**Easy Problem:-**

import java.util.\*;

public class Main {

public static void main(String[] args) {

// Example list of numbers as strings

String[] numberStrings = {"10", "20", "30", "40", "50"};

// Convert strings to Integer objects

List<Integer> numbers = new ArrayList<>();

for (String numStr : numberStrings) {

numbers.add(Integer.parseInt(numStr)); // Autoboxing

}

// Calculate sum using unboxing

int sum = calculateSum(numbers);

// Print the result

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

}

public static int calculateSum(List<Integer> numbers) {

int sum = 0;

for (Integer num : numbers) {

sum += num; // Unboxing

}

return sum;

}

}

**Medium Level:-**

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;

}

@Override

public String toString() {

return "Student ID: " + id + ", Name: " + name + ", GPA: " + gpa;

}

}

public class Main {

public static void main(String[] args) {

String filename = "student.ser";

// Create a Student object

Student student = new Student(1, "John Doe", 3.8);

// Serialize the Student object

try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {

out.writeObject(student);

System.out.println("Student object serialized successfully.");

} catch (FileNotFoundException e) {

System.err.println("File not found: " + e.getMessage());

} catch (IOException e) {

System.err.println("IOException occurred: " + e.getMessage());

}

// Deserialize the Student object

try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {

Student deserializedStudent = (Student) in.readObject();

System.out.println("Deserialized Student: " + deserializedStudent);

} catch (FileNotFoundException e) {

System.err.println("File not found: " + e.getMessage());

} catch (IOException e) {

System.err.println("IOException occurred: " + e.getMessage());

} catch (ClassNotFoundException e) {

System.err.println("Class not found: " + e.getMessage());

}

}

}

**Hard Level:-**import java.io.\*;

import java.util.\*;

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;

}

@Override

public String toString() {

return "Employee ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;

}

}

public class Main {

private static final String FILE\_NAME = "employees.ser";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

List<Employee> employees = loadEmployees();

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Add an Employee");

System.out.println("2. Display All");

System.out.println("3. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

addEmployee(scanner, employees);

saveEmployees(employees);

break;

case 2:

displayEmployees(employees);

break;

case 3:

System.out.println("Exiting application.");

scanner.close();

return;

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

private static void addEmployee(Scanner scanner, List<Employee> employees) {

System.out.print("Enter Employee ID: ");

int id = scanner.nextInt();

scanner.nextLine(); // Consume 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();

employees.add(new Employee(id, name, designation, salary));

System.out.println("Employee added successfully!");

}

private static void displayEmployees(List<Employee> employees) {

if (employees.isEmpty()) {

System.out.println("No employees found.");

} else {

for (Employee emp : employees) {

System.out.println(emp);

}

}

}

private static void saveEmployees(List<Employee> employees) {

try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE\_NAME))) {

out.writeObject(employees);

} catch (IOException e) {

System.err.println("Error saving employees: " + e.getMessage());

}

}

private static List<Employee> loadEmployees() {

try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE\_NAME))) {

return (List<Employee>) in.readObject();

} catch (FileNotFoundException e) {

return new ArrayList<>();

} catch (IOException | ClassNotFoundException e) {

System.err.println("Error loading employees: " + e.getMessage());

return new ArrayList<>();

}

}

}