Experiment- 05

Student Name: Pargat Singh	UID: 22ICS10002

Branch: BE-CSE	Section/Group: IOT-641/B

Semester: 6th	Date of Performance: 05/03/2025

Subject Name: Project Based Learning in JAVA with Lab.	Code: 22CSH-359

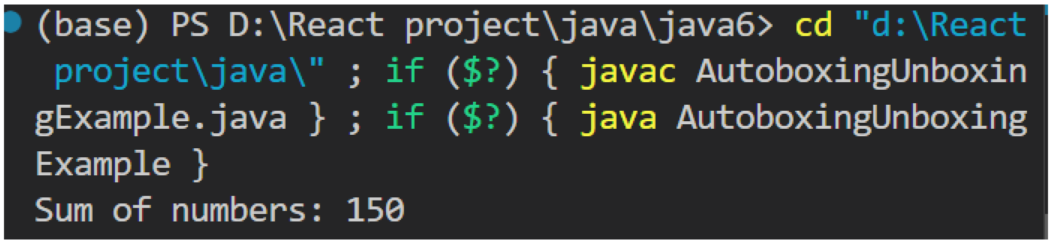
1. **Aim(EASY LEVEL)** : Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing. Include methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

2. **Objective:** To demonstrate autoboxing and unboxing in Java by calculating the sum of a list of integers. The program will also include methods to convert string inputs into integer values using wrapper classes like Integer.parseInt().

3. **Implementation/Code:**

```
import java.util.ArrayList;
import java.util.List;

public class AutoboxingUnboxingExample {

    public static void main(String[] args) {
        String[] numberStrings = {"10", "20", "30", "40", "50"};
        List<Integer> numbers = parseStringArrayToIntegerList(numberStrings);
        int sum = calculateSum(numbers);
        System.out.println("Sum of numbers: " + sum);
    }

    private static List<Integer> parseStringArrayToIntegerList(String[] strArray) {
        List<Integer> intList = new ArrayList<>();
        for (String str : strArray) {
            intList.add(Integer.parseInt(str));
        }
        return intList;
    }

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

4. **Output:**



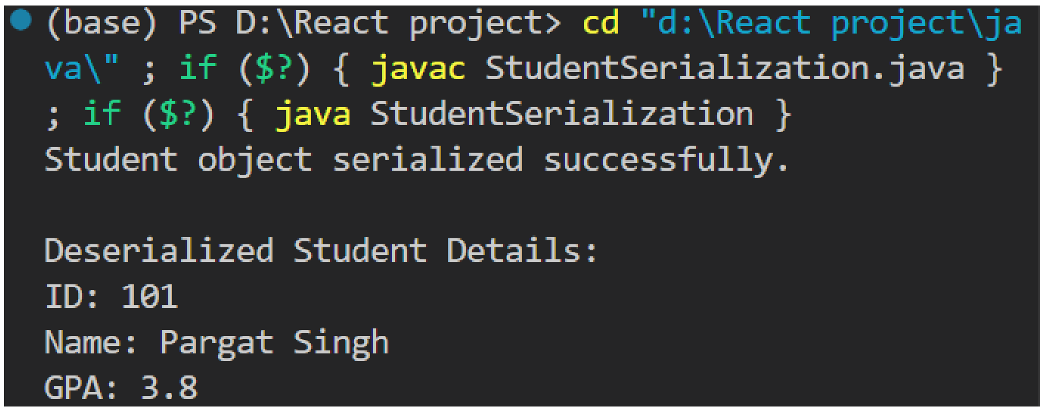
```
(base) PS D:\React project\java\java6> cd "d:\React
 project\java\" ; if ($?) { javac AutoboxingUnboxin
gExample.java } ; if ($?) { java AutoboxingUnboxing
Example }
Sum of numbers: 150
```

AIM(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.

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;
    }

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

public class StudentSerialization {
    private static final String FILE_NAME = "student.ser";
    public static void main(String[] args) {

        Student student = new Student(101, "Pargat Singh", 3.8);
        serializeStudent(student);
        Student deserializedStudent = deserializeStudent();
        if (deserializedStudent != null) {
            System.out.println("\nDeserialized Student Details:");
            deserializedStudent.displayStudent();
        }
    }

    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 (IOException e) {
            e.printStackTrace();
        }
    }

    public static Student deserializeStudent() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            return (Student) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Output:



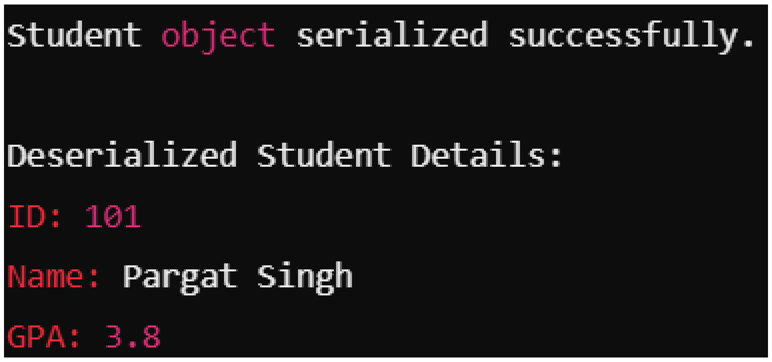
```
(base) PS D:\React project> cd "d:\React project\ja
va\" ; if ($?) { javac StudentSerialization.java }
; if ($?) { java StudentSerialization }
Student object serialized successfully.

