Experiment-5

Student Name: Vaibhav Kumar **UID:** 23BCS13386
Branch: B.E-C.S.E **Section/Group:** 23KRG-2B
Semester: 5th **Date of Performance:** 01/10/2025
Subject Name: PBLJ **Subject Code:** 23CSH-304

Easy Level

1. **Aim:** 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()).

2. **Objective:** To demonstrate the use of Java Wrapper classes and automatic conversion between primitive types and their wrapper equivalents.

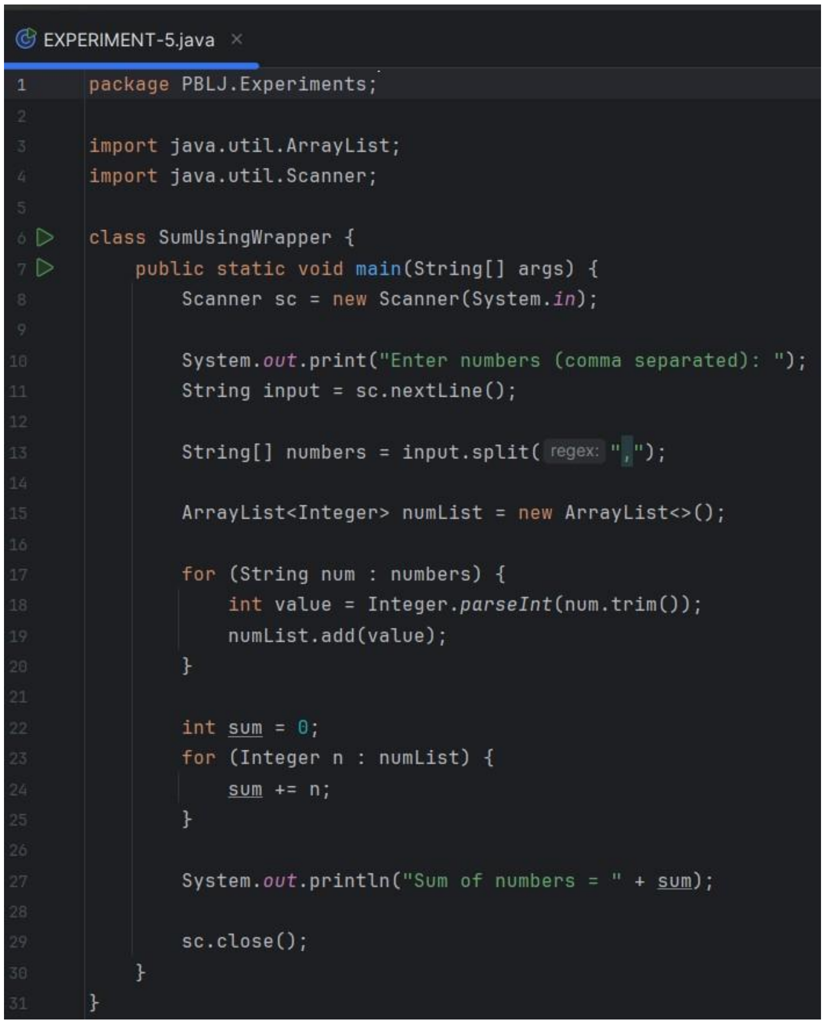
3. **Input/Apparatus Used:** Java ArrayList<Integer>, wrapper methods (parseInt(), valueOf()), autoboxing/unboxing.

4. **Procedure:**
   Accept a comma-separated string of numbers from the user.
   Parse each value using Integer.parseInt().
   Store the values in an ArrayList<Integer>, leveraging autoboxing.
   Iterate through the list and calculate the sum using unboxing.
   Display the total sum.

