Lab 1 - Introduction to Classes, Objects, and Basic Control Structures

This lab introduces fundamental Java programming concepts through six progressive tasks. Students will learn basic object-oriented programming by creating classes and objects, practice user input handling and control structures, and work with strings and character processing. The exercises progress from building a name management system to implementing a number guessing game, creating a Roman numeral converter, developing a simple grade calculator, building a basic calculator with methods, and implementing a word counting program, establishing core programming skills essential for software development.

Contents

Memorize

Submission Deadline

Deadline to upload the solutions for all tasks is Saturday, 11:59 pm before the lab date.

General Information

The following tasks are to be worked on in fixed teams of two. Each team member must be able to explain all solutions. Please submit only one solution for each team of two. The submission must be a PDF file in our Moodle room with the name and matriculation number. Solutions must be in digital format with intermediate steps and detailed explanations (no handwritten scans). You can use any tool or drawing program of your choice to create the diagrams. If you have questions or need support, use the forum in our Moodle room and help each other.

1. Task 1: Name Management System

This program allows different people to change their name every three years. The program to be created should be able to run on different hardware platforms without recompilation. Therefore, create a Java program that makes this possible. Proceed as follows:

- First, create a class called Person that contains the following attributes: first_name, last_name, name_change_date (separated by day, month, and year).
- Now write a Java program that first outputs the Hamburg greeting for “Good day!” on the console.
- Then your program creates three objects of the Person class, named: `person_1`, `person_2`, and `person_3`.
- Then your program asks which person (1, 2, or 3) wants to change their name.
- Inputting 0 leads to program termination and input 4 displays the stored attributes of all objects. If the input is 1, 2, or 3, it asks for the new first and last name and the day, month, and year from when this change should be valid. This date must be checked regarding the three-year limit. If the check is successful, the change is made in the corresponding object and the result is displayed. If it is not successful, an error message is output and the program continues.
- The inputs in the individual input fields themselves do not need to be checked for correctness, i.e., letters, numbers, special characters, meaning the inputs in the individual input fields must be meaningful and do not need to be checked by your program. In a real project, this preprocessing would be checked by another program and is therefore not the subject of this task.

1.1. Preparation

First clarify the task by drawing the Person class according to UML notation and then creating a structure chart or flowchart that solves this task. Then define all necessary classes in Java.

1.2. Assistance

The following program, which demonstrates console input, can be used to solve this task:

```java
import java.util.Scanner;

public class InputExample {
    public static void main(String[] args) {
```

```
5       // Console input
6       Scanner scanner = new Scanner(System.in);
7
8       System.out.print("Enter something here: ");
9       String input = scanner.nextLine();
10      System.out.println("You entered \"" + input + "\"");
11
12      // If a number is needed for case distinction (e.g., age of majority):
13      System.out.print("Enter your age: ");
14      int age = scanner.nextInt();
15      if (age >= 18) {
16        System.out.println("Of age.");
17      } else {
18        System.out.println("Not yet of age.");
19      }
20
21      scanner.close();
22    }
23  }
```

2. Task 2: Number Guessing Game

Create a Java program that implements a number guessing game where the computer generates a random number and the player tries to guess it. This task will help you practice loops, conditionals, random number generation, and user input handling in Java.

The program should work as follows:

- Generate a random number between 1 and 100 (inclusive)
- Display a welcome message explaining the game rules
- Repeatedly ask the user to guess the number
- For each guess, provide feedback:
  - "Too small!" if the guess is lower than the target number
  - "Too big!" if the guess is higher than the target number
  - "You guessed it!" if the guess is correct
- Keep track of the number of attempts
- When the correct number is guessed, display a congratulatory message and the number of attempts it took
- The program ends when the correct number is guessed

2.1. Requirements

- Use the `Random` class for generating random numbers
- Use `Scanner` class for user input
- Use a loop structure to keep asking for guesses until correct
- Display clear and user-friendly messages
- Assume the user will always enter valid integers (no input validation needed)

2.2. Assistance

The following code snippets demonstrate key concepts needed for this task:

Random number generation:

```java
import java.util.Random;

Random rand = new Random();
int randomNumber = rand.nextInt(100) + 1; // generates 1-100
```

Simple user input with Scanner:

```java
import java.util.Scanner;

Scanner scanner = new Scanner(System.in);
System.out.print("Enter your guess: ");
int userGuess = scanner.nextInt();
```

Basic loop structure:

```java
boolean gameRunning = true;
while (gameRunning) {
    // Game logic here
```

```
4       if (userGuess == randomNumber) {
5           gameRunning = false; // End the game
6       }
7   }
```

3. Task 3: Roman Numeral Converter

Create a Java program that converts Roman numerals to integers. This task will help you practice working with strings, character processing, and conditional logic.