Deserialized Student Details:
ID: 101
Name: Pargat Singh
GPA: 3.8
```

Aim(MEDIUM LEVEL): Deserialize the object from the file and display the student details. Handle FileNotFoundException, IOException, and ClassNotFoundException using exception handling.

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;
    }

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

public class StudentSerialization {
    private static final String FILE_NAME = "student.ser";

    public static void main(String[] args) {
        Student student = new Student(101, "Pargat Singh", 3.8);
        serializeStudent(student);
        Student deserializedStudent = deserializeStudent();
        if (deserializedStudent != null) {
            System.out.println("\nDeserialized Student Details:");
            deserializedStudent.displayStudent();
        }
    }

    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 (IOException e) {
            System.err.println("Error during serialization: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public static Student deserializeStudent() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            return (Student) ois.readObject();
        } catch (FileNotFoundException e) {
            System.err.println("File not found: " + FILE_NAME);
        } catch (IOException e) {
            System.err.println("IO Exception occurred while reading the file: " +
e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Class definition not found for deserialization.");
        }
        return null;
    }
}
```

Output:



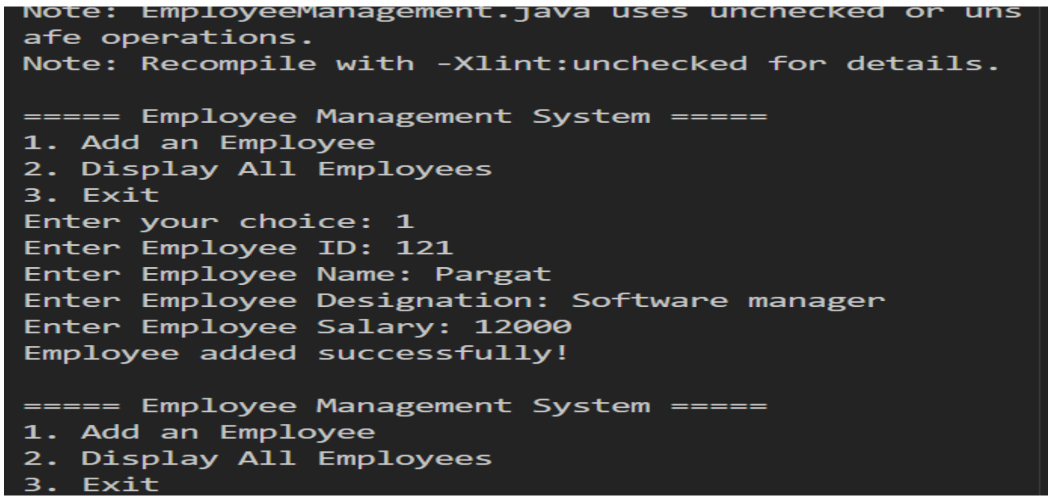
```
Student object serialized successfully.

Deserialized Student Details:
ID: 101
Name: Pargat Singh
GPA: 3.8
```

Aim(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.

Implementation/Code:

```
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;
    }

    public void displayEmployee() {
        System.out.println("\nEmployee ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Designation: " + designation);
        System.out.println("Salary: " + salary);
    }
}

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nMenu:");
            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(); // Consume newline

            switch (choice) {
                case 1:
                    addEmployee(scanner);
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.out.println("Exiting program...");
                    scanner.close();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice! Please enter 1, 2, or 3.");
            }
        }
    }

    private static void addEmployee(Scanner scanner) {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        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 (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME, true))) {
            oos.writeObject(employee);
            System.out.println("Employee added successfully!");
        } catch (IOException e) {
            System.err.println("Error saving employee data: " + e.getMessage());
        }
    }

    private static void displayEmployees() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            System.out.println("\nEmployee Details:");
            while (true) {
                try {
                    Employee emp = (Employee) ois.readObject();
                    emp.displayEmployee();
                } catch (EOFException e) {
                    break; // End of file reached
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("No employees found. Add an employee first.");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Error reading employee data: " + e.getMessage());
        }
    }
}
```

Output:



```
Note: EmployeeManagement.java uses unchecked or uns
afe operations.
Note: Recompile with -Xlint:unchecked for details.

===== Employee Management System =====
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 121
Enter Employee Name: Pargat
Enter Employee Designation: Software manager
Enter Employee Salary: 12000
Employee added successfully!

===== Employee Management System =====
1. Add an Employee
2. Display All Employees
3. Exit
```

5. **Learning Outcomes:**

1. File Handling in Java – Writing and reading objects from a file.
2. Serialization & Deserialization – Storing and retrieving objects persistently.
3. Menu-Driven Programs – Implementing interactive user input handling.
4. Exception Handling – Managing errors like `FileNotFoundException` and `IOException`.
5. Object-Oriented Programming (OOP) Concepts – Using classes, objects, and encapsulation.