5.
Sample Input:
Enter numbers: 10, 20, 30, 40

Sample Output:
Sum of numbers = 100

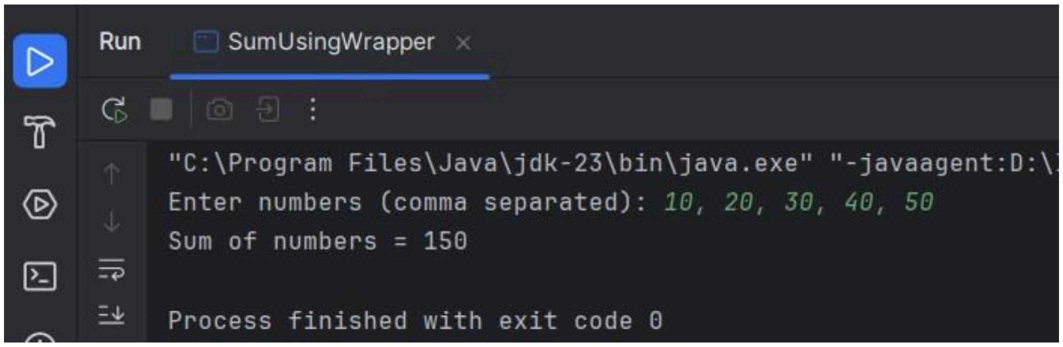
6. Code:



EXPERIMENT-5.java

```java
package PBLJ.Experiments;

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

class SumUsingWrapper {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter numbers (comma separated): ");
        String input = sc.nextLine();

        String[] numbers = input.split( regex: ",");

        ArrayList<Integer> numList = new ArrayList<>();

        for (String num : numbers) {
            int value = Integer.parseInt(num.trim());
            numList.add(value);
        }

        int sum = 0;
        for (Integer n : numList) {
            sum += n;
        }

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

        sc.close();
    }
}
```

7. Output:



```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\
Enter numbers (comma separated): 10, 20, 30, 40, 50
Sum of numbers = 150

Process finished with exit code 0
```

Medium Level

1. **Aim:** reate 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.

2. **Objective:** To demonstrate object serialization, file handling, and exception management in Java.

