA-Placment>
```

Medium Level:

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.

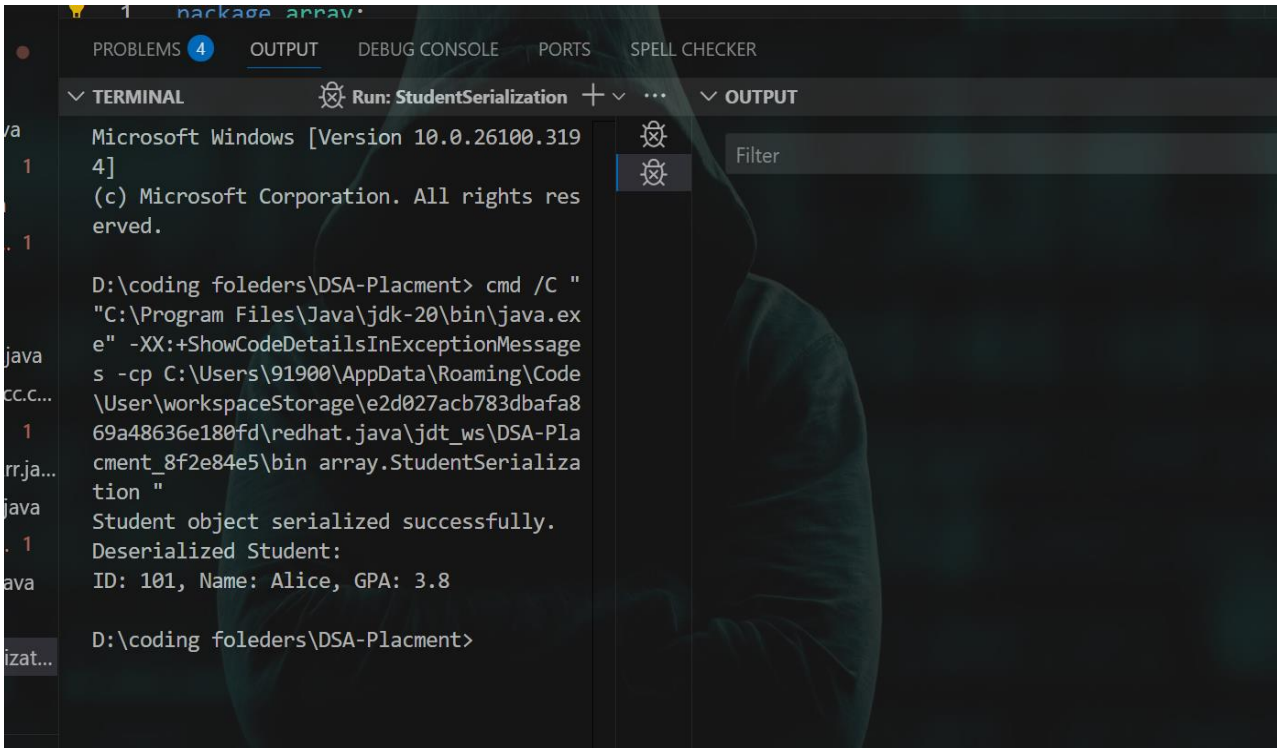
Handle FileNotFoundException, IOException, and ClassNotFoundException using exception handling.

Code:

```
package array;
    import java.io.*;
    // Serializable class
    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;
        }

        public void display() {
            System.out.println("ID: " + id + ", Name: " + name + ", GPA: " + gpa);
        }
    }

    public class StudentSerialization {
```

```
    private static final String FILE_NAME = "student.ser";

    // Serialize object
    public static void serializeStudent(Student student) {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            out.writeObject(student);
            System.out.println("Student object serialized successfully.");
        } catch (IOException e) {
            System.out.println("Error during serialization: " + e.getMessage());
        }
    }

    // Deserialize object
    public static Student deserializeStudent() {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            return (Student) in.readObject();
        } catch (FileNotFoundException e) {
            System.out.println("File not found. No data available for deserialization.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        }
        return null;
    }

    public static void main(String[] args) {
        // Create and serialize a Student object
        Student student = new Student(101, "Alice", 3.8);
        serializeStudent(student);

        // Deserialize and display the Student object
        Student deserializedStudent = deserializeStudent();
        if (deserializedStudent != null) {
            System.out.println("Deserialized Student:");
            deserializedStudent.display();
        }
    }
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.26100.319
4]
(c) Microsoft Corporation. All rights res
erved.

