



## **Experiment 5**

**Student Name: Dipesh Goyal**

**UID: 22BCS12042**

**Branch: CSE**

**Section/Group: 640/A**

**Semester: 6<sup>th</sup>**

**Date of Performance: 12/2/25**

**Subject Name: PROJECT BASED  
LEARNING IN JAVA WITH LAB**

**Subject Code: 22CSH-352**

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

- (a) **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()).
- (b) **Medium Level:** Create a Java program to serialize and deserialize a Student object. The program should: Serialize a Student object (containing id, name, and GP 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) **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 employee details 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. Objective:** To design and implement the Sum of integers using autoboxing and unboxing, Serialization and deserialization of a Student object and Employee Management System with file handling and menu options

### **3. Implementation/Code:**

```
import java.io.*;
import java.util.*;
class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
```



**PARTMENT OF**

## **MPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

```
private int empId;
private String name;
private String designation;
private double salary;
public Employee(int empId, String name, String designation, double salary) {
    this.empId = empId;
    this.name = name;
    this.designation = designation;
    this.salary = salary;
}
public void display() {
    System.out.println("ID: " + empId + ", Name: " + name + ", Designation: " +
designation + ", Salary: " + salary);
}
}
class EmployeeManager {
    private static final String FILE_NAME = "employees.dat";
    public static void addEmployee() {
        Scanner sc = new Scanner(System.in);
        int id = 0;
        String name, designation;
        double salary = 0.0;
        try {
            System.out.print("Enter Employee ID (Integer): ");
            id = sc.nextInt(); // Fixes the InputMismatchException issue
            sc.nextLine(); // Consume newline
            System.out.print("Enter Employee Name: ");
            name = sc.nextLine();
            System.out.print("Enter Designation: ");
            designation = sc.nextLine();
            System.out.print("Enter Salary: ");
            salary = sc.nextDouble();
            Employee emp = new Employee(id, name, designation, salary);
            // Append new employee data properly
            writeEmployee(emp);
            System.out.println("Employee added successfully!");
        } catch (InputMismatchException e) {
            System.out.println("Invalid input! Please enter the correct data type.");
            sc.nextLine(); // Clear the scanner buffer
        }
    }
    private static void writeEmployee(Employee emp) {
```



**DEPARTMENT OF**

# **COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

```
List<Employee> employees = readEmployees();  
employees.add(emp);
```

```
try (ObjectOutputStream oos = new ObjectOutputStream(new  
FileOutputStream(FILE_NAME))) {  
    for (Employee e : employees) {  
        oos.writeObject(e);  
    }  
} catch (IOException e) {  
    System.out.println("Error writing to file: " + e.getMessage());  
}
```

```
private static List<Employee> readEmployees() {  
    List<Employee> employees = new ArrayList<>();
```

```
    try (ObjectInputStream ois = new ObjectInputStream(new  
FileInputStream(FILE_NAME))) {  
        while (true) {  
            try {  
                employees.add((Employee) ois.readObject());  
            } catch (EOFException e) {  
                break;  
            }  
        }  
    } catch (FileNotFoundException e) {  
        System.out.println("No existing employee records found.");  
    } catch (IOException | ClassNotFoundException e) {  
        System.out.println("Error reading file: " + e.getMessage());  
    }  
    return employees;  
}
```

```
public static void displayEmployees() {  
    List<Employee> employees = readEmployees();
```

```
    if (employees.isEmpty()) {  
        System.out.println("No employees found.");  
        return;  
    }  
    System.out.println("Employee List:");  
    for (Employee emp : employees) {  
        emp.display();  
    }  
}
```



**PARTMENT OF**

# **MPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

```
}  
public static void run() {  
    Scanner sc = 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 = sc.nextInt();  
        switch (choice) {  
            case 1:  
                addEmployee();  
                break;  
            case 2:  
                displayEmployees();  
                break;  
            case 3:  
                System.out.println("Exiting...");  
                return;  
            default:  
                System.out.println("Invalid option. Try again.");  
        }  
    }  
}
```



# **DEPARTMENT OF**

# **COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

## **4. Output:**

```
P:\exp 4 java>cd "p:\exp 4 java\" && javac Main.java && java Main
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8

Main Menu:
1. Sum With Autoboxing
2. Student Serialization
3. Employee Manager
4. Exit
Choose an option: 1
Sum of numbers: 60

Main Menu:
1. Sum With Autoboxing
2. Student Serialization
3. Employee Manager
4. Exit
Choose an option: 2
Student data saved.
Deserialized Student:
ID: 101, Name: Alice, GPA: 3.9

Main Menu:
1. Sum With Autoboxing
2. Student Serialization
3. Employee Manager
4. Exit
Choose an option: 3

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID (Integer): 111
Enter Employee Name: amit
Enter Designation: software enginner
Enter Salary: 5000
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID (Integer): 112
Enter Employee Name: shivam
Enter Designation: software engineer
Enter Salary: 4000
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 2
Employee List:
ID: 111, Name: shivam, Designation: sd, Salary: 50000.0
ID: 111, Name: amit, Designation: software enginner, Salary: 5000.0
ID: 112, Name: shivam, Designation: software engineer, Salary: 4000.0
```



**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

### **5. Learning Outcome:**

- Learn how to use the graphics.h library for drawing basic shapes and setting up a graphical environment in Dev-C++.
- Gain hands-on experience with two different methods: the Circle Generator Algorithm (direct computation) and the Midpoint Circle Algorithm.
- Understood the concept of coordinating system.
- Learn about various command that is used in drawing a circle.