



Experiment 5

Student Name: Jagadish Bhatta
Branch: CSE
Semester: 6th
Subject: Java

UID: 22BCS14017
Section: IOT-642 -B
DOP: 24/02/2025
Subject Code: 22CSH-359

Problem - 5.1

Aim: Writing a Java program to calculate the sum of a list of integers using autoboxing and unboxing, along with methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

Code:

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

public class SumCalculator
{
    public static Integer parseStringToInteger(String str)
    {
        try {
            return Integer.parseInt(str);
        } catch (NumberFormatException e)
        {
            System.out.println("Invalid number format: " + str);
            return null;
        }
    }

    public static int calculateSum(List<Integer> numbers)
    {
        int sum = 0;
        for (Integer num : numbers)
        {
            if (num != null) sum += num;
        }
        return sum;
    }

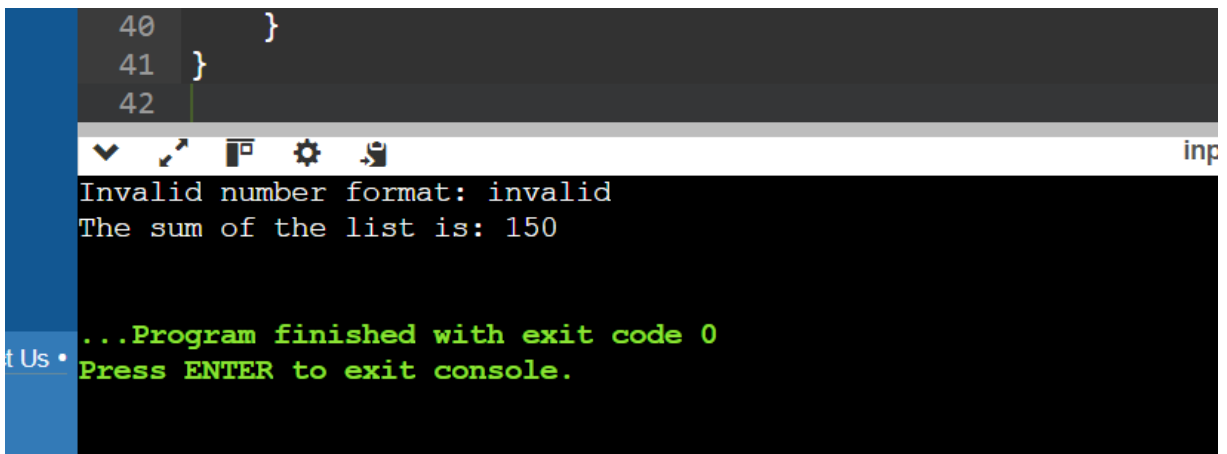
    public static void main(String[] args)
    {
        List<Integer> numbers = new ArrayList<>();

        numbers.add(10);
```

```
numbers.add(20);
numbers.add(30);
String[] inputStrings = {"40", "50", "invalid"};
for (String str : inputStrings)
{
    Integer num = parseStringToInteger(str);
    if (num != null) numbers.add(num);
}

System.out.println("The sum of the list is: " + calculateSum(numbers));
}
```

Output:



```
40 }
41 }
42
Invalid number format: invalid
The sum of the list is: 150
...Program finished with exit code 0
Press ENTER to exit console.
```

Test Cases:

Test Case 1:

Input: 10, 20, 30, "40", "50"

Expected Output: The sum of the list is: 150

Description: The list contains a mix of primitive integers and integers parsed from strings.

Test Case 2:

Input: "100", "200", "300"

Expected Output: The sum of the list is: 600

Description: All values are parsed from strings, and the sum is calculated.

Test Case 3:



Input: "50", "invalid", "70"

Expected Output:

Invalid number format: invalid

The sum of the list is: 120

Description: One of the inputs is not a valid integer, so it's skipped, and the sum of valid values is calculated.

Problem - 5.2

Aim : Java program that serializes and deserializes a Student object. It saves the Student object to a file and then reads it back, displaying the student details.

The program handles exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

Code :

