Lab 02: Structured Programming with Java

Objective(s):

1. Control Statements
2. Strings in JAVA
3. Arrays in JAVA
4. Java Math Class

1: Control Statements

Control structures are programming blocks that can change the path we take through those instructions.

There are three kinds of control structures:

- **Conditional Branches**, which we use for choosing between two or more paths. There are three types in Java: if/else/else if, ternary operator and switch.
- **Loops** that are used to iterate through multiple values/objects and repeatedly run specific code blocks. The basic loop types in Java are for, while and do while.
- **Branching Statements**, which are used to alter the flow of control in loops. There are two types in Java: break and continue.

If / else/ else if:

The *if/else* statement is the most basic of control structures, but can also be considered the very basis of decision making in programming.

While *if* can be used by itself, the most common use-scenario is choosing between two paths with *if/else*:

```
if (count > 2) {
    System.out.println("Count is higher than 2");
} else {
    System.out.println("Count is lower or equal than 2");
}
```

Theoretically, we can infinitely chain or nest if/else blocks but this will hurt code readability,

and that's why it's not advised.

Ternary Operator:

We can use a ternary operator as a shorthand expression that works like an *if/else* statement.

Syntax:

```
System.out.println(count > 2 ? "Count is higher than 2" : "Count is lower or equal than 2");
```

Switch:

If we have multiple cases to choose from, we can use a switch statement.

Syntax:

```
int count = 3;
switch (count) {
case 0:
    System.out.println("Count is equal to 0");
    break;
case 1:
    System.out.println("Count is equal to 1");
    break;
default:
    System.out.println("Count is either negative, or higher than 1");
    break;
}
```

Three or more if/else statements can be hard to read. As one of the possible workarounds, we can use switch, as seen above.

Loops:

We use loops when we need to repeat the same code multiple times in succession.

Syntax:

```
for (int i = 1; i <= 50; i++) {
    System.out.println("Hello World!");
}

int whileCounter = 1;
while (whileCounter <= 50) {
    System.out.println("Hello World!");
    whileCounter++;
}
```

Break:

We need to use break to exit early from a loop.

```
List<String> names = getNameList();
String name = "John Doe";
int index = 0;
for ( ; index < names.size(); index++) {
    if (names[index].equals(name)) {
        break;
    }
}
```

Here, we are looking for a name in a list of names, and we want to stop looking once we've found it.

A loop would normally go to completion, but we've used break here to short-circuit that and exit early.

Continue:

Simply put, continue means to skip the rest of the loop we're in:

Syntax:

```
List<String> names = getNameList();
String name = "John Doe";
String list = "";
for (int i = 0; i < names.size(); i++) {
    if (names[i].equals(name)) {
        continue;
    }
    list += names[i];
}
```

Here, we skip appending the duplicate names into the list.

As we've seen here, break and continue can be handy when iterating, though they can often be rewritten with return statements or other logic

2: Strings in JAVA

The **char** type represents only one character. To represent a string of characters, use the data type called **String**.

For example, the following code declares **message** to be a string with the value **"Welcome to Java"**.

```
String message = "Welcome to Java";
```

String is a predefined class in the Java library, just like the classes **System** and **Scanner**. The **String** type is not a primitive type. It is known as a *reference type*. Any Java class can be used as a reference type for a variable. The variable declared by a reference type is known as a reference

variable that references an object. Here, **message** is a reference variable that references a string object with contents **Welcome to Java**.

The `java.lang.String` class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.

Getting String Length

You can use the `length()` method to return the number of characters in a string. For example, the following code:

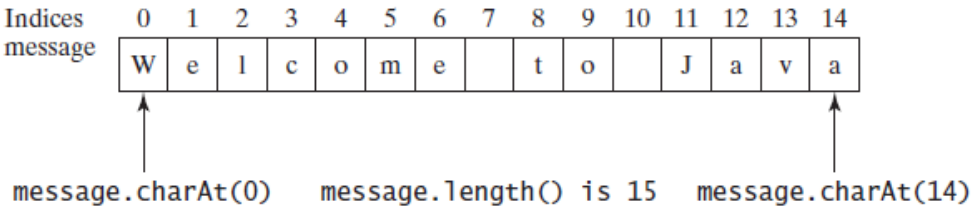
```
String message = "Welcome to Java";
System.out.println("The length of " + message + " is "+ message.length());
```

Getting Characters from a String

The `s.charAt(index)` method can be used to retrieve a specific character in a string **s**, where the index is between **0** and `s.length()`**–1**.

