Name:	PRINCE
UID:	23BCS11945
Subject:	PBLJ
Section:	622 B

Aim: To develop Java programs using autoboxing, serialization, file handling.

Part A: Sum of Integers Using Autoboxing and Unboxing

Algorithm:

1. Start.
2. Declare two Integer objects and assign primitive int values (autoboxing happens).
3. Add the two Integer objects; they are unboxed to int automatically.
4. Store the result in an int variable.
5. Display the sum.
6. Stop.

Code:

```
import java.util.*;
public class SumIntegers {
   public static void main(String[] args) {
      Integer a = 10;
      Integer b = 20;
      int sum = a + b;
      System.out.println("Sum: " + sum);
   }
}
```

Output:

```
Output                                    Clear
Sum: 30

=== Code Execution Successful ===
```

Part B: Serialization and Deserialization of a Student Object

Algorithm:

1. Start.
2. Define a Student class implementing Serializable.
3. Create a Student object with id and name.
4. Open a FileOutputStream and wrap it with an ObjectOutputStream.
5. Write the Student object to the file (serialization).
6. Close the streams.
7. Open a FileInputStream and wrap it with an ObjectInputStream.
8. Read the Student object from the file (deserialization).
9. Close the streams.
10. Display the id and name of the deserialized object.
11. Stop.

Code:

```
import java.io.*;

class Student implements Serializable {

   int id;

   String name;

   Student(int id, String name) {

      this.id = id;

      this.name = name;

   }

}

public class Main {

   public static void main(String[] args) {

      try {

         Student s1 = new Student(1, "Karan");

         FileOutputStream fos = new FileOutputStream("student.ser");

         ObjectOutputStream oos = new ObjectOutputStream(fos);

         oos.writeObject(s1);

         oos.close();

         fos.close();
```

```
            FileInputStream fis = new FileInputStream("student.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Student s2 = (Student) ois.readObject();
            ois.close();
            fis.close();
            System.out.println("ID: " + s2.id + ", Name: " + s2.name);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Output                                                    Clear
ID: 1, Name: Karan

=== Code Execution Successful ===
```

Part C: Menu-Based Employee Management System Using File Handling

Algorithm:

1. Start.
2. Define an Employee class with id, name, and salary.
3. Create a menu with options:
   Option 1: Add Employee
   1. Accept id, name, and salary from the user.
   2. Convert to string and append to a text file (employees.txt).
   Option 2: View Employees
   1. Open the file and read each line.
   2. Display employee details.
   Option 3: Exit program.
4. Repeat menu until user chooses Exit.
5. Stop.

Code:

```
import java.io.*;
import java.util.*;
class Employee {
```

```
    int id;
    String name;
    double salary;
    Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    public String toString() {
        return id + "," + name + "," + salary;
    }
}

public class EmployeeManagementSystem {
    static final String FILE = "employees.txt";
    public static void addEmployee(Employee e) {
        try {
            FileWriter fw = new FileWriter(FILE, true);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(e.toString());
            bw.newLine();
            bw.close();
            fw.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
    public static void viewEmployees() {
        try {
            BufferedReader br = new BufferedReader(new FileReader(FILE));
```

```
        String line;
        while ((line = br.readLine()) != null) {
            System.out.println(line);
        }
        br.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int choice;
    do {
        System.out.println("1. Add Employee");
        System.out.println("2. View Employees");
        System.out.println("3. Exit");
        System.out.print("Enter choice: ");
        choice = sc.nextInt();
        switch (choice) {
            case 1:
                System.out.print("Enter ID: ");
                int id = sc.nextInt();
                sc.nextLine();
                System.out.print("Enter Name: ");
                String name = sc.nextLine();
                System.out.print("Enter Salary: ");
                double salary = sc.nextDouble();
                addEmployee(new Employee(id, name, salary));
                break;
            case 2:
```

```
                viewEmployees();
                break;
            case 3:
                System.out.println("Exiting...");
                break;
        }
    } while (choice != 3);
    sc.close();
  }
}
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL        Run: EmployeeManagementSystem

EmployeeManagementSystem
1. Add Employee
2. View Employees
3. Exit
Enter choice: 1
Enter ID: 1
Enter Name: Karan
Enter Salary: 100000
1. Add Employee
2. View Employees
3. Exit
Enter choice: 1
Enter ID: 2
Enter Name: Arjun
Enter Salary: 95000
1. Add Employee
2. View Employees
3. Exit
Enter choice: 2
1,Karan,100000.0
2,Arjun,95000.0
1. Add Employee
2. View Employees
3. Exit
Enter choice: 3
Exiting...
root@907f021835a7:~/my-project#
```

```
EmployeeManagementSystem.java    employees.txt
employees.txt
1    1,Karan,100000.0
2    2,Arjun,95000.0
```