```
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;

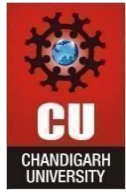
    }

    public int getId() {

        return id;

    }

    public String getName() {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        return name;
    }

    public double getGpa() {
        return gpa;
    }

    @Override
    public String toString() {
        return "Student ID: " + id + ", Name: " + name + ", GPA: " + gpa;
    }
}

public class StudentSerializationDemo {

    public static void serializeStudent(Student student, String filename) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(student);
            System.out.println("Student object has been serialized and saved to file.");
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
        } catch (IOException e) {
            System.out.println("Error: IO Exception occurred.");
        }
    }

    public static Student deserializeStudent(String filename) {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            Student student = (Student) ois.readObject();
            System.out.println("Student object has been deserialized.");
            return student;
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
        } catch (IOException e) {
            System.out.println("Error: IO Exception occurred.");
        } catch (ClassNotFoundException e) {
            System.out.println("Error: Class not found.");
        }
        return null;
    }

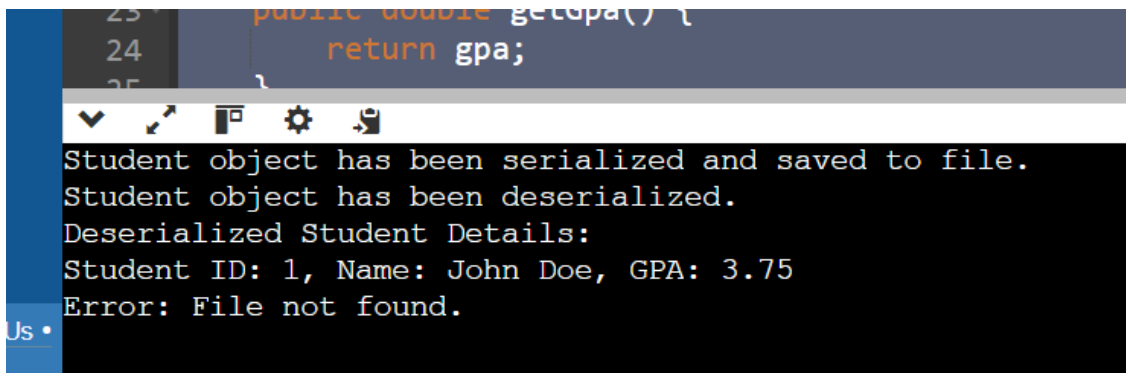
    public static void main(String[] args) {
        Student student = new Student(1, "John Doe", 3.75);
        String filename = "student.ser";

        serializeStudent(student, filename);

        Student deserializedStudent = deserializeStudent(filename);
        if (deserializedStudent != null) {
```

```
        System.out.println("Deserialized Student Details:");  
        System.out.println(deserializedStudent);  
    }  
  
    deserializeStudent("non_existent_file.ser");  
  
    }  
}
```

Output :



```
23 public double getGpa() {  
24     return gpa;  
25 }  
  
Student object has been serialized and saved to file.  
Student object has been deserialized.  
Deserialized Student Details:  
Student ID: 1, Name: John Doe, GPA: 3.75  
Error: File not found.
```

Test Cases:

Test Case 1: Serialize and Deserialize a valid student object.

Input: Student(1, "John Doe", 3.75)

Expected Output:

Student object has been serialized and saved to file.

Student object has been deserialized.

Deserialized Student Details:

Student ID: 1, Name: John Doe, GPA: 3.75

Test Case 2: Try to deserialize from a non-existent file.

Expected Output:

Error: File not found.

Test Case 3: Handle invalid class during deserialization.

Input: Manually modify the class file to simulate a ClassNotFoundException.

Expected Output:

Error: Class not found.