UNIVERSITY OF MISSOURI-COLUMBIA
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

3330 – OBJECT-ORIENTED PROGRAMMING

Laboratory 6: Loops

Due Date: Friday October 10th, 2025, at 5:00 pm

(No late submissions will be accepted for this Lab)

1 Goal

Java provides several kinds of loops for repeating a block of code while a certain condition is true. In this Lab, we will learn about three kinds of loops: while, do-while, and For-loop.

2 Resources

- Lecture notes "Unit 3"

3 Directed Lab Work

3.1 The while loop

In a **`while`** loop, the condition is tested first. The body is only executed if the condition is **`true`**, and then the process repeats.

The basic syntax of this loop is the following:

```
while (condition) {
    // do something in the body
}
```

The body of the loop may contain any valid Java statements, including conditional statements and even other loops (nested loops).

A loop becomes an infinite loop if the condition is always **`true`**. For example,

```
while (true) {
    // infinite loop
}
```

We will talk more about using infinite loops in later topics.

Example 1. The following loop prints consecutive integer numbers until reaching 5.

```
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

The output:

```
0
1
2
3
4
```

Example 2. The following code prints the English alphabet in a single line.

```
public class WhileDemo {
    public static void main(String[] args) {
        char letter = 'A';
        while (letter <= 'Z') {
            System.out.print(letter);
            letter++;
        }
    }
}
```

Here's what's happening: the program takes the first character, 'A', and then repeats:

if letter is less than or equal to 'Z', the program goes to the loop body; inside the body, it prints the current value of letter and moves on to the next.

The output:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

3.2 The do-while loop

In a **`do-while`** loop, the body is executed first and only then is the condition statement tested. Therefore, the body will be always executed at least once.

The basic syntax of the loop is the following:

```
do {
    // do something
} while (condition);
```

The program below keeps reading integer numbers from standard input and outputting them. If 0 is entered, the program outputs it and stops.

```
public class DoWhileDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int value;
        do {
            value = scanner.nextInt();
            System.out.println(value);
        } while (value > 0);
    }
}
```

The input:

```
1 2 4 0 3
```

The output:

```
1
2
4
0
```

Note that a do-while loop may become infinite, just like a while loop.

3.3 Task 1

- Use IntelliJ IDE to create a new Java project named Task1.
- Create a new Java class and name it **StopsAtEleven**.
- In this class write a **main** method that uses a while loop to accept, as input, a set of non-negative integers, that is either positive or zero, each written on a separate line. When the program encounters 11, it should stop and output the total length of the sequence so far (not including the final 11).
- The number 11 might be the first integer entered for the program input.
  - Sample Input:
    ```
    1
    3
    9
    11
    ```
  - Sample Output:
    ```
    3
    ```
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.4 Task 2

- Use IntelliJ IDE to create a new Java project named Task2.
- Create a new Java class and name it **SumAll**.
- In this class write a **main** method that asks the user for a positive nonzero integer value. The program should use a do-while loop to get the sum of all the integers from 1 up to the number entered. For example, if the user enters 10, the loop will find the sum of 1, 2, 3, 4, . . . , 10
- Sample Input:
  ```
  50
  ```
- Sample Output:
  ```
  1275
  ```
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.5 Task 3

- Use IntelliJ IDE to create a new Java project named Task3.
- Create a new Java class and name it **PrintSquares**.

- In this class write a **main** method that asks the user for a positive nonzero integer value N, print a sequence of all of the squares of natural numbers that do not exceed N, in ascending order.
- For example, if the user enters **16**, the loop will find the sequence **1, 4, 9, 16.**
- Note: the squares should not exceed N.
  - Sample Input:
    50
  - Sample Output:
    1, 4, 9, 16, 25, 36, 49.
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.6 The base for-loop syntax

Sometimes, we need to repeat a block of code a certain number of times. Java provides the for-loop that is very convenient for this purpose. This loop is often used to iterate over a range of values or an array. If the number of iteration or the range borders are known, it is recommended to use the for-loop. If they are unknown the while-loop may be a preferable solution. This loop has the following base syntax:

```
for (initialization; condition; increment/decrement) {
    // do something
}
```

The explanations:

- the initialization statement is executed once when the loop begins, usually, here loop variables are initialized.
- the condition determines the need for the next iteration; if it's false, the loop terminates.
- the increment/decrement expression is invoked after each iteration of the loop; usually, here loop variables are changed.

Inside the loop's body, the program can perform any correct java statements. It can even contain other loops. The order of execution for any for-loop is always the same:

1. the initialization statements
2. if the condition is false then terminate
3. the body
4. the increment/decrement
5. go to the stage 2 (condition).

Let's write a loop for printing integer numbers from 0 to 9 in the same line.

```
int n = 9;
for (int i = 0; i <= n; i++) {
    System.out.print(i + " ");
}
```

The code outputs:

```
0 1 2 3 4 5 6 7 8 9
```

The variable declared in the initialization expression is visible only inside the loop including the condition, the body and the increment/decrement expression. The integer loop's variables is often called i, j, k or index.

Another example. Let's calculate the sum of integer numbers from 1 to 10 (inclusive) using the for-loop.

```
int startIncl = 1, endExcl = 11;
int sum = 0;
for (int i = startIncl; i < endExcl; i++) {
    sum += i;
}
System.out.println(sum); // it prints "55"
```

Sometimes it's needed to declare a variable outside the loop. it's possible as well.