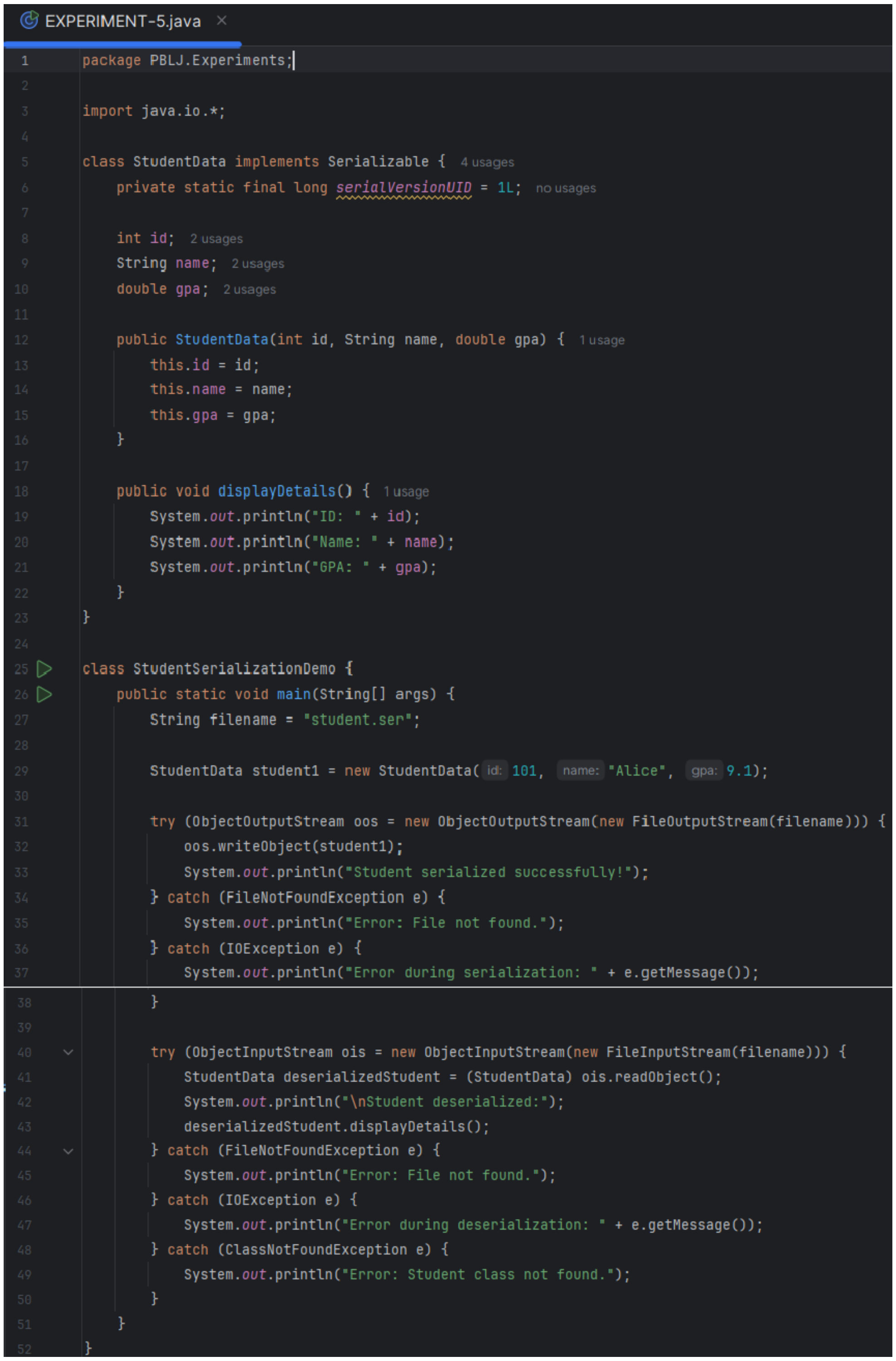
3. **Input/Apparatus Used:** ObjectOutputStream, ObjectInputStream, Serializable interface, FileOutputStream, FileInputStream.

4. **Procedure:**
   1. Define a Student class implementing Serializable with id, name, and GPA.
   2. Create an object and serialize it using ObjectOutputStream.
   3. Save the object to a file.
   4. Deserialize the object from the file using ObjectInputStream.
   5. Handle exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

5. **Sample Output :**

```
Student serialized successfully!

Student deserialized:

ID: 101

Name: Alice

GPA: 9.1
```

