

Experiment 6

Student Name: Shaurya Anand

UID: 22BCS15925

Branch: BE-C.S.E

Section/Group: 639-B

Semester: 6th

Date of Performance: 05-03-25

Subject Name: Project Based Learning

Subject Code: 22CSH-359

In Java with Lab

1. Problem 6.1: 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.

Implementation/Code:

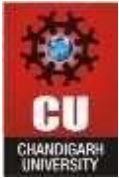
```
import java.util.ArrayList;
import java.util.Scanner;

class Employee {
    int id;
    String name;
    double salary;

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

    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Salary: $" + salary;
    }
}

public class EmployeeManagement {
    private static ArrayList<Employee> employeeList = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
public static void main(String[] args) {
    while (true) {
        System.out.println("\nEmployee Management System");

        System.out.println("1. Add Employee");
        System.out.println("2. Update Employee");
        System.out.println("3. Remove Employee");
        System.out.println("4. Search Employee");
        System.out.println("5. Display All Employees");
        System.out.println("6. Exit");
        System.out.print("Choose an option: ");

        int choice = scanner.nextInt();
        scanner.nextLine();

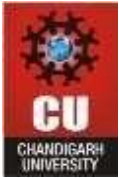
        switch (choice) {
            case 1 -> addEmployee();
            case 2 -> updateEmployee();
            case 3 -> removeEmployee();
            case 4 -> searchEmployee();
            case 5 -> displayEmployees();
            case 6 -> {
                System.out.println("Exiting program. Goodbye!");
                return;
            }
            default -> System.out.println("Invalid choice! Please try again.");
        }
    }
}

private static void addEmployee() {
    System.out.print("Enter Employee ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    System.out.print("Enter Employee Name: ");
    String name = scanner.nextLine();

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

    employeeList.add(new Employee(id, name, salary));
    System.out.println("Employee added successfully!");
}
```



```
private static void updateEmployee() {
    System.out.print("Enter Employee ID to update: ");
    int id = scanner.nextInt();

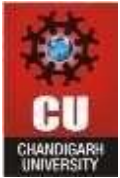
    scanner.nextLine();    for
(Employee emp : employeeList) {
    if (emp.id == id) {
        System.out.print("Enter new Name: ");
        emp.name = scanner.nextLine();
        System.out.print("Enter new Salary: ");
        emp.salary = scanner.nextDouble();
        System.out.println("Employee updated successfully!");
    }
    return;
    }
    System.out.println("Employee ID not found!");
}

private static void removeEmployee() {
    System.out.print("Enter Employee ID to remove: ");
    int id = scanner.nextInt();

    for (Employee emp : employeeList) {
    if (emp.id == id) {
        employeeList.remove(emp);
        System.out.println("Employee removed successfully!");
    }
    return;
    }
    System.out.println("Employee ID not found!");
}

private static void searchEmployee() {
    System.out.print("Enter Employee ID to search: ");
    int id = scanner.nextInt();

    for (Employee emp : employeeList) {
        if (emp.id == id) {
            System.out.println("Employee Found: " + emp);
        }
    }
    return;
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }  
    }  
    System.out.println("Employee ID not found!");  
}  
  
private static void displayEmployees() {  
  
    if (employeeList.isEmpty()) {  
        System.out.println("No employees found!");  
    } else {  
        System.out.println("Employee List:");  
        for (Employee emp : employeeList) {  
            System.out.println(emp);  
        }  
    }  
}  
}
```

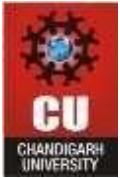
OUTPUT :

```
Employee Management System  
1. Add Employee  
2. Display All Employees  
3. Exit  
Choose an option: 1  
Enter Employee ID: 101  
Enter Employee Name: John Doe  
Enter Employee Designation: Manager  
Enter Employee Salary: 75000  
Employee added successfully!  
  
Choose an option: 2  
Employee List:  
ID: 101, Name: John Doe, Designation: Manager, Salary: $75000.0
```

Figure 6.1

2. Problem 6.2 :- Create a Java program to serialize and deserialize a Student object

Implementation Code :- import
java.io.*;



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

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

    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Grade: " + grade;
    }
}

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

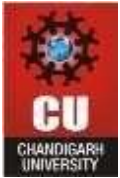
        // Serialize Student object
        Student student = new Student(101, "Alice", 92.5);
        serializeStudent(student, filename);

        // Deserialize Student object
        Student deserializedStudent = deserializeStudent(filename);
        System.out.println("Deserialized Student: " + deserializedStudent);
    }

    private static void serializeStudent(Student student, String filename) {
        try (ObjectOutputStream out = new ObjectOutputStream(new
        FileOutputStream(filename))) {
            out.writeObject(student);
            System.out.println("Student object serialized successfully.");
        } catch (IOException e) {
            System.out.println("Serialization error: " + e.getMessage());
        }
    }

    private static Student deserializeStudent(String filename) {

```

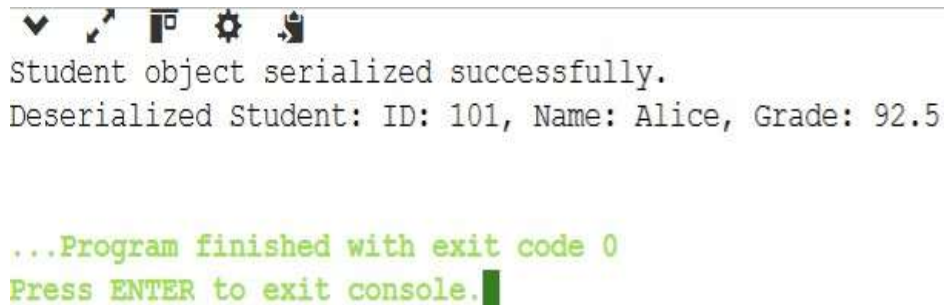


DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {  
    return (Student) in.readObject();  
  
} catch (IOException | ClassNotFoundException e) {  
  
    System.out.println("Deserialization error: " + e.getMessage());  
    return null;  
}  
}
```

OUTPUT :-



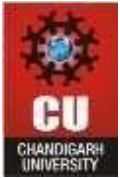
```
Student object serialized successfully.  
Deserialized Student: ID: 101, Name: Alice, Grade: 92.5  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Figure 6.2

3. Problem 6.3 : 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()).

Implementation Code :-

```
import  
java.util.ArrayList;  
import java.util.Scanner;  
  
public class AutoboxingUnboxingSum {  
    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 (true) {
```



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
String input = scanner.nextLine();
if (input.equalsIgnoreCase("done")) break;
try {
    numbers.add(Integer.parseInt(input)); // Autoboxing
} catch (NumberFormatException e) {
    System.out.println("Invalid input! Please enter a valid integer.");
}
}

int sum = calculateSum(numbers);
System.out.println("Sum of the entered numbers: " + sum);
}

private static int calculateSum(ArrayList<Integer> numbers)
{
    int sum = 0;
    for (Integer num : numbers) {
        sum += num; // Unboxing
    }
    return sum;
}
}
```

OUTPUT :-

```
Enter numbers (type 'done' to finish):
10
20
30
done
Sum of the entered numbers: 60
```

Figure 6.3