For example, `message.charAt(0)` returns the character **W**, as shown in figure.

Note that the index for the first character in the string is 0.



Converting Strings

The `toLowerCase()` method returns a new string with all lowercase letters and the `toUpperCase()` method returns a new string with all uppercase letters.

For example,

`"Welcome".toLowerCase()` returns a new string **welcome**.

`"Welcome".toUpperCase()` returns a new string **WELCOME**.

The `trim()` method returns a new string by eliminating whitespace characters from both ends of the string. The characters ' ', **\t**, **\f**, **\r**, or **\n** are known as *whitespace characters*.

3: Arrays in JAVA

An array stores a sequence of values that are all of the same type. We want not just to store values but also to be able to quickly access each individual value. The length of an array is established when the array is created. After creation, its length is fixed. Each item in an array is

called an *element*, and each element is accessed by its numerical *index*. The method that we use to refer to individual values in an array is to number and then *index* them—if we have *n* values, we think of them as being numbered from 0 to *n*−1.

Making an array in a Java program involves three distinct steps:

- Declare the array name.
- Create the array.
- Initialize the array values.

We refer to an array element by putting its index in square brackets after the array name.

To use an array in a program, you must declare a variable to reference the array and specify the array's *element type*.

Here is the syntax for declaring an array variable:

```
elementType[] arrayRefVar;
```

The **`elementType`** can be any data type, and all elements in the array will have the same data type.

Unlike declarations for primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array. It creates only a storage location for the reference to an array. If a variable does not contain a reference to an array, the value of the variable is **null**. You cannot assign elements to an array unless it has already been created. After an array variable is declared, you can create an array by using the **new** operator and assign its reference to the variable with the following **syntax**:

```
arrayRefVar = new elementType[arraySize];
```

Java has a shorthand notation, known as the *array initializer*, which combines the declaration, creation, and initialization of an array in one statement using the following **syntax**:

```
elementType[] arrayRefVar = {value0, value1, ..., valuek};
```

Arrays Class

Arrays class which is in `java.util.Arrays` package, is a provision by Java that provides you a number of methods through which arrays can be manipulated. This class also lets you perform sorting and searching operations on an array.

4: JAVA Math Class

The Java programming language supports basic arithmetic with its arithmetic operators: +, -, *, /, and %. The `Math` class provides methods and constants for doing more advanced mathematical computation.

The `Math` is located in the `java.lang` package, and not in the `java.math` package. Thus, the fully qualified class name of the Math class is `java.lang.Math`

The methods in the Math class are all static, so you call them directly from the class, like this:

```
Math.cos(angle);
```

Note: Using the static import language feature, you don't have to write Math in front of every math function: `import static java.lang.Math.*;`

This allows you to invoke the Math class methods by their simple names.

For example: `cos(angle);`

Constants

The `Math` class includes two constants:

- `Math.E`, which is the base of natural logarithms, and
- `Math.PI`, which is the ratio of the circumference of a circle to its diameter.

Basic Math Mehtods

The `Math` class includes more than 40 static methods. They can be categorized as *trigonometric methods*, *exponent methods*, and *service methods*. Service methods include the rounding, min, max, absolute, and random methods.

Trigonometric Methods

The `Math` class contains the following methods.

Method	*Description*
`sin(radians)`	Returns the trigonometric sine of an angle in radians.
`cos(radians)`	Returns the trigonometric cosine of an angle in radians.
`tan(radians)`	Returns the trigonometric tangent of an angle in radians.
`toRadians(degree)`	Returns the angle in radians for the angle in degree.
`toDegree(radians)`	Returns the angle in degrees for the angle in radians.
`asin(a)`	Returns the angle in radians for the inverse of sine.
`acos(a)`	Returns the angle in radians for the inverse of cosine.
`atan(a)`	Returns the angle in radians for the inverse of tangent.

The parameter for `sin`, `cos`, and `tan` is an angle in radians. The return value for `asin`, `acos`, and `atan` is a degree in radians in the range between -pi/2 and pi/2.

- One degree is equal to pi/180 in radians,
- 90 degrees is equal to pi/2 in radians
- 30 degrees is equal to pi/6 in radians.

Exponent Methods

There are five methods related to exponents in the `Math` class.

Method	*Description*
`exp(x)`	Returns e raised to power of x (e^x).
`log(x)`	Returns the natural logarithm of x ($\ln(x) = \log_e(x)$).
`log10(x)`	Returns the base 10 logarithm of x ($\log_{10}(x)$).
`pow(a, b)`	Returns a raised to the power of b (a^b).
`sqrt(x)`	Returns the square root of x ($\sqrt{x}$) for x >= 0.

