# Experiment 5

Student Name: Manvendra Singh
UID: 22BCS16432
Branch: BE-CSE
Section/Group: DL_903_B
Semester: 6th
Date of Performance: 19-02-2025
Subject Name: Program Based Learning in Java with Lab
Subject Code: 22CSH-359

## 1. Aim:

a). 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()).

b.) 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.

c.) 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.

## 2. Implementation/Code:

1.) Easy: 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()).

Code:

```
import java.util.*;

public class AutoboxingUnboxingSum {

   public static void main(String[] args) {
      List<Integer> numbers = new ArrayList<>();
      String[] inputNumbers = {"10", "20", "30", "40", "50"};

      // Parse strings and add to the list (Autoboxing)
      for (String numStr : inputNumbers) {
         numbers.add(parseInteger(numStr));

   }
```

```
        // Calculate sum (Unboxing)
        int sum = calculateSum(numbers);

        System.out.println("Sum of numbers: " + sum);
    }

    // Method to parse string to Integer
    private static Integer parseInteger(String numberStr) {
        return Integer.parseInt(numberStr); // Autoboxing
    }

    // Method to calculate sum using unboxing
    private static int calculateSum(List<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num; // Unboxing
        }
        return sum;
    }
}
```

2.) 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:

```
import java.io.*;

// Student class implementing Serializable
class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private double gpa;
```

```
// Constructor
    public Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    // Method to display student details
    public void display() {
        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";

    // Method to serialize a Student object
    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("Error: File not found.");
        } catch (IOException e) {
            System.err.println("Error: IOException occurred while serializing.");
        }
    }

    // Method to deserialize a Student object
    public static Student deserializeStudent() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            return (Student) ois.readObject();
        } catch (FileNotFoundException e) {
            System.err.println("Error: File not found.");
        } catch (IOException e) {
            System.err.println("Error: IOException occurred while deserializing.");
        } catch (ClassNotFoundException e) {
            System.err.println("Error: Class not found.");
        }
        return null;
    }
```

```
    public static void main(String[] args) {
          // Create a Student object
          Student student = new Student(1, "John Doe", 3.8);

          // Serialize the Student object
       serializeStudent(student);
 // Deserialize the Student object
       Student deserializedStudent = deserializeStudent();

       // Display the deserialized student details if successful
       if (deserializedStudent != null) {
          System.out.println("\nDeserialized Student Details:");
          deserializedStudent.display();
       }
    }
     }
```

3).Hard: 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:

```
import java.io.*;
import java.util.*;

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 + "\nName: " + name + "\nDesignation: " + designation + "\nSalary: " +
salary + "\n";
    }
}
```

```
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");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    addEmployee(scanner);
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.out.println("Exiting application.");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }

    private static void addEmployee(Scanner scanner) {
        System.out.print("Enter Employee Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();  // Consume newline
        System.out.print("Enter Designation: ");
        String designation = scanner.nextLine();
        System.out.print("Enter Salary: ");
        double salary = scanner.nextDouble();

        Employee emp = new Employee(name, id, designation, salary);
        List<Employee> employees = readEmployees();
        employees.add(emp);
        writeEmployees(employees);
        System.out.println("Employee added successfully!");
```

```
    }

    private static void displayEmployees() {
        List<Employee> employees = readEmployees();
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            for (Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }

    private static List<Employee> readEmployees() {
        List<Employee> employees = new ArrayList<>();
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            employees = (List<Employee>) ois.readObject();
        } catch (FileNotFoundException e) {
            // File not found, return empty list
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error reading employees: " + e.getMessage());
        }
        return employees;
    }

    private static void writeEmployees(List<Employee> employees) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            oos.writeObject(employees);
        } catch (IOException e) {
            System.out.println("Error writing employees: " + e.getMessage());
        }
    }
}
```

3.Output

1.) Easy problem output

```
Sum of numbers: 150


...Program finished with exit code 0
Press ENTER to exit console.
```

2.) Medium problem output

```
Student object serialized successfully.

Deserialized Student Details:
ID: 1
Name: John Doe
GPA: 3.8


...Program finished with exit code 0
Press ENTER to exit console.
```

3.) Hard problem output

```
Menu:
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 1
Enter Employee Name: Manvendra
Enter Employee ID: 101
Enter Designation: Manager
Enter Salary: 103000
Employee added successfully!

Menu:
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 2
Employee ID: 101
Name: Manvendra
Designation: Manager
Salary: 103000.0


Menu:
```

## 4 Learning Outcomes

1. Learned to use classes and objects for organizing employee and designation data in Java.
2. Implemented salary calculations using switch-case and array data handling.
3. Practiced input handling with the Scanner class and validating user input.
4. Gained experience in searching arrays and structuring conditional logic.

Displayed formatted output for real-world applications like employee management systems