



## Experiment 5

**Student Name:** Pulkit

**Branch:** BE-CSE

**Semester:** 6<sup>th</sup>

**Subject Name:** Project Based Learning  
in Java with Lab

**UID:** 22BCS11634

**Section/Group:** 638/B

**Date of perf :-** 18-02-25

**Subject Code:** 22CSH-359

### **Problem-1**

1. **Aim** Develop a Java program to calculate the sum of a list of integers using autoboxing and unboxing. The program should also include methods to parse strings into their respective wrapper classes.

2. **Objective:**

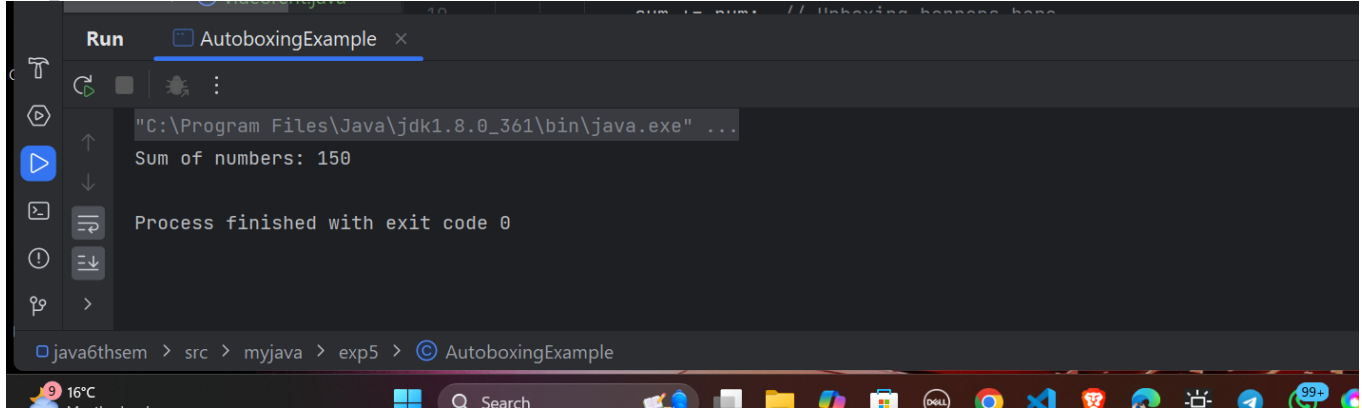
This program demonstrates autoboxing and unboxing in Java by automatically converting primitive types into wrapper objects and vice versa. It also showcases the usage of wrapper class methods such as `Integer.parseInt()`.

3. **Implementation/Code:**

```
package myjava.exp5;
import java.util.ArrayList;
import java.util.List;
public class AutoboxingExample {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        String[] strNumbers = {"10", "20", "30", "40", "50"};

        for (String str : strNumbers) {
            numbers.add(Integer.parseInt(str));
        }
        int sum = 0;
        for (Integer num : numbers) {
            sum += num;
        }
        System.out.println("Sum of numbers: " + sum);
    }
}
```

## 4. Output:



```
Run AutoboxingExample x
"C:\Program Files\Java\jdk1.8.0_361\bin\java.exe" ...
Sum of numbers: 150
Process finished with exit code 0
java6thsem > src > myjava > exp5 > AutoboxingExample
```

## 5. Learning Outcomes:

- Understanding autoboxing and unboxing in Java.
- Converting strings to wrapper class objects using Integer.parseInt().
- Performing arithmetic operations on wrapper objects.

## Problem-2

1. **Aim** Develop a Java program to serialize a Student object (containing ID, name, and GPA) and save it to a file. Deserialize the object and display student details while handling exceptions.
2. **Objective:**  
This program demonstrates object serialization and deserialization in Java. It highlights handling file operations using `ObjectOutputStream` and `ObjectInputStream` while managing exceptions like `FileNotFoundException`, `IOException`, and `ClassNotFoundException`.