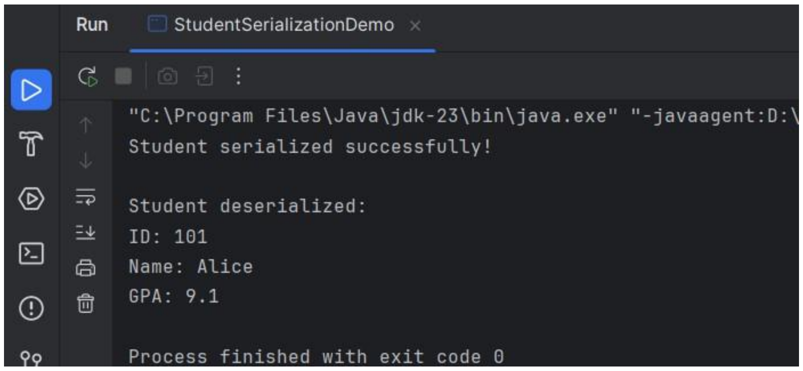
6. **Code:**



EXPERIMENT-5.java

```java
package PBLJ.Experiments;

import java.io.*;

class StudentData implements Serializable {  4 usages
    private static final long serialVersionUID = 1L;  no usages

    int id;  2 usages
    String name;  2 usages
    double gpa;  2 usages

    public StudentData(int id, String name, double gpa) {  1 usage
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    public void displayDetails() {  1 usage
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("GPA: " + gpa);
    }
}

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

        StudentData student1 = new StudentData( id: 101,  name: "Alice",  gpa: 9.1);

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(student1);
            System.out.println("Student serialized successfully!");
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
        } catch (IOException e) {
            System.out.println("Error during serialization: " + e.getMessage());
        }

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            StudentData deserializedStudent = (StudentData) ois.readObject();
            System.out.println("\nStudent deserialized:");
            deserializedStudent.displayDetails();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
        } catch (IOException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Error: Student class not found.");
        }
    }
}
```

7. Output:



```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\
Student serialized successfully!

Student deserialized:
ID: 101
Name: Alice
GPA: 9.1

Process finished with exit code 0
```

Hard Level

1. 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, 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

2. Objective: To combine object-oriented programming, file handling, and menu-driven console interaction.

3. Input/Apparatus Used: Java I/O (BufferedWriter, BufferedReader, FileWriter, FileReader), Scanner, ArrayList.

4. Procedure:

1. Present a menu:

a) Add Employee

b) Display All

c) Exit

2. On choosing Add, take input for:

a) Employee Name

b) Employee ID

c) Designation

d) Salary

3. Write this data to a file.

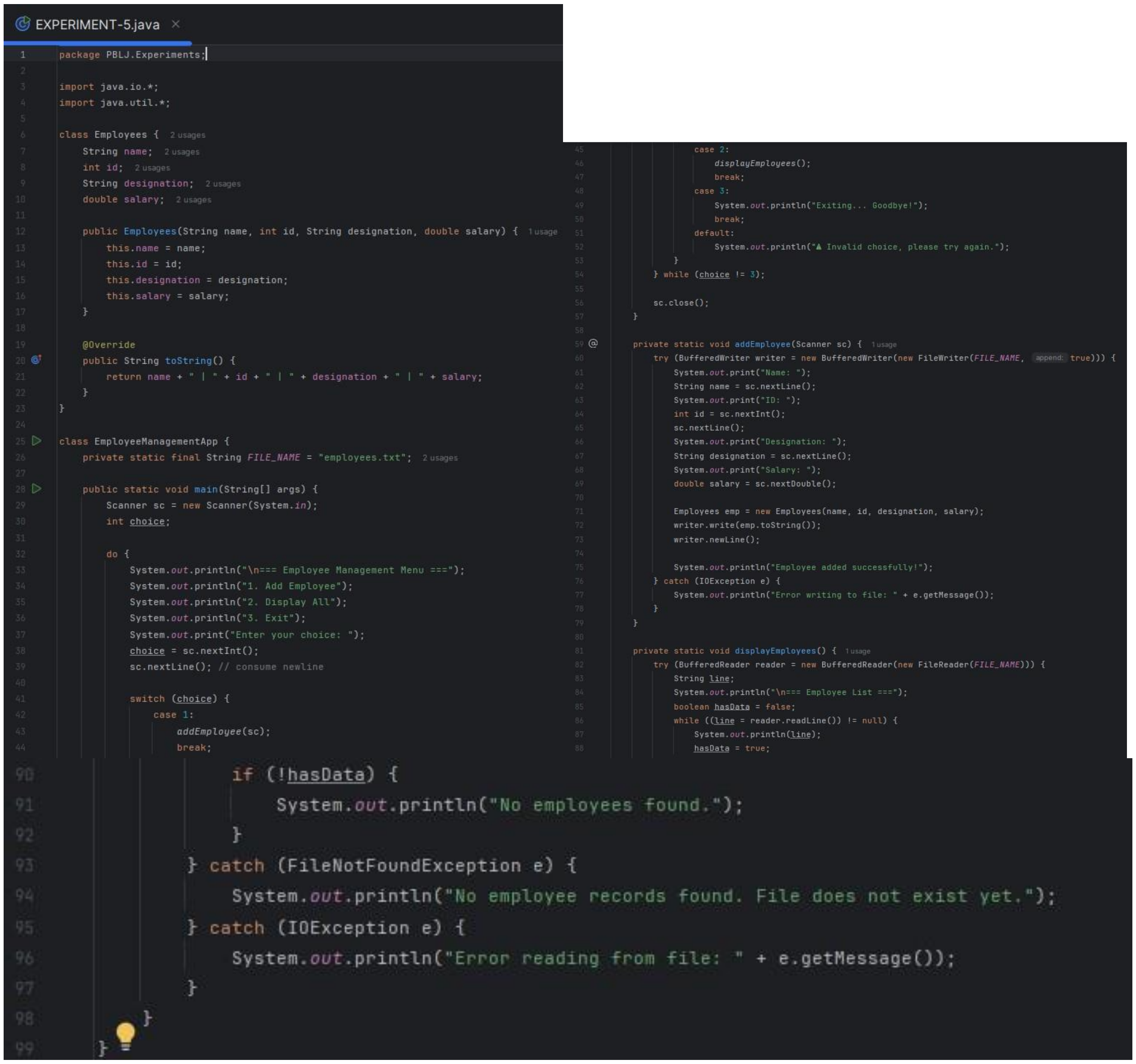
4. On choosing Display, read and display all employee data from the file.

