



Experiment 5

Student Name: Ishika Thakur

Branch: BE/CSE

Semester: 6th

Subject Name: Project Based
Learning in JAVA with Lab

UID: 22BCS10765

Section/Group: 22BCS_IOT-618/B

Date of Performance: 28/02/25

Subject Code: 22CSH-359

- 1. Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.
- 2. Objective:** The objective of Developing Java Programs Using Autoboxing, Serialization, File Handling, and Efficient Data Processing and Management is to equip developers with the skills to write robust, scalable, and efficient Java applications

3. Implementation/Code:

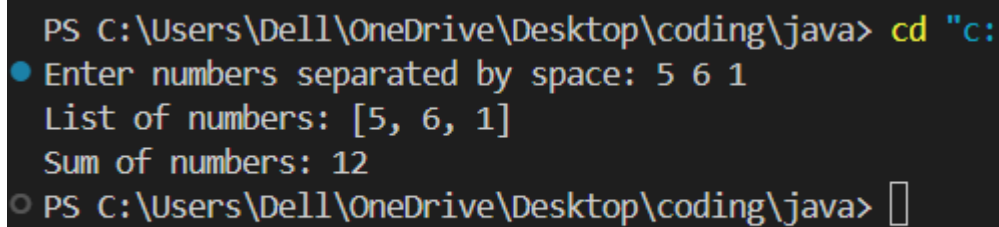
5.1: Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing. Include methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

Code:

```
import java.util.*;
public class SumCalculator {
    public static List<Integer> parseStringsToIntegers(String[] numbers) {
        List<Integer> integerList = new ArrayList<>();
        for (String num : numbers) {
            integerList.add(Integer.parseInt(num)); // Autoboxing: int → Integer
        }
        return integerList;
    }
    public static int calculateSum(List<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num; // Unboxing: Integer → int
        }
        return sum;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter numbers separated by space: ");
```

```
String input = scanner.nextLine();
String[] numStrings = input.split("\\s+");
List<Integer> numbers = parseStringsToIntegers(numStrings);
int sum = calculateSum(numbers);
System.out.println("List of numbers: " + numbers);
System.out.println("Sum of numbers: " + sum);
scanner.close();
}}
```

Output:



```
PS C:\Users\Dell\OneDrive\Desktop\coding\java> cd "c:
● Enter numbers separated by space: 5 6 1
  List of numbers: [5, 6, 1]
  Sum of numbers: 12
○ PS C:\Users\Dell\OneDrive\Desktop\coding\java> █
```

5.2: - This program aims to demonstrate object serialization and deserialization in Java by saving a student object to a file and then reading it back. It ensures proper handling of exceptions like `FileNotFoundException`, `IOException`, and `ClassNotFoundException`.

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

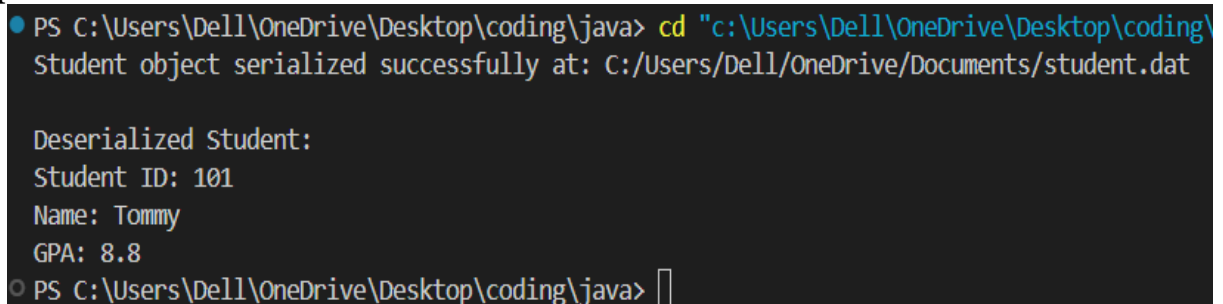
public class StudentSerialization {
```

```
public static void main(String[] args) {
    String filePath = "C:/Users/Dell/OneDrive/Documents/student.dat";

    Student student = new Student(101, "Tommy ", 8.8);
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filePath))) {
        oos.writeObject(student);
        System.out.println("Student object serialized successfully at: " + filePath);
    } catch (FileNotFoundException e) {
        System.err.println("Error: File not found - " + e.getMessage());
    } catch (IOException e) {
        System.err.println("Error during serialization - " + e.getMessage());
    }

    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filePath))) {
        Student deserializedStudent = (Student) ois.readObject();
        System.out.println("\nDeserialized Student:");
        deserializedStudent.display();
    } catch (FileNotFoundException e) {
        System.err.println("Error: File not found - " + e.getMessage());
    } catch (IOException e) {
        System.err.println("Error during deserialization - " + e.getMessage());
    } catch (ClassNotFoundException e) {
        System.err.println("Error: Class not found - " + e.getMessage());
    }
}
}
```

Output:-