D:\coding foleders\DSA-Placment> cmd /C "
"C:\Program Files\Java\jdk-20\bin\java.ex
e" -XX:+ShowCodeDetailsInExceptionMessage
s -cp C:\Users\91900\AppData\Roaming\Code
\User\workspaceStorage\e2d027acb783dbafa8
69a48636e180fd\redhat.java\jdt_ws\DSA-Pla
cment_8f2e84e5\bin array.StudentSerializa
tion "
Student object serialized successfully.
Deserialized Student:
ID: 101, Name: Alice, GPA: 3.8

D:\coding foleders\DSA-Placment>
```

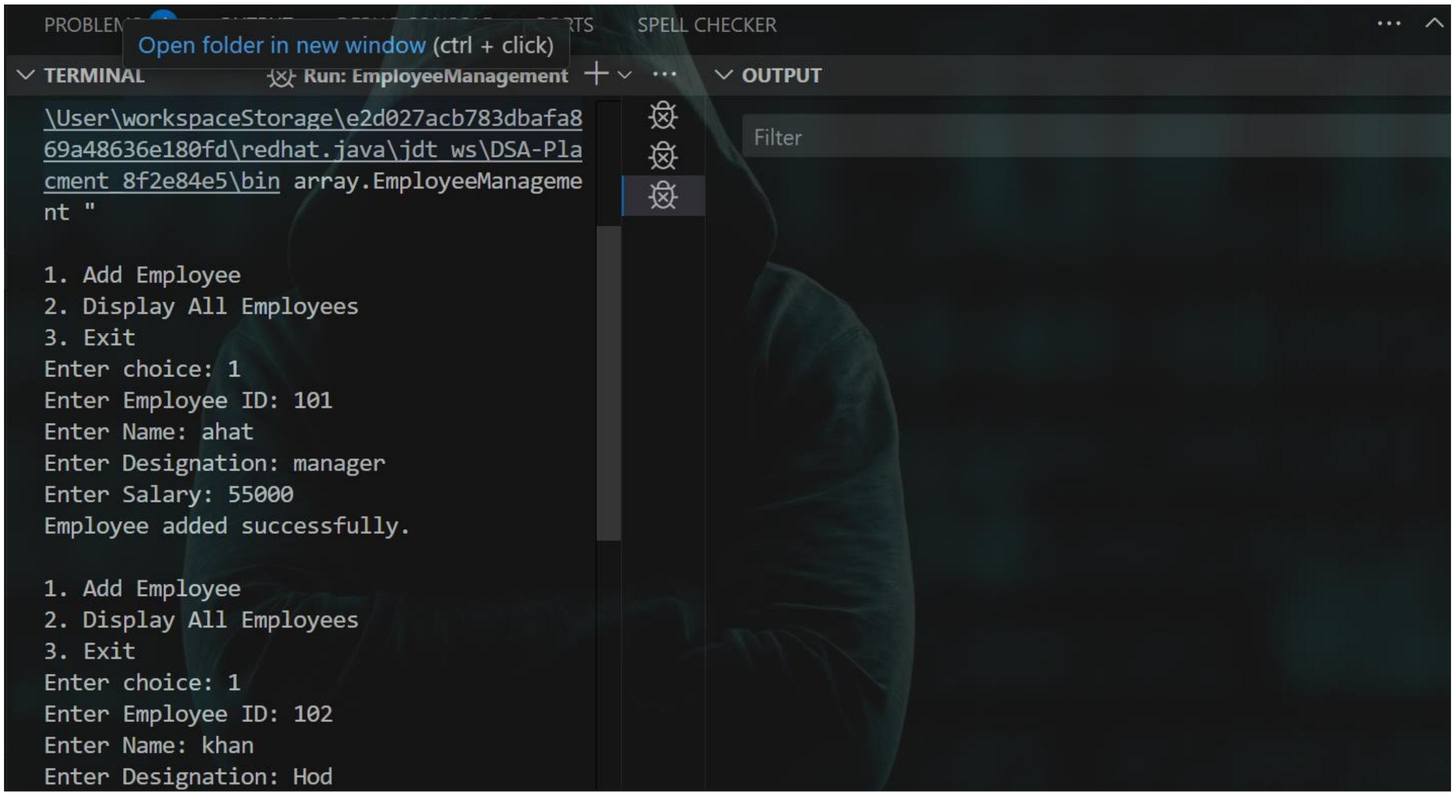
Hard Level:

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.

Code:

```
package array;
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 "ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;
    }
}

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

    // Add an employee and save to file
    public static void addEmployee(Employee emp) {
        List<Employee> employees = loadEmployees();
        employees.add(emp);
        saveEmployees(employees);
        System.out.println("Employee added successfully.");
    }

    // Display all employees
    public static void displayEmployees() {
        List<Employee> employees = loadEmployees();
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
            return;
        }
        System.out.println("\nEmployee List:");
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }

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

    // Load employee list from file
    @SuppressWarnings("unchecked")
    private static List<Employee> loadEmployees() {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            return (List<Employee>) in.readObject();
        } catch (FileNotFoundException e) {
            return new ArrayList<>(); // Return empty list if file does not exist
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error loading employees: " + e.getMessage());
            return new ArrayList<>();
        }
    }

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

        while (true) {
            System.out.println("\n1. Add Employee\n2. Display All Employees\n3. Exit");
            System.out.print("Enter choice: ");

            // Ensure valid integer input
            while (!scanner.hasNextInt()) {
                System.out.println("Invalid input! Please enter a number.");
                scanner.next();
            }
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter Employee ID: ");
                    while (!scanner.hasNextInt()) {
                        System.out.println("Invalid input! Please enter a valid ID.");
                        scanner.next();
                    }
                    int id = scanner.nextInt();
                    scanner.nextLine(); // Consume newline

                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();

                    System.out.print("Enter Designation: ");
                    String designation = scanner.nextLine();

                    System.out.print("Enter Salary: ");
                    while (!scanner.hasNextDouble()) {
                        System.out.println("Invalid input! Please enter a valid salary.");
                        scanner.next();
                    }
                    double salary = scanner.nextDouble();

                    addEmployee(new Employee(id, name, designation, salary));
                    break;

                case 2:
                    displayEmployees();
                    break;

                case 3:
                    System.out.println("Exiting program.");
                    scanner.close();
                    return;

                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
}
```

OUTPUT:



```
\User\workspaceStorage\e2d027acb783dbafa8
69a48636e180fd\redhat.java\jdt_ws\DSA-Pla
cment_8f2e84e5\bin array.EmployeeManageme
nt "

1. Add Employee
2. Display All Employees
3. Exit
Enter choice: 1
Enter Employee ID: 101
Enter Name: ahat
Enter Designation: manager
Enter Salary: 55000
Employee added successfully.

1. Add Employee
2. Display All Employees
3. Exit
Enter choice: 1
Enter Employee ID: 102
Enter Name: khan
Enter Designation: Hod
```

2. Learning outcomes:

1. **Collections in Java:** Learn ArrayList, HashMap, and Collection interfaces for efficient data storage and retrieval.

2. **CRUD Operations:** Implement basic operations like Add, Update, Remove, and Search using Java collections.

3. **Multithreading & Synchronization:** Use synchronized and ReentrantLock to handle concurrent access and prevent race conditions.
4. **Thread Priorities:** Assign priorities (MAX_PRIORITY, NORM_PRIORITY) to ensure important tasks (e.g., VIP bookings) execute first.