



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment5

Student Name: Yasir Malik

UID: 22BCS13301

Branch: BE-CSE

Section/Group: IOT-640/B

Semester: 6th

Date of Performance: 24/02/2025

Subject Name: PBLJ Lab

SubjectCode:22CSH-359

1. Aim:

Implement and manage data using Java Collections and Exception Handling.

2. Problem Statements:

- **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()).
- **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.
- **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.



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

3. Implementation/Code:

Problem 1.1: Sum of Integers Using Autoboxing and Unboxing

```
import java.util.*;

public class SumCalculator {
    public static int calculateSum(List<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num; // Autounboxing happens here
        }
        return sum;
    }

    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
        System.out.println("Sum: " + calculateSum(numbers));
    }
}
```

Problem 1.2: Serialization and Deserialization of a Student Object

```
import java.io.*;

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

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

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", GPA: " + gpa;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}

public class StudentSerialization {
    public static void main(String[] args) {
        Student student = new Student(101, "Alice", 3.9);
        String filename = "student.ser";

        // 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("IOException: " + e.getMessage());
        }

        // Deserialization
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
            Student deserializedStudent = (Student) in.readObject();
            System.out.println("Deserialized Student: " + deserializedStudent);
        } catch (FileNotFoundException e) {
            System.err.println("FileNotFoundException: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("IOException: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("ClassNotFoundException: " + e.getMessage());
        }
    }
}
```

Problem 1.3: Employee Management System (Menu-Based Application)

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

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name, designation;
    double salary;
```



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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.ser";
```

```
    public static void addEmployee(Employee emp) {  
        List<Employee> employees = loadEmployees();  
        employees.add(emp);  
        saveEmployees(employees);  
    }
```

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

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



DEPARTMENT OF

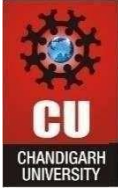
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("\n1. Add Employee\n2. Display All\n3. Exit\nEnter choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                System.out.print("Enter ID: ");
                int id = scanner.nextInt();
                scanner.nextLine();
                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();
                addEmployee(new Employee(id, name, designation, salary));
                break;
            case 2:
                displayEmployees();
                break;
            case 3:
                scanner.close();
                System.exit(0);
        }
    }
}
```



4. Output:

```
PS D:\today java> cd "d:\today java\" ; if ($?) {  
sum : 150  
PS D:\today java> 
```

(Fig. 1 - Sum of Integers Output)

```
PS D:\today java> cd "d:\today java\" ; if ($?) { javac StudentSerialization  
Student object serialized to student.ser  
Deserialized Student: Student{id=15377, name='Jobajeet Singh', gpa=7.5}  
PS D:\today java> 
```

(Fig. 2 - Student Serialization & Deserialization Output)

```
PS D:\today java> cd "d:\today java\" ; if ($?) { javac Main.java } ; if ($?) { java Main  
Menu:  
1. Add an Employee  
2. Display All  
3. Exit  
Enter your choice: 1  
Enter employee name: jobanjeet singh  
Enter employee id: 15377  
Enter employee designation: Amritsar  
Enter employee salary: 200000  
Menu:  
1. Add an Employee  
2. Display All  
3. Exit  
Enter your choice: 2  
Employee{name='jobanjeet Singh', id=15377, designation='Amritsar', salary=200000.0}  
Error reading from file: invalid type code: AC  
Menu:
```



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

(Fig. 3 - Employee Management System Output)

5. Learning Outcome:

1. Understand autoboxing and unboxing in Java.
2. Learn how to parse and work with wrapper classes.
3. Gain hands-on experience with serialization and deserialization of objects.
4. Implement exception handling for file operations in Java.
5. Develop a menu-based Java application with file handling.

1.