```
PS C:\Users\Dell\OneDrive\Desktop\coding\java> cd "c:\Users\Dell\OneDrive\Desktop\coding\
Student object serialized successfully at: C:/Users/Dell/OneDrive/Documents/student.dat

Deserialized Student:
Student ID: 101
Name: Tommy
GPA: 8.8
PS C:\Users\Dell\OneDrive\Desktop\coding\java> █
```

5.3 Aim:- This program aims to create a menu-driven Java application that allows users to add employee details, store them in a file, and display all stored employee records. It ensures proper file handling and exception management for a smooth user experience.

Code:-

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

```
class Employee {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
String name;
int id;
String designation;
double salary;

public Employee(String name, int id, String designation, double salary) {
    this.name = name;
    this.id = id;
    this.designation = designation;
    this.salary = salary;
}

public String toCSV() {
    return id + "," + name + "," + designation + "," + salary;
}

public static Employee fromCSV(String line) {
    String[] parts = line.split(",");
    return new Employee(parts[1], Integer.parseInt(parts[0]), parts[2], Double.parseDouble(parts[3]));
}

public void display() {
    System.out.println("ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: $"
+ salary);
}

public class EmployeeManagement {
    private static final String FILE_NAME = "C:\\Users\\Dell\\OneDrive\\Documents\\employees.csv";
    public static void saveEmployees(List<Employee> employees) {
        try (PrintWriter writer = new PrintWriter(new FileWriter(FILE_NAME))) {
            for (Employee emp : employees) {
                writer.println(emp.toCSV());
            }
            System.out.println("Employee data saved at: " + new File(FILE_NAME).getAbsolutePath());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static List<Employee> loadEmployees() {
        List<Employee> employees = new ArrayList<>();
        File file = new File(FILE_NAME);
        if (!file.exists()) {
            return employees;
        }
        try (Scanner scanner = new Scanner(file)) {
            while (scanner.hasNextLine()) {
                employees.add(Employee.fromCSV(scanner.nextLine()));
            }
        }
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return employees;
}

public static Employee searchEmployee(List<Employee> employees, int searchId) {
    for (Employee emp : employees) {
        if (emp.id == searchId) {
            return emp;
        }
    }
    return null;
}

public static void main(String[] args) {
    List<Employee> employees = loadEmployees();
    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. Search Employee by ID");
        System.out.println("4. Exit");
        System.out.print("Choose an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                System.out.print("Enter Employee Name: ");
                String name = scanner.nextLine();
                System.out.print("Enter Employee ID: ");
                int id = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Enter Designation: ");
                String designation = scanner.nextLine();
                System.out.print("Enter Salary: ");
                double salary = scanner.nextDouble();

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

            case 2:
                if (employees.isEmpty()) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.println("No employees found.");
    } else {
        System.out.println("\nEmployee List:");
        for (Employee emp : employees) {
            emp.display();
        }
    }
}
break;

case 3:
    System.out.print("Enter Employee ID to search: ");
    int searchId = scanner.nextInt();
    Employee found = searchEmployee(employees, searchId);
    if (found != null) {
        System.out.println("Employee found:");
        found.display();
    } else {
        System.out.println("Employee with ID " + searchId + " not found.");
    }
    break;

case 4:
    System.out.println("Exiting application.");
    scanner.close();
    System.exit(0);
    break;

default:
    System.out.println("Invalid choice! Please select a valid option.");
}
}
}
```

Output:-

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee by ID
4. Exit
Choose an option: 1
Enter Employee Name: ishu
Enter Employee ID: 1
Enter Designation: clerk
Enter Salary: 30000
Employee data saved at: C:\Users\Dell\OneDrive\Documents\employees.csv
Employee added successfully.

Menu:
1. Add Employee
2. Display All Employees
3. Search Employee by ID
4. Exit
Choose an option: 1
Enter Employee Name: abhi
Enter Employee ID: 2
Enter Designation: developer
Enter Salary: 400000
Employee data saved at: C:\Users\Dell\OneDrive\Documents\employees.csv
Employee added successfully.

Menu:
1. Add Employee
2. Display All Employees
3. Search Employee by ID
4. Exit
Choose an option: 2

Employee List:
ID: 1, Name: ishu, Designation: clerk, Salary: $30000.0
ID: 2, Name: abhi, Designation: developer, Salary: $400000.0
```

4. Learning Outcomes:

- **Understanding Autoboxing & Unboxing:** Learners understand how Java automatically converts between primitive types (int) and wrapper classes (Integer).
- **String to Integer Parsing:** Demonstrates the use of Integer.parseInt() to convert string inputs into integer values.
- **List Handling in Java:** Shows how to store and manipulate integers using ArrayList<Integer>.
- **Looping & Summation Logic:** Reinforces iteration concepts using a for loop to sum up a list of numbers.
- **User Input Handling:** Teaches how to take space-separated user input and process it effectively using Scanner and split().