Experiment - 5

Student Name: Garvi Dabas **UID:** 23BCS11346
Branch: BE-CSE **Section/Group:** KRG-2B
Semester: 5th **Date of Performance:** 18/9/25
Subject Name: Project Based Learning in Java
Subject Code: 23CSH-304

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

Medium-level Problem-

Aim: Create a program to **collect and store cards**, and assist users in finding all cards of a given symbol using Collection interfaces.

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

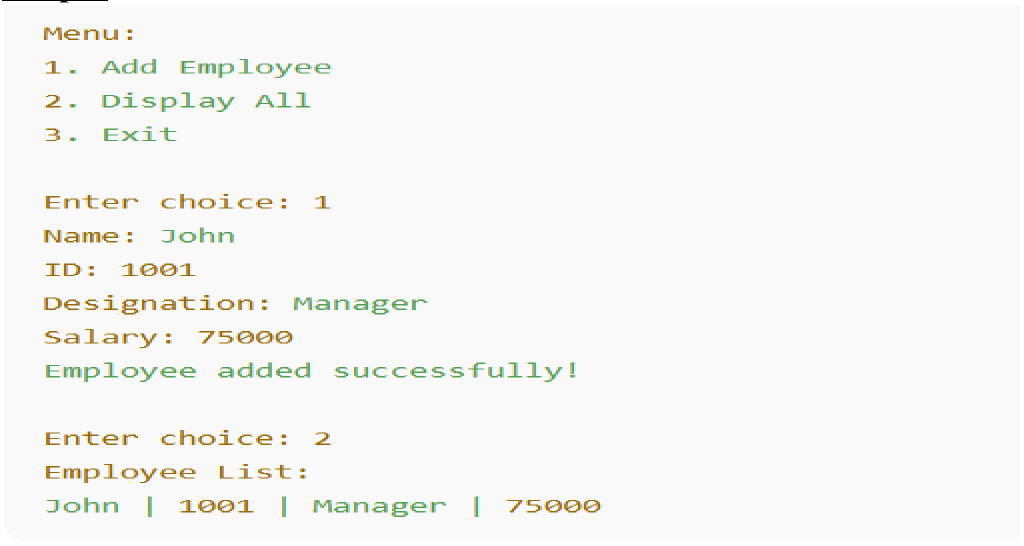
Procedure:

1. Define a Student class implementing Serializable with id, name, and GPA.
2. Create a Student 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: FileNotFoundException, IOException, ClassNotFoundException.

Code -

```
package exp2;

import java.io.*;

class Student implements Serializable {
int id;
String name;
double gpa;

Student(int id, String name, double gpa) {
this.id = id;
this.name = name;
this.gpa = gpa;
}
public String toString() {
return "ID: " + id + "\nName: " + name + "\nGPA: " + gpa;
}
}

public class Medium {
public static void main(String[] args) {
Student s = new Student(101, "Alice", 9.1);

// Serialization
try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("student.dat"))) {
out.writeObject(s);
System.out.println("Student serialized successfully!");
} catch (IOException e) {
e.printStackTrace();
}

// Deserialization
try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("student.dat"))) {
Student s2 = (Student) in.readObject();
System.out.println("Student deserialized:");
System.out.println(s2);
} catch (IOException | ClassNotFoundException e) {
e.printStackTrace();
}
}
}
```

<u>Output</u> -



```
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
```

Hard- Level Problem -

Aim : Develop a **menu-driven Java application** to store and display **employee details** using **file handling**.

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

Procedure:

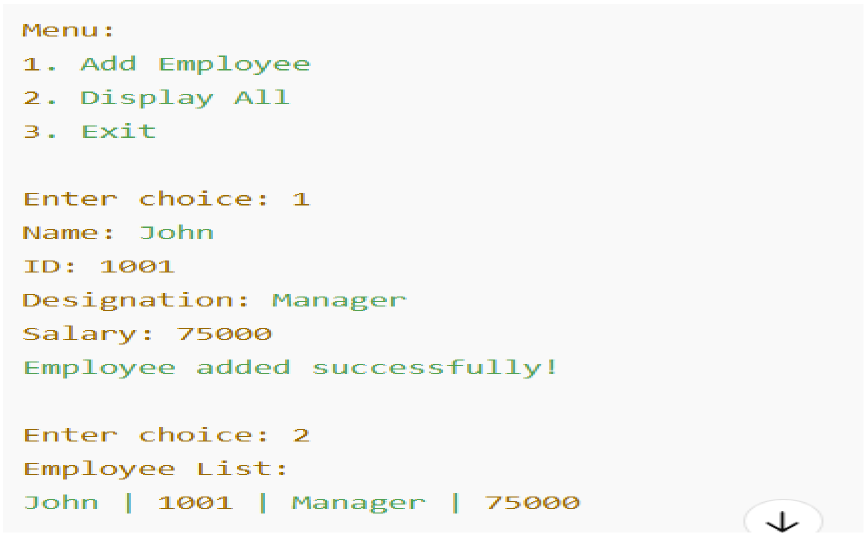
1. Present a menu:
   - Add Employee
   - Display All
   - Exit
2. On choosing Add Employee, take input for:
   - Employee Name
   - Employee ID
   - Designation
   - Salary
3. Write this data to a file.
4. On choosing **Display All**, read and display all employee data from the file.
5. Exit on selection of option 3.

Code :

```
package exp2;
import java.io.*;
import java.util.Scanner;


public class Hard {
public static void main(String[] args) throws IOException {
Scanner sc = new Scanner(System.in);
String fileName = "employees.txt";
while (true) {
System.out.println("\nMenu:\n1. Add Employee\n2. Display All\n3. Exit");
System.out.print("Enter choice: ");
int choice = sc.nextInt();
sc.nextLine(); // consume newline
```

```
switch (choice) {
case 1:
System.out.print("Name: ");
String name = sc.nextLine();
System.out.print("ID: ");
String id = sc.nextLine();
System.out.print("Designation: ");
String designation = sc.nextLine();
System.out.print("Salary: ");
String salary = sc.nextLine();

try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileName, true))) {
bw.write(name + "|" + id + "|" + designation + "|" + salary);
bw.newLine();
System.out.println("Employee added successfully!");
}
break;

case 2:
System.out.println("Employee List:");
try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
String line;
while ((line = br.readLine()) != null) {
System.out.println(line.replace("|", " | "));
}
} catch (FileNotFoundException e) {
System.out.println("No employee data found.");
}
break;
case 3:
System.out.println("Exiting...");
return;

default:
System.out.println("Invalid choice.");
}
}
}
}
```

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
```

Conclusion:

1. Learned autoboxing and unboxing to convert between primitives and wrapper objects automatically.
2. Understood Java wrapper classes and their use in collections and data processing.
3. Implemented serialization and deserialization to persist and retrieve object states from files.