



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 5

Student Name: Khyati Singh

Branch: CSE

Semester: 6

Subject Name: Java with Lab

UID: 22BCS16405

Section/Group: 901/DL/A

Date of Performance: 24/02/25

Subject Code: 22CSH-359

- 1. Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.
- 2. Objective:** To implement and analyze wrapper classes in Java- Integer, Character, Long, Boolean. Autoboxing and Unboxing. Byte stream, Character stream, Object serialization, cloning. Introduce lambda syntax, functional interfaces, method references, stream operations, sorting, filtering, mapping, reducing.

3. Implementation/Code:

```
a. import java.util.ArrayList;
import java.util.List;
public class SumCalculator{
    public static Integer parseStringToInteger(String str)
    { return Integer.parseInt(str);
    }
    public static int calculateSum(List<Integer> numbers)
    { int sum = 0;
      for (Integer num : numbers)
      { sum += num;
      }
      return sum;
    }
    public static void main(String[] args) {
        String[] numberStrings = { "10", "20", "30", "40", "50" };
        List<Integer> numbers = new ArrayList<>();
        for (String numStr : numberStrings)
            { numbers.add(parseStringToInteger(numStr));
            }
        int sum = calculateSum(numbers);
        System.out.println("Sum of numbers: " + sum);
    }
}
```

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

class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private 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("Student ID: " + id);
      System.out.println("Name: " + name);
      System.out.println("GPA: " + gpa);
    }
}

public class StudentSerialization {
    private static final String FILE_NAME = "student.ser";
    public static void main(String[] args) {
        Student student = new Student(16405, "Khyati Singh", 8.2);
        serializeStudent(student);
        deserializeStudent();
    }
    private static void serializeStudent(Student student) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME)))
        { oos.writeObject(student);
          System.out.println("Student object serialized successfully.");
        } catch (IOException e) {
            System.out.println("Error during serialization: " + e.getMessage());
        }
    }
    private static void deserializeStudent() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
        { Student student = (Student) ois.readObject();
          System.out.println("Student object deserialized successfully.");
          student.display();
        }
    }
}
```

```
        } catch (FileNotFoundException e)
        { System.out.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        } catch (ClassNotFoundException e)
        { System.out.println("Class not found: " + e.getMessage());
        }
    }
}
    string combined = s + s;
    return combined.find(goal) != string::npos;
}
int missingNumber(vector<int>& nums)
{ int n = nums.size();
  int expected_sum = n * (n + 1) / 2;
  int actual_sum = accumulate(nums.begin(), nums.end(), 0);
  return expected_sum - actual_sum;
}
```

```
c. import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
class Employee implements Serializable
{ private static final long serialVersionUID =
  1L; private int id;
  private String name;
  private String designation;
  private 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() {
```



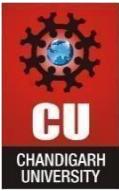
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser";
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        while (true)
        {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employee");
            System.out.println("2. Display All Employees");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice)
            {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.out.println("Exiting program.");
                    return;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        }
    }

    private static void addEmployee()
    {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Employee Name: ");
        String name = scanner.nextLine();
    }
}
```



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

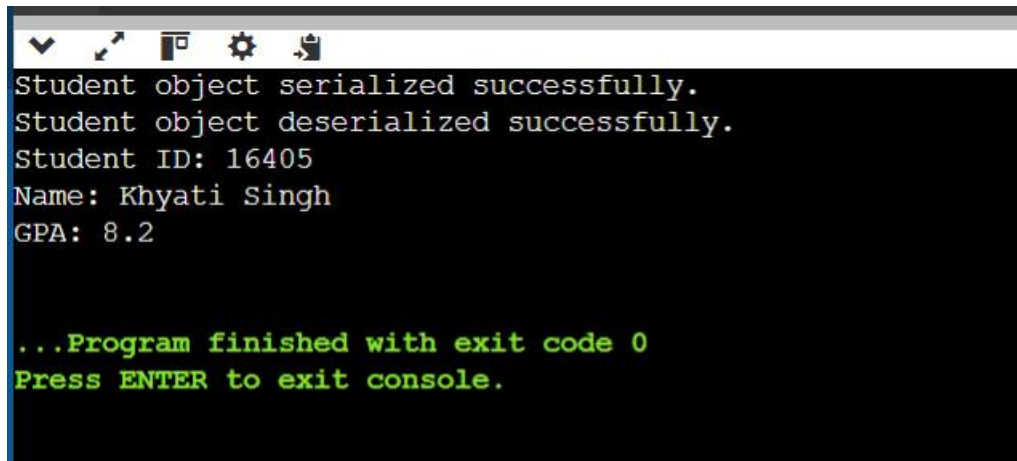
```
System.out.print("Enter Designation: ");
String designation = scanner.nextLine();
System.out.print("Enter Salary: ");
double salary = scanner.nextDouble();
Employee employee = new Employee(id, name, designation, salary);
List<Employee> employees = loadEmployees();
employees.add(employee);
saveEmployees(employees);
System.out.println("Employee added successfully!");
}
private static void displayEmployees()
{ List<Employee> employees =
loadEmployees(); if (employees.isEmpty()) {
    System.out.println("No employees found.");
} else {
    System.out.println("\nEmployee List:");
    for (Employee emp : employees) {
        emp.display();
    }
}
}
@SuppressWarnings("unchecked")
private static List<Employee> loadEmployees()
{ File file = new File(FILE_NAME);
if (!file.exists()) {
    return new ArrayList<>();
}
try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file)))
{ return (List<Employee>) ois.readObject();
} catch (IOException | ClassNotFoundException e)
{ System.out.println("Error reading file: " + e.getMessage());
return new ArrayList<>();
}
}
private static void saveEmployees(List<Employee> employees) {
try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME)))
{ oos.writeObject(employees);
```

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

4. Output:



```
Sum of numbers: 150  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```



```
Student object serialized successfully.  
Student object deserialized successfully.  
Student ID: 16405  
Name: Khyati Singh  
GPA: 8.2  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
input
Menu:
1. Add Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 16405
Enter Employee Name: Khyati Singh
Enter Designation: SDE Intern
Enter Salary: 75000
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Enter your choice: 2

Employee List:
ID: 16405, Name: Khyati Singh, Designation: SDE Intern, Salary: 75000.0
```

5. Learning Outcome:

- Learn file handling through object serialization and deserialization with exception handling.
- Develop skills in Java I/O streams for reading and writing data to files.
- Gain hands-on experience in managing structured data using serialization and deserialization.
- Explore functional programming concepts like lambda expressions, method references, and stream operations for efficient data processing.
- Build a menu-driven application for employee management using file handling and serialization