Roman numerals use seven different symbols with specific values:

- I = 1
- V = 5
- X = 10
- L = 50
- C = 100
- D = 500
- M = 1000

3.1. How Roman Numerals Work

Most of the time, Roman numerals are written from largest to smallest (left to right) and you simply add up the values:

- “III” = 1 + 1 + 1 = 3
- “XII” = 10 + 1 + 1 = 12
- “LVIII” = 50 + 5 + 1 + 1 + 1 = 58

However, there are special cases where a smaller numeral appears before a larger one, meaning you subtract instead of add:

- “IV” = 5 - 1 = 4 (not “IIII”)
- “IX” = 10 - 1 = 9
- “XL” = 50 - 10 = 40
- “XC” = 100 - 10 = 90
- “CD” = 500 - 100 = 400
- “CM” = 1000 - 100 = 900

3.2. Your Task

Write a program that:

1. Asks the user to enter a Roman numeral as a string
2. Converts it to an integer using the rules above
3. Displays the result

Test your program with these examples:

- “III” should give 3
- “IV” should give 4
- “IX” should give 9
- “LVIII” should give 58
- “MCMXCIV” should give 1994

3.3. Requirements

- Use `Scanner` for input
- Process the string character by character
- Handle both normal addition and subtraction cases
- Display clear output

3.4. Assistance

Getting individual characters from a string:

```java
String roman = "XIV";
char firstChar = roman.charAt(0);  // Gets 'X'
int length = roman.length();       // Gets 3
```

Simple approach - check current and next character:

```java
 String roman = "IV";
 int total = 0;

 for (int i = 0; i < roman.length(); i++) {
     char current = roman.charAt(i);

     // Check if we need to subtract (when current < next)
     if (i < roman.length() - 1) {
         char next = roman.charAt(i + 1);
         // Add your logic here to compare current and next
     }
 }
```

Getting the value of a Roman numeral character:

```java
// You can use if-else statements or a switch
int getValue(char c) {
    if (c == 'I') return 1;
    if (c == 'V') return 5;
    if (c == 'X') return 10;
    // ... continue for other characters
    return 0;
}
```

4. Task 4: Grade Manager

Create a Java program that calculates and manages student grades. This task will help you practice working with arrays, loops, mathematical operations, and basic data processing in Java.

Your program should implement a simple grade management system that:

- Stores grades for multiple students
- Calculates statistics like average, highest, and lowest grades
- Determines letter grades based on numerical scores
- Provides a summary report

4.1. Program Requirements

Your program should:

1. Ask the user how many students they want to enter grades for
2. Create an array to store the grades
3. Prompt the user to enter each student's grade (0-100)
4. Calculate and display:
    - The average grade
    - The highest grade
    - The lowest grade
    - How many students passed (grade >= 60)
    - How many students failed (grade < 60)
5. Display each student's numerical grade along with their letter grade

4.2. Letter Grade Scale

Use the following grading scale:

- A: 90-100
- B: 80-89
- C: 70-79
- D: 60-69
- F: 0-59

4.3. Requirements

- Use an array to store the grades
- Use loops to process the data
- Use the `Scanner` class for user input
- Display results in a clear, formatted way
- Assume the user will always enter valid numbers between 0-100

4.4. Assistance

Creating and working with arrays:

```java
// Create an array to hold grades
int numberOfStudents = 5;
double[] grades = new double[numberOfStudents];

// Store a grade in the array
grades[0] = 85.5;  // First student's grade
```

```
7
8 // Get a grade from the array
9 double firstGrade = grades[0];
```

Sample output format:

```
1  === Grade Report ===
2  Student 1: 85.5 (B)
3  Student 2: 92.0 (A)
4  Student 3: 78.5 (C)
5  Student 4: 67.0 (D)
6  Student 5: 88.5 (B)
7
8  === Statistics ===
9  Average Grade: 82.3
10 Highest Grade: 92.0
11 Lowest Grade: 67.0
12 Students Passed: 5
13 Students Failed: 0
```

5. Task 5: Simple Calculator with Methods

Create a Java program that implements a basic calculator using methods. This task will help you practice method creation, parameter passing, return values, and program organization in Java.

