Experiment 5

Student Name: Anchal **UID: 22BCS10564**
Branch: CSE **Section: 22KPIT-902/A**
Semester: 6th **Date of Performance:28/02/2025** **Subject: Project Based Learning in Java** **Subject Code: 22CSH-359**

1. **Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

2. **Objective 1: Easy Level**
   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()).

3. **Code/Implementation:**

```
import java.util.ArrayList; import
java.util.List;

public class Experiment5A{
    public static void main(String args[]){
List<Integer> numbers = new ArrayList<>();
        //Autoboxing : int is converted to Integer automatically
numbers.add(10);         numbers.add(20);
numbers.add(30);

        //Unboxing: conversion of wrapper class objects back into
primitive data types         int sum=0;
        for (Integer num: numbers){
sum += num; //Unboxing
        }

        System.out.println("Sum of integers: " + sum);

        String numStr = "50";
        int parsedNumber = Integer.parseInt(numStr);
System.out.println("Parsed number from string " + parsedNumber);

    }
}
```

Output:

```
Sum of integers: 60
Parsed number from string 50
```

4. **Objective 2: Medium Level**

Create 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.

5. **Code/Implementation:**

```
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
Experiment5B{
    public static void main(String[] args) {
        String filename = "student.ser";

        Student student = new Student(1114, "Manyata",8.5);

        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());
        }

        //Deserialization
        try(ObjectInputStream in = new ObjectInputStream(new
FileInputStream(filename))){
```

```
                Student deserializedStudent = (Student) in.readObject();
        System.out.println("Deserialized Student details: ");
        deserializedStudent.display();
                } catch(FileNotFoundException e){
                    System.out.println("File not found: " + e.getMessage());
                } catch(IOException e){
                    System.out.println("IO Error: "+ e.getMessage());
                } catch(ClassNotFoundException e){
                    System.out.println("Class not found: " +e.getMessage());
                }
            } }
```

Output:

```
Student object Serialized successfully!
Deserialized Student details:
ID: 1114, Name: Manyata, GPA: 8.5
```

6. Objective 3 : Hard Level

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.

7. Code/Implementation:

```
import java.io.*; import
java.util.*;
 class Employee implements Serializable
{    int id;
    String name;
String designation;
double salary;

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

```
    @Override     public String toString() {         return "ID: " + id + ", Name:
" + name + ", Designation: " + designation + ", Salary: " + salary;
    }
}  public class
Experiment5C {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        File file = new File("employees.dat");

        // Load existing employees from file
if (file.exists()) {
            try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(file))) {                 employees =
(ArrayList<Employee>) ois.readObject();
            } catch (Exception e) {
                System.out.println("Error reading file: " + e.getMessage());
            }
        }

        Scanner scanner = new Scanner(System.in);
         while (true)
{             try {
              System.out.println("\n1. Add an Employee");
              System.out.println("2. Display All");
              System.out.println("3. Exit");
              System.out.print("Enter your choice: ");

              if (!scanner.hasNextInt()) {
                  System.out.println("Invalid input! Please enter a number.");
scanner.next();  // Clear the invalid input                    continue;
               }
              int choice = scanner.nextInt();
              scanner.nextLine();  // Consume the newline character
               switch (choice)
{                    case 1:
                       System.out.print("Enter Employee ID: ");
int id = scanner.nextInt();
scanner.nextLine();  // Consume the newline

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

                       System.out.print("Enter Salary: ");
double salary = scanner.nextDouble();
scanner.nextLine();  // Consume the newline
```

```
                        employees.add(new Employee(id, name,
designation, salary));
                        System.out.println("Employee added successfully.");
                        // Save updated list to file
                        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(file))) {
oos.writeObject(employees);
                        } catch (IOException e) {
                            System.out.println("Error writing to file: " +
e.getMessage());                            }                            break;
                    case
2:
                        if (employees.isEmpty()) {
                            System.out.println("No employees to display.");
                        } else {
                            for (Employee emp : employees) {
                                System.out.println(emp);
                            }
}
break;
                    case
3:
                        System.out.println("Exiting the application.");
scanner.close();                            System.exit(0);
                        break;

                    default:
                        System.out.println("Invalid choice! Please enter 1, 2, or
3.");
                }
            } catch (Exception e) {
                System.out.println("Unexpected error: " + e.getMessage());
scanner.nextLine(); // Prevent infinite loop
            }
        }
    }
}
```

Output:

```
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 1
Enter Employee ID: 1008
Enter Employee Name: Manyata
Enter Designation: Software Engineer
Enter Salary: 1000000
Employee added successfully.

1. Add an Employee
2. Display All
3. Exit
Enter Employee ID: 2007
Enter Employee Name: Vaibhav
Enter Employee ID: 2007
Enter Employee Name: Vaibhav
Enter Designation: Manager
Enter Salary: 100000
Employee added successfully.
```

```
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 2
ID: 1008, Name: Manyata, Designation:
 Software Engineer, Salary: 1000000.0

ID: 2007, Name: Vaibhav, Designation:
 Manager, Salary: 100000.0

1. Add an Employee
2. Display All
3. Exit
Enter your choice: 3
Exiting the application.
```

8. Learning Outcomes:

- Understand how Java automatically converts primitives to wrapper classes.
- Learn to handle errors using `try-catch` for smooth program execution.
- Gain knowledge of saving and retrieving objects using serialization.
- Develop a menu-driven program for interactive employee management.