```
int i;
for (i = 10; i > 0; i--) {
    System.out.print(i + " ");
}
```

3.7 Infinite for-loop

The initialization statement, the condition, and the increment/decrement expression are optional, the for loop may not have one or all of them. Moreover, it's possible to write an infinite loop:

```
for (;;) {
    // do something
}
```

3.8 Nested loops

It's possible to nest one for-loop into another for-loop. For instance, the following code prints the multiplication table of numbers from 1 to 9 (inclusive).

```
for (int i = 1; i < 10; i++) {
    for (int j = 1; j < 10; j++) {
        System.out.print(i * j + "\t");
    }
    System.out.println();
}
```

It outputs:

```
1   2   3   4   5   6   7   8   9
2   4   6   8   10  12  14  16  18
3   6   9   12  15  18  21  24  27
```

```
4   8   12  16  20  24  28  32  36
5   10  15  20  25  30  35  40  45
6   12  18  24  30  36  42  48  54
7   14  21  28  35  42  49  56  63
8   16  24  32  40  48  56  64  72
9   18  27  36  45  54  63  72  81
```

3.9 Task 4

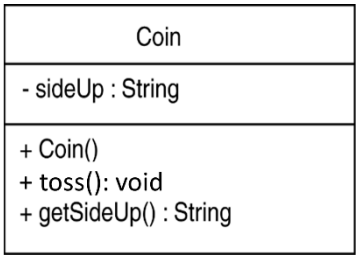
- Using IntelliJ IDE, create a new Java project called "**Task4**" that has a class named **SumA2B**.
- The main method of the **SumA2B** class should use for-loop to print the sum of all integers from a to b (a <= b), a and b are two user input values.
- Sample Input:
  ```
  52
  107
  ```
- Sample Output:
  ```
  4452
  ```
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.10 Task 5

- Using IntelliJ IDE, create a new Java project called "**Task5**" that has a class named **FizzBuzz**.
- The main method of the **FizzBuzz** class should be able to take the input of two integers: the beginning and the end of the interval (both numbers belong to the interval).
- The program should use for-loop to output the numbers from this interval, but if the number is divisible by 3, the program should output **Fizz** instead of number, if the number is divisible by 5, it will output **Buzz**, and if it is divisible both by 3 and by 5, it will output **FizzBuzz**.
- Output each number or the word on a separate line
  - Sample Input:
    ```
    10 18
    ```
  - Sample Output:
    ```
    Buzz
    11
    Fizz
    13
    14
    FizzBuzz
    16
    17
    Fizz
    ```
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.11 Task 6

- Using IntelliJ IDE, create a new Java project called "**Task6**" that has two classes, a **Coin** class and a **CoinTossSimulator** class.
- Implement the class **Coin** as shown in the following UML diagram below:



- Note that:
  - The sideUp field will hold either "heads" or "tails" indicating the side of the coin that is facing up.
  - The default constructor calls the **toss** method, described next
  - The **toss** method simulates the tossing of the coin. When the **toss** method is called, it generates a random number in the range of 0 through 1. If the random number is 0, then it sets the **sideUp** field to "heads". If the random number is 1, then it sets the **sideUp** field to "tails".
  - The **getSideUp** method returns the value of the sideUp field.
- Make sure that your Coin class does not show any syntax error.
- In the **CoinTossSimulator** class, write a program that demonstrates the Coin class. The program should create an instance of the class and display the side that is initially facing up. Then, use the for-loop to toss the coin 10 times. Each time the coin is tossed, display the side that is facing up. The program should keep count of the number of times a head is facing up and the number of times a tail is facing up and display those values after the loop finishes.
- Sample Output:

```
The side initially facing up: heads
Now I will toss the coin 10 times.
Toss:   tails
Toss:   tails
Toss:   heads
Toss:   tails
Toss:   tails
Toss:   tails
Toss:   tails
Toss:   heads
Toss:   heads
Toss:   heads
Total Heads: 4
Total Tails: 6
```

- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.12 Task 7

- In this task you will be writing the quiz part of this lab. Use the Tests & Quizzes tab on Canvas to complete and submit answer for this quiz.

- The quiz includes one programming question to evaluate your understanding of this lab material. You will be given **30 minutes** to complete and submit your answer, please follow the following important instructions.
  - You are to complete this quiz independently and alone. Collaborating, discussing, or sharing of information during the quiz is not permitted.
  - Use the IntelliJ Idea IDE to create a new Java project file and named it yourUMId_Lab6_Quiz. yourUMId is the first part of your UM (University of Missouri) email address before the @ sign.
  - **The quiz link will be inactive on October 10 at 5:00 PM. Note that, it is allowed to submit only once, so please review your code carefully before submitting your answer.**
  - To start, click Final Assessment link. Once you click "Begin Assessment," you will have 30 minutes to complete this quiz.
  - Read the question and start to write your codes using the opened project file in the IntelliJ IDE.
  - Once you finished writing your program, export your project into yourUMId_Lab6_Quiz.zip.
  - To submit your answers, click the Browse button, navigate the file explorer to your exported zip file, click Upload button, and then click Submit for Grading button.

4 Hand In

- Create a new folder and name it yourUMId_Lab6, then copy the folders of projects from Task1 to Task6 in your new folder yourUMId_Lab6. yourUMId is the first part of your UM (University of Missouri) email address before the @ sign Zip yourUMId_Lab6 folder into yourUMId_Lab6.zip.
- **Please note that the naming conventions, the types, and the submitted file structure are very important. The improper file structures, types or names will cause losing marks.**
- Submit your *yourUMId_* Lab6.zip file into Canvas on **October 10, before 5:00 PM**.
- **No submissions will be accepted after October 10, 5:00 PM.**