3. **Implementation/Code:**

```
package myjava.exp5;
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;

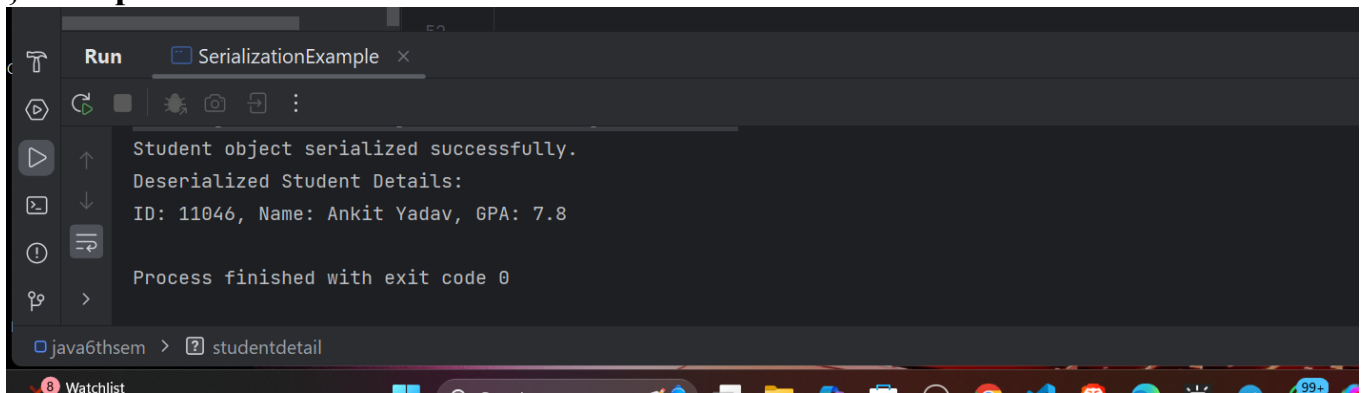
class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name;
    double gpa;
    public Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }
    public void display() {
  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

public class SerializationExample {
    public static void main(String[] args) {
        Student student = new Student(11046, "Ankit Yadav", 7.8);
        String filename = "studentdetail";

        // Serialization
        try (ObjectOutputStream out = new
```

```
ObjectOutputStream(Files.newOutputStream(Paths.get(filename)))) {  
    out.writeObject(student);  
    System.out.println("Student object serialized successfully.");  
} catch (IOException e) {  
    System.out.println("IOException occurred: " + 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("IOException occurred: " + e.getMessage());  
} catch (ClassNotFoundException e) {  
    System.out.println("Class not found: " + e.getMessage());  
}  
}
```

#### 4. Output:



```
Run  SerializationExample x  
Student object serialized successfully.  
Deserialized Student Details:  
ID: 11046, Name: Ankit Yadav, GPA: 7.8  
Process finished with exit code 0  
java6thsem > studentdetail
```

#### 5. Learning Outcomes:

- Understanding Java object serialization and deserialization.
- Using ObjectOutputStream and ObjectInputStream for file handling.
- Handling exceptions in file operations.

## Problem-3

1. **Aim** Develop a menu-based Java application to manage employee records. Users can add employee details, display stored records, and exit the application.

2. **Objective:**

This program demonstrates file handling in Java by storing and retrieving employee records in a text file. It uses exception handling for robust error management and follows an interactive menu-driven approach.

