CSE 215: Programming Language II Lab
Sec – 8, Faculty - MUO
Lab Officer: Tanzina Tazreen
Lab – 3
String and Math Class

Objective:
To learn different kinds String & Math class method

Java Strings:

Strings are used for storing text.

```
String str = "Hello";
```

String Concatenation

The '+' operator can be used between strings to combine them. This is called **concatenation**:

```
public class Main {
  public static void main(String args[]) {
    String firstName = "John";
    String lastName = "Doe";
    System.out.println(firstName + " " + lastName);
    // System.out.println(firstName.concat(lastName));
  }
}
```

```
John Doe
```

If you add a number and a string, the result will be a string concatenation:

```
public class Main {
  public static void main(String[] args) {
    String x = "10";
    int y = 20;
    String z = x + y;
    System.out.println(z);
  }
}
```

```
1020
```

Strings - Special Characters

Suppose we want to print this:

```
String txt = "We are the so-called "Vikings" from the north.";
```

To print quotation marks inside a string, we will need a special character: **backslash escape character (/).**

```
public class Main {
  public static void main(String[] args) {
    String txt = "We are the so-called \"Vikings\" from the north.";
    System.out.println(txt);
  }
}
```

```
We are the so-called "Vikings" from the north.
```

String Methods

A String in Java is actually an object, which contain methods that can perform certain operations on strings. The String class has a set of built-in methods that you can use on strings.

Method	Description	Return Type
charAt()	Returns the character at the specified index (position)	char
concat()	Appends a string to the end of another string	String
contains()	Checks whether a string contains a sequence of characters	boolean
equals()	Compares two strings. Returns true if the strings are equal, and false if not	boolean
equalsIgnoreCase()	Compares two strings, ignoring case considerations	boolean
indexOf()	Returns the position of the first found occurrence of specified characters in a string	int
isEmpty()	Checks whether a string is empty or not	boolean
length()	Returns the length of a specified string	int
replace()	Searches a string for a specified value, and returns a new string where the specified values are replaced	String
toLowerCase()	Converts a string to lower case letters	String
toUpperCase()	Converts a string to upper case letters	String

trim()	Removes whitespace from both ends of a string	String
substring(startIndex)	Returns a new string object from specified startIndex to end of the given string	String
substring(begIndex, endIndex)	returns a new String object containing the substring of the given string from specified startIndex to endIndex.	String

TABLE 4.10 The String class contains the methods for finding substrings.

Method	*Description*
`index(ch)`	Returns the index of the first occurrence of `ch` in the string. Returns `-1` if not matched.
`indexOf(ch, fromIndex)`	Returns the index of the first occurrence of `ch` after `fromIndex` in the string. Returns `-1` if not matched.
`indexOf(s)`	Returns the index of the first occurrence of string `s` in this string. Returns `-1` if not matched.
`indexOf(s, fromIndex)`	Returns the index of the first occurrence of string `s` in this string after `fromIndex`. Returns `-1` if not matched.
`lastIndexOf(ch)`	Returns the index of the last occurrence of `ch` in the string. Returns `-1` if not matched.
`lastIndexOf(ch, fromIndex)`	Returns the index of the last occurrence of `ch` before `fromIndex` in this string. Returns `-1` if not matched.
`lastIndexOf(s)`	Returns the index of the last occurrence of string `s`. Returns `-1` if not matched.
`lastIndexOf(s, fromIndex)`	Returns the index of the last occurrence of string `s` before `fromIndex`. Returns `-1` if not matched.

Let's see some Examples:

```
public class Main {
  public static void main(String[] args) {
    String txt = "Hello World";
    System.out.println("The length of the txt string is: " +
txt.length());
    System.out.println(txt.toUpperCase());
  }
}
```

```
The length of the txt string is: 11
HELLO WORLD
```

Java Math Methods

The Java Math class has many methods that allows you to perform mathematical tasks on numbers.

Method	**Description**	**Return Type**
abs(x)	Returns the absolute value of x	double|float|int|long
cbrt(x)	Returns the cube root of x	double

ceil(x)	Returns the value of x rounded up to its nearest integer	double
cos(x)	Returns the cosine of x (x is in radians)	double
exp(x)	Returns the value of E^x	double
floor(x)	Returns the value of x rounded down to its nearest integer	double
max(x, y)	Returns the number with the highest value	double|float|int|long
min(x, y)	Returns the number with the lowest value	double|float|int|long
pow(x, y)	Returns the value of x to the power of y	double
random()	Returns a random number between 0 and 1	double
round(x)	Returns the value of x rounded to its nearest integer	int
sqrt(x)	Returns the square root of x	double

Let's see an Examples:

```
public class Practice {
    public static void main(String[] args) {
        int t = -3;
        System.out.println(Math.abs(t));
    }
}
```

Task:

1) Compute the area of a hexagon using the following formula:

$$Area = \frac{6 \times s^2}{4 \times \tan\left(\frac{\pi}{6}\right)}$$

Here, s is the side of a hexagon.

➢ Write a program that prompts the user to enter the side of a hexagon and display its area. If the side length is invalid (<0), display a message showing "The area can not be computed due to invalid side length". Below, a sample run is given:

```
Enter the side: 5.5 Enter
The area of the hexagon is 78.59
```

2) Given a string s1 and another string s2, check if string s1 starts or ends with string s2.

For example, for s1 = “apple” and s2=”app” , display the following output:
“The string “apple” starts with “app”.

for s1 = “apple” and s2=”ple” , display the following output:
“The string “apple” ends with “ple”.

for s1 = “apple” and s2=”ppl” , display the following output:
“The string “apple” does not start or end with “ppl”.

3) Given a string, find the index of the second last occurrence of any vowel. For example, if your string is “corresponding” , then return the index of ‘o’ which is the second last vowel (after ‘i’).

4) * (Take Home Task) Modify task 2 to check if two strings s1 and s2 are equal, or equal after ignoring cases, or if s1 starts or ends with s2 (and vice versa), or if s2 is substring of s2 (or vice versa). If none of these happen, display an appropriate message.

5.
(a) Generate two random numbers, lower and upper, within the range 1-1000. Random numbers in Java can be generated by using the method Math.random().
e.g. To generate a random integer between min and max range:

```
int randNumber = (int)(min + Math.random() * (max – min + 1));
```

N.B. The (int) on the right hand side of the assignment operator is called type-casting.

(b) Write a program which will use while loop to print all the integers between lower and upper which are divisible by 5 or 8 in descending order to the console **in one line**.