Lab 02
Control Structures and Java Class Libraries

Objectives:

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

1: Control Structures

A program is a list of instructions or blocks of instructions. Java provides control structures that can change the path of execution and control the execution of instructions.

There are *three* kinds of control structures in java**:**

i. **Conditional Branches**, which are used for choosing between two or more paths. There are three types in Java: if/else/else if, ternary operator and switch.
ii. **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.
iii. **Branching Statements**, which are used to alter the flow of control in loops. There are two types in Java: “break” and “continue”.

Conditional Branches:

If / else/ else if:

If a certain condition is true, then if block will be executed otherwise else block will be executed.

Theoretically, we can infinitely chain or nest if/else blocks but this will hurt code readability that's why it's not advised.

Syntax:

```
class ExampleIfElseStatement
{
public static void main(String args[])
{
int age = 15;
```

```
if (age > 18) // if age is greater than 18
{
// It will be print if block condition is true.
System.out.println("The age of person is : " + age + "He/she is eligible for
voting");
}
else{

// It will be print if block condition is false.
System.out.println("He/she is not eligible for voting");
}
}
}
```

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:

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

Loops:

Loops are very useful when a programmer wants to execute a statement or block of statement multiple times until certain condition is true.

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++;
}

class ExampleDoWhileLoop
{
public static void main(String args[])
{
int i=1;
do
{
System.out.println(i);
i++;
}while(i <=10);
}
}
```

Branching Statements:

Break:

It is used to exit from a loop or switch statement. A loop would normally go to completion, but *break* would cause the early exit

Syntax:

```
class ExampleBreak
{
public static void main(String args[])
{
for(int i = 1 ; i <= 10 ; i++)
{
System.out.println(i);
if(i == 5)
break;
}
System.out.println("After the for loop");
}
}
```

Continue:

If the condition is true, then the *continue* statement skips the current iteration and transfer the control of the loop immediately to the next iteration

Syntax:

```
public class ContinueExample
{
public static void main(String args[])
{
for (int i = 1; i <= 5; i++)
{
if (i == 2)
{
```

```
continue;
}
System.out.println("Value of i ="+ i);
}
}
}
```

2: Strings in JAVA

Strings in Java are objects that are backed internally by a char. **String** is a predefined class in the Java library, just like the classes **System** and **Scanner**. The **String** type is non primitive type.

Syntax:

<String_Type> <string_variable> = "<sequence_of_string>";

Example:

String str = "oop";

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

Getting String Length

You can use the `length()` method to return the number of characters in a string.

Example:

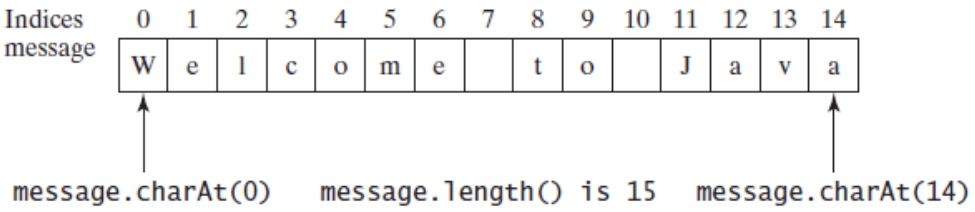
String message = "Welcome to Java";

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.



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.

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.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*.

Syntax:

```
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. Write a Java program to concatenate a given string to the end of another string
2. Write a program that computes your initials from your full name and displays them.
3. Write a Java program to replace each substring of below given sample string
    a. Sample string : "The quick brown fox jumps over the lazy dog."
    b. In the above string replace all the fox with cat
4. Write a program to print a table of all prime numbers less than 600, take the first 600 integers and cross out all those that are multiples of 2, 3, 5, etc. until only primes remain, then print out the table.
5. Write a program which solves quadratic equations of the form: ax2 + bx + c = 0. Values of a, b, c and x can be taken as input from user.

6. Write code that creates an array named odds and stores all odd numbers between 1 and 30 into it using a for loop.
7. Write a Java program to round up the result of integer division