

Experiment - 5

Student Name: Shivam

Branch: B.ECSE

Semester: 6th

Subject: PBLJ

UID: 22BCS50010

Section: IOT-643-A

DOP: 24/02/25

Subject Code: 22CSH-359

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

2) **Problem Statement:**

- 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.

3) **Algorithm:**

a. **Sum of a List of Integers Using Autoboxing and Unboxing:**

- Initialize a list of integers.
- Convert a list of string numbers into integers using `Integer.parseInt()`.
- Use autoboxing to store the integers in an `ArrayList<Integer>`.
- Iterate through the list and use unboxing to sum the integers.
- Display the sum.

b. **Serialization & Deserialization of Student Object:**

- Create a Student class with attributes: ID, Name, and GPA.
- Implement `Serializable` interface in the Student class.
- Serialize the object using `ObjectOutputStream` and save it in a file.
- Deserialize the object using `ObjectInputStream` and display its attributes.
- Handle exceptions: `FileNotFoundException`, `IOException`, and `ClassNotFoundException`.

c. **Menu-based Employee Management System:**

- Create an `Employee` class with attributes: ID, Name, Designation, and Salary.
- Implement a menu-driven system:

Option 1: Take employee details as input and save to a file.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Option 2: Read from the file and display all employee records.

Option 3: Exit the application.

- Use serialization to store and retrieve employee data.

4) Program:

a. Sum of a List of Integers Using Autoboxing and Unboxing:

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

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

```
public class AutoBoxingSum {  
    public static void main(String[] args) {  
        // Given list of strings representing numbers  
        String[] strNumbers = {"10", "20", "30", "40", "50"};  
  
        // Autoboxing: Converting string array into a list of Integers  
        List<Integer> numbers = new ArrayList<>();  
        for (String str : strNumbers) {  
            numbers.add(Integer.parseInt(str)); // Autoboxing  
        }  
  
        // Calculating the sum with unboxing  
        int sum = 0;  
        for (Integer num : numbers) {  
            sum += num; // Unboxing  
        }  
  
        // Displaying the sum  
        System.out.println("Sum of numbers: " + sum);  
    }  
}
```

b. Serialization & Deserialization of Student Object:

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

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



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
this.id = id;
this.name = name;
this.gpa = gpa;
}

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

public class StudentSerialization {
    public static void main(String[] args) {
        String filename = "student_data.ser";

        // Creating a student object
        Student student = new Student(101, "Alice", 3.9);

        // Serialization
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(filename))) {
            out.writeObject(student);
            System.out.println("Student object serialized successfully.");
        } catch (IOException e) {
            System.err.println("Error during serialization: " + e);
        }

        // Deserialization
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
            Student deserializedStudent = (Student) in.readObject();
            System.out.println("Deserialized Student details:");
            deserializedStudent.display();
        } catch (FileNotFoundException e) {
            System.err.println("File not found: " + e);
        } catch (IOException e) {
            System.err.println("I/O error: " + e);
        } catch (ClassNotFoundException e) {
            System.err.println("Class not found: " + e);
        }
    }
}
```

c. Menu-based Employee Management System:

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



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
import java.util.List;
import java.util.Scanner;

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name, designation;
    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 display() {
        System.out.println("ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: "
+ salary);
    }
}

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> employees = loadEmployees();

        while (true) {
            System.out.println("\nMenu:\n1. Add Employee\n2. Display All\n3. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    // Adding an employee
                    System.out.print("Enter Employee ID: ");
                    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: ");
                    double salary = scanner.nextDouble();

                    employees.add(new Employee(id, name, designation, salary));
                    saveEmployees(employees);
                    System.out.println("Employee added successfully!");
                
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
break;
```

```
case 2:
```

```
// Displaying all employees
System.out.println("\nEmployee Details:");
if (employees.isEmpty()) {
    System.out.println("No employee records found.");
} else {
    for (Employee emp : employees) {
        emp.display();
    }
}
break;
```

```
case 3:
```

```
// Exit the program
System.out.println("Exiting the application...");
scanner.close();
System.exit(0);
break;
```

```
default:
```

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

```
}
}
}
```

```
// Load employees from the file
```

```
public static List<Employee> loadEmployees() {
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
        return (List<Employee>) in.readObject();
    } catch (FileNotFoundException e) {
        return new ArrayList<>();
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
        return new ArrayList<>();
    }
}
```

```
// Save employees to the file
```

```
public static void saveEmployees(List<Employee> employees) {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
        out.writeObject(employees);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

5) OUTPUT:

a. Sum of a List of Integers Using Autoboxing and Unboxing:

```
Sum of numbers: 150
```

b. Serialization & Deserialization of Student Object:

```
Student object serialized successfully.  
Deserialized Student details:  
ID: 101, Name: Alice, GPA: 3.9
```

c. Menu-based Employee Management System:

Menu:

1. Add Employee
2. Display All
3. Exit

Choose an option: 1

Enter Employee ID: 101

Enter Name: John Doe

Enter Designation: Developer

Enter Salary: 60000

Employee added successfully!

Menu:

1. Add Employee
2. Display All
3. Exit

Choose an option: 2

Employee Details:

ID: 101, Name: John Doe, Designation: Developer, Salary: 60000.0

Menu:

1. Add Employee
2. Display All
3. Exit





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

6) Learning Outcomes:

- ☐ Understand and implement **autoboxing and unboxing** in Java.
- ☐ Learn **serialization and deserialization** to store and retrieve objects.
- ☐ Gain proficiency in **file handling** for data storage and retrieval.
- ☐ Develop **exception handling** skills for robust applications.
- ☐ Design and implement a **menu-driven application** for user interaction.
- ☐ Work with **object-oriented programming (OOP) concepts** in Java.
- ☐ Build real-world **data management applications** using Java.



DEPARTMENT OF **COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.