



Experiment 5

Name: Sanchay

Branch: BE-IT

Semester: 6th

Subject Name: Project based Learning in Java

UID: 22BET10294

Section/Group: 22BET_IOT_703/A

Date of Performance: 18-02-25

Subject Code: 22ITH-352

Problem 1:

Aim: Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing.

Objective:

- understanding of **wrapper classes, autoboxing/unboxing, parsing techniques, and exception handling** in Java.
- Understand how wrapper classes help in converting primitive data types into objects.
- Using **Integer.parseInt()** to convert string inputs into integer values.

Code: package autoboxingunboxing;

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

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

```
public class AutoBoxingExample {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        ArrayList<Integer> numbers = new ArrayList<>();
```

```
        System.out.println("Enter numbers (type 'done' to finish): ");
```

```
        while (scanner.hasNext()) {
```

```
            String input = scanner.next();
```

```
            if (input.equalsIgnoreCase("done")) break;
```

```
            try {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        numbers.add(Integer.parseInt(input)); // Autoboxing

    } catch (NumberFormatException e) {

        System.out.println("Invalid input! Please enter a valid integer.");

    }

}

int sum = 0;

for (Integer num : numbers) {

    sum += num; // Unboxing

}

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

scanner.close();

}
```

OUTPUT:

```
Enter numbers (type 'done' to finish):
30
40
36
45
done
Sum of numbers: 151
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Learning Outcomes:

- Gain knowledge of **autoboxing and unboxing** and their role in Java.
- Learn to **parse string inputs into wrapper classes** using Integer.parseInt().
- Use **ArrayList** to dynamically store numbers instead of fixed-size arrays.

Problem 2 :

Aim: Create a Java program to serialize and deserialize a Student object.

Objective:

- Understand object serialization in Java.
- Learn to write an object to a file using ObjectOutputStream.
- Handle FileNotFoundException, IOException, and ClassNotFoundException.

CODE:

```
package studentSerialization;

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

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

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

    public void display() {
        System.out.println("Student ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("CGPA: " + cgpa);
        System.out.println("-----");
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}

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

        // Creating two student objects
        List<Student> students = new ArrayList<>();
        students.add(new Student(10258, "Anuj Yadav", 7.8));
        students.add(new Student(15806, "Diksha", 8.5));

        // Serialization
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {
            out.writeObject(students);
            System.out.println("Serialization successful! Students saved to file.");
        } catch (IOException e) {
            System.out.println("IOException during serialization: " + e.getMessage());
        }

        // Deserialization
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
            List<Student> deserializedStudents = (List<Student>) in.readObject();
            System.out.println("\nDeserialized Student Details:");
            for (Student student : deserializedStudents) {
                student.display();
            }
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Serialization successful! Students saved to file.
```

```
Deserialized Student Details:
```

```
Student ID: 10258
```

```
Name: Anuj Yadav
```

```
CGPA: 7.8
```

```
-----
Student ID: 15806
```

```
Name: Diksha
```

```
CGPA: 8.5
-----
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Learning Outcomes:

1. Learn the concept of serialization and deserialization.
2. Understand the importance of serialVersionUID in serialized classes.
3. Implement file handling for saving and loading objects.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Problem 3:

Aim: Create a menu-based Java application with the following options. 1.Add an Employee 2. Display All 3. Exit If option 1 is selected.

Objective:

- To use file handling using `FileOutputStream` and `FileInputStream`.
- Store and retrieve **multiple objects** using **serialization** (`List<Employee>`).
- Ensure **data consistency** by writing and reading a list of employees.

Code:

```
package employeeManagement123;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

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

public class EmployeeManagement123 {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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 Employee");
        System.out.println("2. Display All Employees");
        System.out.println("3. Exit");
        System.out.print("Choose an option: ");

        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);
            default:
                System.out.println("Invalid option! Please try again.");
        }
    }
}

private static void addEmployee(Scanner scanner) {
    List<Employee> employees = readEmployees(); // Read existing employees

    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();
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
System.out.print("Enter Salary: ");  
double salary = scanner.nextDouble();
```

```
employees.add(new Employee(id, name, designation, salary)); // Add new employee
```

```
// Write the updated list back to the file
```

```
try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {  
    out.writeObject(employees); // Store the entire List<Employee>  
    System.out.println("Employee added successfully!");  
} catch (IOException e) {  
    System.out.println("Error while adding employee: " + e.getMessage());  
}  
}
```

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

```
private static List<Employee> readEmployees() {  
    File file = new File(FILE_NAME);  
    if (!file.exists()) {  
        return new ArrayList<>(); // Return an empty list if file does not exist  
    }  
}
```

```
try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE_NAME))) {  
    Object obj = in.readObject();
```

```
    // Ensure we read a List<Employee> and not a single Employee object  
    if (obj instanceof List) {  
        return (List<Employee>) obj;  
    } else {  
        System.out.println("File contains incorrect data. Resetting employee list.");  
        return new ArrayList<>();  
    }  
}
```

```
} catch (FileNotFoundException e) {
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        return new ArrayList<>();  
    } catch (IOException | ClassNotFoundException e) {  
        System.out.println("Error while reading employee data: " + e.getMessage());  
        return new ArrayList<>();  
    }  
}  
}
```

Output:

```
2. Display All Employees  
3. Exit  
Choose an option: 1  
File contains incorrect data. Resetting employee list.  
Enter Employee ID: 10258  
Enter Name: Anuj  
Enter Designation: SDE  
Enter Salary: 45000  
Employee added successfully!  
  
Menu:  
1. Add Employee  
2. Display All Employees  
3. Exit  
Choose an option: 1  
Enter Employee ID: 15806  
Enter Name: Diksha  
Enter Designation: Senior SDE  
Enter Salary: 86000  
Employee added successfully!  
  
Menu:  
1. Add Employee  
2. Display All Employees  
3. Exit  
Choose an option: 2  
|  
Employee Details:  
  
Employee ID: 10258  
Name: Anuj  
Designation: SDE  
Salary: 45000.0  
  
Employee ID: 15806  
Name: Diksha  
Designation: Senior SDE  
Salary: 86000.0
```

Learning Outcomes:

- Learn to store and retrieve multiple objects using List<Employee>.
- Understand the difference between writing a single object vs. a list of objects.
- Handle data persistence using object streams.