The Rounding Methods

The `Math` class contains five rounding methods

Method	*Description*
`ceil(x)`	x is rounded up to its nearest integer. This integer is returned as a double value.
`floor(x)`	x is rounded down to its nearest integer. This integer is returned as a double value.
`rint(x)`	x is rounded up to its nearest integer. If x is equally close to two integers, the even one is returned as a double value.
`round(x)`	Returns (int)Math.floor(x + 0.5) if x is a float and returns (long)Math.floor(x + 0.5) if x is a double.

The Service Methods

The `min`, `max`, and `abs` Methods

The `min` and `max` methods return the minimum and maximum numbers of two numbers (`int`, `long`, `float`, or `double`).

For example, `max(4.4, 5.0)` returns `5.0`, and `min(3, 2)` returns **2**.

The `abs` method returns the absolute value of the number (`int`, `long`, `float`, or `double`).

This method generates a random `double` value greater than or equal to 0.0 and less than 1.0 (`0 <= Math.random() < 1.0`). You can use it to write a simple expression to generate random numbers in any range.

Lab Tasks:

Exercises

1. **Game:** How many fingers am I holding up? Write a program that prompts user to guess the number of fingers that the system is holding up. The program should then tell the user whether the guesses number was correct or not. (**Hint:** Use math functions)

2. **Wind-chill Temperature**

   How cold is it outside? The temperature alone is not enough to provide the answer. Other factors including wind speed, relative humidity, and sunshine play important roles in determining coldness outside.

   In 2001, the National Weather Service (NWS) implemented the new wind-chill temperature to measure the coldness using temperature and wind speed. The formula is

   $$t_{wc} = 35.74 + 0.6215t_a - 35.75v^{0.16} + 0.4275t_av^{0.16}$$

   where t_a is the outside temperature measured in degrees Fahrenheit and v is the speed measured in miles per hour. t_{wc} is the wind-chill temperature. The formula cannot be used for wind speeds below 2 mph or temperatures below -58 ^oF or above 41^oF.

   Write a program that prompts the user to enter a temperature between -58 ^oF and 41^oF and a wind speed greater than or equal to 2 and displays the wind-chill temperature. **(Hint:**Use Math.pow(a, b) to compute $v^{0.16}$.**)**

3. **Generate vehicle plate numbers**
   Assume a vehicle plate number consists of three uppercase letters followed by four digits. Write a program to generate a plate number.

4. **Beautifying the sentences**

   Write an application that takes a sentence from user and checks if the sentence stars from a capital letter and ends with a full stop. If it doesn't, the program should add it.

5. Given the following array, display its data graphically by plotting each numeric value as a bar of asterisks (*) as shown in the diagram.

   ```
   int[] array = {10, 19, 5, 1, 7, 14, 0, 7, 5};
   ```

```
Element  Value   Histogram
0        10      **********
1        19      ******************
2        5       *****
3        1       *
4        7       *******
5        14      **************
6        0
7        7       *******
8        5       *****
```

6. Write a Java program(with switch) that allows the user to choose the correct answer of a question.

   See the example below:

   What is the correct way to declare a variable to store an integer value in Java?

   a. int 1x=10;

   b. int x=10;

   c. float x=10.0f;

   d. string x="10";

   Enter your choice: c

7. Write a Java program to test if a given string contains the specified sequence of char values.
8. **Random Sentences**

   Write an application that uses random-number generation to create sentences. Use four arrays of strings called article, noun, verb and preposition. Create a sentence by selecting a word at random from each array in the following order: article, noun, verb, preposition, article and noun. As each word is picked, concatenate it to the previous words in the sentence.

   The words should be separated by spaces. When the final sentence is output, it should start with a capital letter and end with a period. The application should generate and display 20 sentences.

   The article array should contain the articles "the", "a", "one", "some" and "any"; the noun array should contain the nouns "boy", "girl", "dog", "town" and "car"; the verb array should contain the verbs "drove", "jumped", "ran", "walked" and "skipped"; the preposition array should contain the prepositions "to", "from", "over", "under" and "on".

9. **Game: heads or tails**

   Write a program that lets the user guess whether the flip of a coin results in heads or tails. The program randomly generates an integer 0 or 1, which represents head or tail. The program prompts the user to enter a guess and reports whether the guess is correct or incorrect.