Your calculator should be able to perform basic arithmetic operations (addition, subtraction, multiplication, division) and be organized using separate methods for each operation.

5.1. Program Requirements

Your program should:

1. Display a menu of available operations to the user
2. Ask the user to select an operation
3. Prompt for two numbers
4. Perform the calculation using the appropriate method
5. Display the result
6. Allow the user to perform multiple calculations
7. Exit when the user chooses to quit

5.2. Menu Options

```
=== Simple Calculator ===
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1-5):
```

5.3. Requirements

- Create separate methods for each arithmetic operation
- Each method should take two parameters and return the result
- Handle division by zero appropriately
- Use a loop to allow multiple calculations
- Use the `Scanner` class for user input
- Display results in a clear format

5.4. Assistance

Sample program execution:

```
=== Simple Calculator ===
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1-5): 1
Enter first number: 15.5
Enter second number: 8.3
```

```
Result: 15.5 + 8.3 = 23.8

=== Simple Calculator ===
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1-5): 4
Enter first number: 10
Enter second number: 0
Error: Division by zero!
Result: 10.0 / 0.0 = 0.0

=== Simple Calculator ===
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1-5): 5
Thank you for using the calculator!
```

6. Task 6: Word Counter Program

Create a Java program that analyzes text input and counts various statistics about words and characters. This task will help you practice string manipulation, character analysis, and data processing in Java.

Your program should analyze a sentence or paragraph entered by the user and provide detailed statistics about the text.

6.1. Program Requirements

Your program should:

1. Ask the user to enter a sentence or paragraph
2. Count and display:
   - Total number of characters (including spaces)
   - Total number of characters (excluding spaces)
   - Total number of words
   - Total number of sentences (based on periods, exclamation marks, question marks)
   - Number of vowels (a, e, i, o, u - case insensitive)
   - Number of consonants
   - The longest word in the text
3. Display all results in a clear, formatted report

6.2. Requirements

- Use string methods for text processing
- Handle both uppercase and lowercase letters
- Consider punctuation appropriately
- Display results in a well-formatted report
- Use the `Scanner` class for user input

6.3. Assistance

Character analysis loop:

```java
String text = "Hello World!";
int vowelCount = 0;
int consonantCount = 0;
int totalChars = text.length();
int charsWithoutSpaces = 0;

for (int i = 0; i < text.length(); i++) {
    char c = text.charAt(i);

    if (Character.isLetter(c)) {
        charsWithoutSpaces++;
        if (isVowel(c)) {
            vowelCount++;
        } else {
            consonantCount++;
```

```
16          }
17      } else if (c != ' ') {
18          charsWithoutSpaces++; // Count non-space, non-letter characters
19      }
20  }
```

Sample program output:

```
1   Enter a sentence or paragraph to analyze:
2   Hello world! This is a great example of text analysis. How cool is that?
3
4   === TEXT ANALYSIS REPORT ===
5   Original text: "Hello world! This is a great example of text analysis. How cool
    is that?"
6
7   Character Statistics:
8   - Total characters (with spaces): 78
9   - Total characters (without spaces): 64
10  - Number of vowels: 24
11  - Number of consonants: 40
12
13  Word Statistics:
14  - Total number of words: 14
15  - Longest word: "analysis"
16
17  Sentence Statistics:
18  - Total number of sentences: 2
```

7. Lab Execution

If your program is not yet working without issue, we will try to correct this during the course of the lab. With good preparation, this should not be a problem. Every student is required to be able to explain their thought process at the beginning of the lab. By the end of the lab, the task needs to be completed. Of course, we will support you, but your personal commitment must also be clearly recognizable! Julian Moldenhauer, Furkan Yildirim, and Emily Antosch wish you lots of fun and success!