UNIT 3. STRING AND MATH CLASSES

Part I: classes from the standard library

1. The String class

String is not a Java primitive data type (like int, double, etc). The String type is a class. Therefore, we can instantiate String type objects.
In Java String objects are given a special treatment, because they are frequently used in programs.

- The sentence **String greeting = "Hello";**
  automatically generates an object of the class String
  This is the same as doing: **String greeting = new String("Hello");**

- Java allows us to concatenate Strings using the special **+** operator
  System.out.println("Good morning," + greeting + "!");

- If we concatenate anything to a String, it will automatically be converted to String and then the concatenation will take place.
  In the next example:
  int value = 5;
  System.out.println("Value = " + value);
  value is first converted to the expression "5" and then concatenated to "Value="

What's the output of the next statement?

```
System.out.println ("Result:" + 2 + 2);
```

How could we first perform an arithmetic "+" ?

We can create an empty String in 2 ways:

```
String str="";
String str=new String();
```

Some existing methods of the String class

int **length**()

Returns the length of the string

boolean **equals** (String str2)

Compares the content of the string with the content of another string object (passed as a parameter). It returns true is they are equal, false if they are different.
*Remember that Java is case sensitive

1) String **substring** (int start, int end)
2) String **substring** (int start)

1) Returns a new string which is a fragment of the string that invokes the method.
This new string contains the characters from *start* position up to, but not including, *end* position.
The positions are interpreted as indexes that start from 0
2) If it is given just one argument, it returns the substring from the given *start* index to the end

int **indexOf** (char car)
int **indexOf**(String str)

Returns the index of the first found occurrence of the parameter in the string. It returns -1 if the value to search for never occurs.

int **compareTo** (String str)

Returns an int as a result of a string comparison. It returns 0 if they are equal, a negative value if the parameter string (*str*) is greater than the string, or a positive value otherwise.

static String **valueOf** (primitive_type x)

Static method that returns the String representation of the primitive data passed as a parameter(int, long, float, double, etc)

String **trim** ()

Returns a copy of the string stripped of whitespace from both ends

char **charAt** (int pos)

Returns the character at the specified position (considered as an index)

String **toLowerCase** ()
String **toUpperCase** ()

toLowerCase returns the lowercase equivalent of the string
toUpperCase returns the uppercase equivalent of the string

Other methods: **equalsIgnoreCase**, **lastIndexOf**, **replace**, **split**, **startsWith**, **endsWith**, **replace**, etc

Now it's time to remark that all Java classes have the method toString( ) (inherited from the java.lang.Object class). This method is used when we need a String representation of an object.

Example 1:

```
String s= "Hello";
String s2=s.substring(0,2);          //s2="He"
char c=s2.charAt(1);                 //c='e'
if(s2.equals("HE")) ...;             //false
if(s2.equalsIgnoreCase("HE") ...;    //true
s= "john".toUpper();                 //s= "JOHN"
int i=s.indexOf("JO");               //i=0
```

Example 2:

```
String str = "  hello how are you  "
System.out.println (str.length());       //19
String strBis = str.trim();
System.out.println (strBis.length());   //17
```

Example 3:

```
float f = 3.141592;
String PI = String.valueOf( f );          //PI="3.141592"
```

Example 4:

```
String str="Thomas";
int result=str.compareTo("Albert");  //result is greater than 0
```

2. The Math class

The Math class (from the java.lang package) provides static constants and methods for performing common mathematical calculations (roots, logarithms, trigonometric and exponential functions, etc).

Some static methods of the Math class

double Math.**abs**(double x)	Absolute value or modulus of a real number
double Math.**sin**(double x) double Math.**cos**(double x) double Math.**tan**(double x) double Math.**asin**(double x) double Math.**acos**(double x) ……	Sine, cosine, tangent, etc… of an angle that must be given in radians
double Math.**exp**(double x)	Exponential function (which takes *e* to the power of whatever is in the parentheses) (Math.exp(1) is **e^1=2,71**)
double Math.**log**(double x)	Natural logarithm (logarithm with base e)
double Math.**sqrt**(double x)	Returns square root of the parameter
long Math.**round**(double x) int Math.**round**(float x)	Obtains the result of rounding the argument to the nearest integer (3.5 is rounded to 3)
double Math.**ceil**(double x) double Math.**floor**(double x)	Returns the ceiling of x as a double (the smallest integer value greater than or equal to x) Returns the floor of x as a double
double Math.**pow**(double a, double b)	Power: returns *a* raised to the power of *b*
double Math.**random**()	Returns a random positive double value, greater than or equal to 0.0 and less than 1.0.
Math.**max**(a,b) Math.**min**(a,b)	Returns the greater of 2 values Returns the smaller of 2 values
Math.**E**	Constant with the value of *e* (Euler's number)
Math.**PI**	Constant with the value of π

Examples:

```
System.out.println(Math.ceil(2.3));          // shows 3.0
System.out.println(Math.floor(-1.87);  // shows –2.0

 z = Math.pow( 2,4 );                  // z=16

num = Math.random ()* 100;             // num is a random number between 0 and
                                       // 100, 100 not included

num = Math.random()*100 + 1            // random number between 1 y 101,
                                       // 101 not included

x = Math.PI;                           // x=3.14159…..

z= Math.sin(270*Math.PI/180);          // z = -1 (sine of 270 degrees)
z = Math.sqrt(Math.round(25.8));       // z= 5 (rounded square root of 25.8)
```