3. **Implementation/Code:**

```
package myjava.exp5;
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;
    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 FILENAME = "employees";
```

```
private static Scanner scanner = new Scanner(System.in);

public static void addEmployee() {
    System.out.print("Enter Employee ID: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter an integer.");
        scanner.next();
    }
    int id = scanner.nextInt();
    scanner.nextLine();

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

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

    System.out.print("Enter Salary: ");
    while (!scanner.hasNextDouble()) {
        System.out.println("Invalid salary. Please enter a valid number.");
        scanner.next();
    }
    double salary = scanner.nextDouble();
    scanner.nextLine();

    Employee emp = new Employee(id, name, designation, salary);
    List<Employee> employees = loadEmployees();
    employees.add(emp);

    saveEmployees(employees);
    System.out.println("Employee added successfully.");
}

public static void displayEmployees() {
    List<Employee> employees = loadEmployees();
    if (employees.isEmpty()) {
```

```
        System.out.println("No employees found.");
        return;
    }

    System.out.println("\nEmployee Records:");
    for (Employee emp : employees) {
        emp.display();
    }
}

private static List<Employee> loadEmployees() {
    List<Employee> employees = new ArrayList<>();
    try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(FILENAME))) {
        employees = (List<Employee>) in.readObject();
    } catch (EOFException e) {
        // No employees stored yet
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Error reading employees: " + e.getMessage());
    }
    return employees;
}

private static void saveEmployees(List<Employee> employees) {
    try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILENAME))) {
        out.writeObject(employees);
    } catch (IOException e) {
        System.out.println("Error saving employees: " + e.getMessage());
    }
}

public static void main(String[] args) {
    while (true) {
        System.out.println("\n1. Add Employee");
        System.out.println("2. Display All Employees");
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
System.out.println("3. Exit");
System.out.print("Choose an option: ");

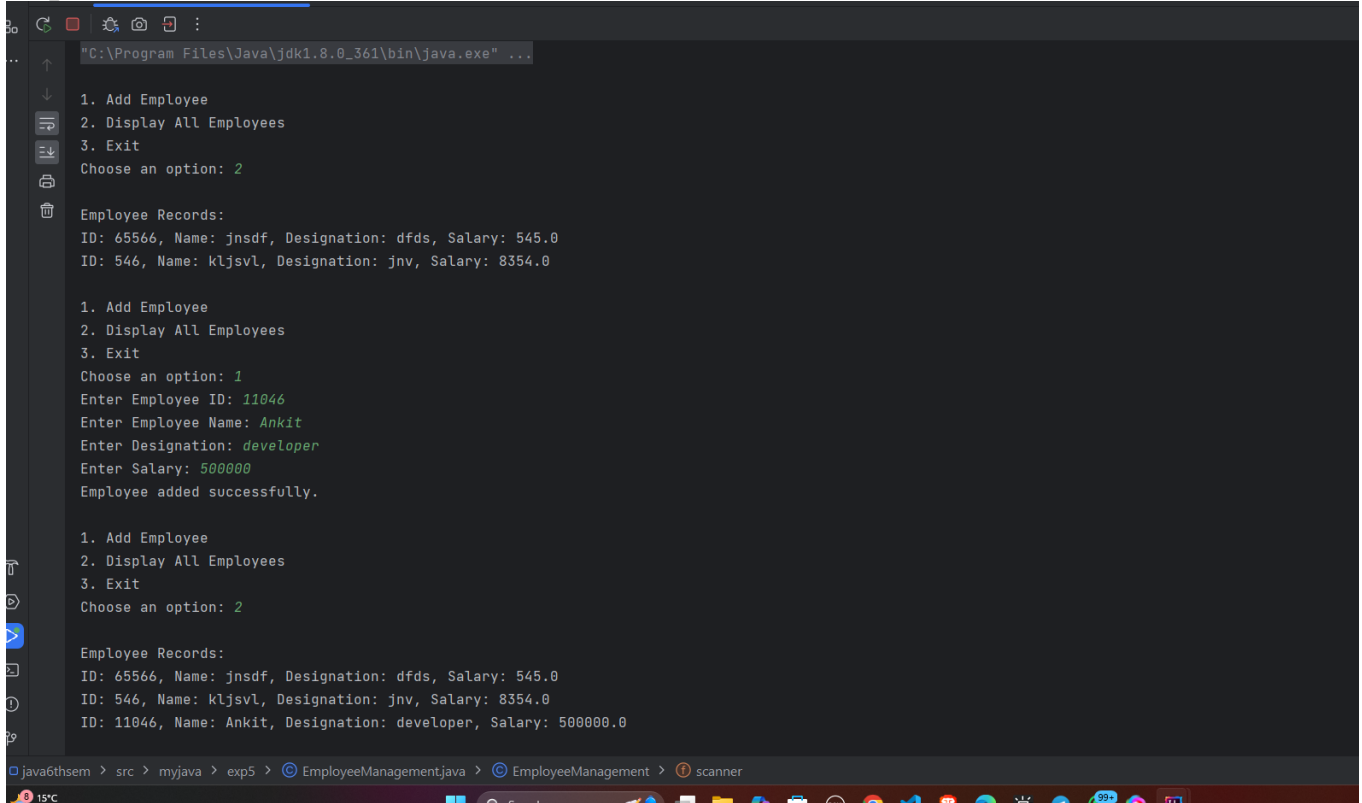
if (!scanner.hasNextInt()) {
    System.out.println("Invalid choice. Please enter a number.");
    scanner.next();
    continue;
}

int choice = scanner.nextInt();
scanner.nextLine();

switch (choice) {
    case 1:
        addEmployee();
        break;
    case 2:
        displayEmployees();
        break;
    case 3:
        System.out.println("Exiting application.");
        scanner.close();
        return;
    default:
        System.out.println("Invalid choice. Try again.");
}
}
}
```



## 4. Output:



```
"C:\Program Files\Java\jdk1.8.0_361\bin\java.exe" ...

1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 2

Employee Records:
ID: 65566, Name: jnsdf, Designation: dfds, Salary: 545.0
ID: 546, Name: kljsvl, Designation: jnv, Salary: 8354.0

1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID: 11046
Enter Employee Name: Ankit
Enter Designation: developer
Enter Salary: 500000
Employee added successfully.

1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 2

Employee Records:
ID: 65566, Name: jnsdf, Designation: dfds, Salary: 545.0
ID: 546, Name: kljsvl, Designation: jnv, Salary: 8354.0
ID: 11046, Name: Ankit, Designation: developer, Salary: 500000.0
```

## 5. Learning Outcomes:

- implementing a menu-driven application in Java.
- Storing and retrieving objects using serialization.
- Handling user input and file operations effectively.