5. Exit on selection of option 3.

Sample Output:

```
Menu:
1. Add Employee
2. Display All
3. Exit

Enter choice: 1
Name: John
ID: 1001
Designation: Manager
Salary: 75000
Employee added successfully!
Enter choice: 2
Employee List:
John | 1001 | Manager | 75000
```

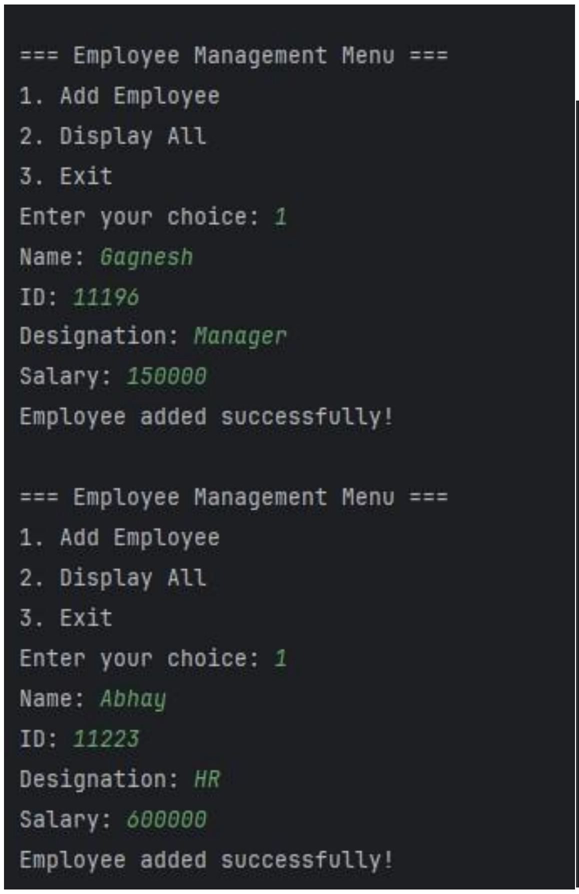
5. Code:



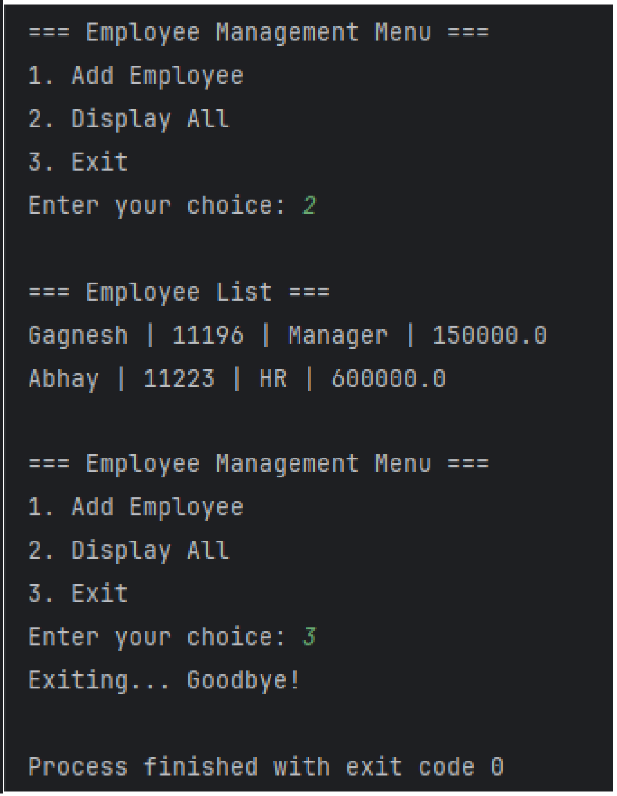
EXPERIMENT-5.java

```java
package PBLJ.Experiments;

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

class Employees {  2 usages
    String name;  2 usages
    int id;  2 usages
    String designation;  2 usages
    double salary;  2 usages

    public Employees(String name, int id, String designation, double salary) {  1 usage
        this.name = name;
        this.id = id;
        this.designation = designation;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return name + " | " + id + " | " + designation + " | " + salary;
    }
}

class EmployeeManagementApp {
    private static final String FILE_NAME = "employees.txt";  2 usages

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n=== Employee Management Menu ===");
            System.out.println("1. Add Employee");
            System.out.println("2. Display All");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    addEmployee(sc);
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.out.println("Exiting... Goodbye!");
                    break;
                default:
                    System.out.println("⚠ Invalid choice, please try again.");
            }
        } while (choice != 3);

        sc.close();
    }

    private static void addEmployee(Scanner sc) {  1 usage
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE_NAME,  append: true))) {
            System.out.print("Name: ");
            String name = sc.nextLine();
            System.out.print("ID: ");
            int id = sc.nextInt();
            sc.nextLine();
            System.out.print("Designation: ");
            String designation = sc.nextLine();
            System.out.print("Salary: ");
            double salary = sc.nextDouble();

            Employees emp = new Employees(name, id, designation, salary);
            writer.write(emp.toString());
            writer.newLine();

            System.out.println("Employee added successfully!");
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }

    private static void displayEmployees() {  1 usage
        try (BufferedReader reader = new BufferedReader(new FileReader(FILE_NAME))) {
            String line;
            System.out.println("\n=== Employee List ===");
            boolean hasData = false;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
                hasData = true;
            if (!hasData) {
                System.out.println("No employees found.");
            }
        } catch (FileNotFoundException e) {
            System.out.println("No employee records found. File does not exist yet.");
        } catch (IOException e) {
            System.out.println("Error reading from file: " + e.getMessage());
        }
    }
}
```

6. Output:



```
=== Employee Management Menu ===
1. Add Employee
2. Display All
3. Exit
Enter your choice: 1
Name: Gagnesh
ID: 11196
Designation: Manager
Salary: 150000
Employee added successfully!

=== Employee Management Menu ===
1. Add Employee
2. Display All
3. Exit
Enter your choice: 1
Name: Abhay
ID: 11223
Designation: HR
Salary: 600000
Employee added successfully!
```



```
=== Employee Management Menu ===
1. Add Employee
2. Display All
3. Exit
Enter your choice: 2

=== Employee List ===
Gagnesh | 11196 | Manager | 150000.0
Abhay | 11223 | HR | 600000.0

=== Employee Management Menu ===
1. Add Employee
2. Display All
3. Exit
Enter your choice: 3
Exiting... Goodbye!

Process finished with exit code 0
```