Experiment 5

Name: Gourvi

UID: 22BCS11023

Section: 616 A

Date: 17/02/25

Easy Level: Autoboxing, Unboxing, and Parsing Strings

```
import java.util.*;

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

        System.out.println("Enter numbers separated by space:");
        String input = sc.nextLine();

        String[] numbers = input.split(" ");
        int sum = 0;

        for (String num : numbers) {
            sum += Integer.parseInt(num); // Parsing string to Integer
        }

        System.out.println("Sum of the numbers: " + sum);
sc.close();
    }
}
```

Medium Level: Serialization and Deserialization of a Student Object

```
import java.io.*;
```

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

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

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

public class StudentSerialization {
public static void main(String[] args) {
      Student student = new Student(101, "John Doe", 3.8);         try
(ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("student.dat"))) {
oos.writeObject(student);
         System.out.println("Student object serialized.");
      } catch (IOException e) {
         System.out.println("Serialization error: " + e.getMessage());
      }

      try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("student.dat"))) {
         Student deserializedStudent = (Student) ois.readObject();
         System.out.println("Deserialized Student:");
deserializedStudent.display();         } catch
(FileNotFoundException e) {
         System.out.println("File not found.");
      } catch (IOException | ClassNotFoundException e) {
         System.out.println("Deserialization error: " + e.getMessage());
      }
   }
}
```

Hard Level: Menu-Based Employee Management System

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

class Employee implements Serializable {
   private static final long serialVersionUID = 1L;
int id;
   String name, designation;
double salary;

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

   public void display() {
      System.out.println("ID: " + id + ", Name: " + name + ", Designation:
" + designation + ", Salary: " + salary);
   }
}

public class EmployeeManagement {
   private static final String FILE_NAME = "employees.dat";

   public static void addEmployee() {
Scanner sc = new Scanner(System.in);
System.out.print("Enter Employee ID: ");
      int id = sc.nextInt();
sc.nextLine();
      System.out.print("Enter Name: ");
      String name = sc.nextLine();
      System.out.print("Enter Designation: ");
```

```
        String designation = sc.nextLine();
System.out.print("Enter Salary: ");
        double salary = sc.nextDouble();

        Employee emp = new Employee(id, name, designation, salary);
        List<Employee> employees = readEmployees();
        employees.add(emp);
writeEmployees(employees);
        System.out.println("Employee added successfully!");
    }

    public static void displayEmployees() {
List<Employee> employees = readEmployees();         if
(employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            System.out.println("Employee Details:");
for (Employee emp : employees) {
                emp.display();
            }
        }
    }

    @SuppressWarnings("unchecked")     private static
List<Employee> readEmployees() {         List<Employee>
employees = new ArrayList<>();         try (ObjectInputStream
ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            employees = (List<Employee>) ois.readObject();
        } catch (FileNotFoundException e) {
            // No previous data, return empty list
        } catch (IOException | ClassNotFoundException e) {
System.out.println("Error reading employees: " + e.getMessage());
        }
        return employees;
    }

    private static void writeEmployees(List<Employee> employees) {
try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
```

```
            oos.writeObject(employees);
        } catch (IOException e) {
            System.out.println("Error saving employees: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("\n1. Add Employee\n2. Display All
Employees\n3. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
switch (choice) {                case
1:                  addEmployee();
                    break;
case 2:
                    displayEmployees();
                    break;
case 3:
                    System.out.println("Exiting...");
                    sc.close();
return;                 default:
                    System.out.println("Invalid choice, try again.");
            }
        }
    }
}
```