Testat 2

Programming in Java

SS 2020

Learning objectives:

- **You perform research in the Java Documentation successfully**
- **You can program method calls**
- **You create a more sophisticated program**

Task 1: Preparation

Create a second project in NetBeans, regarding **BIS2151_AttestationRules_SS2020.pdf** and the **syllabus**! Create a subfolder called **Docu** inside the created project folder. Answer the following questions in a plain text file: Create a PDF file **Testat-2-Theory.pdf** from it. Place this file in the newly created **Docu** folder!

Task 2: Search in JavaDoc

Research the Java documentation and answer the following questions:

Look on class **Math**
1. What does the method **pow(double a, double b)** do?
2. What is the name of the method that returns you the absolute value of a number?
3. Specify Pi as it is stored in Java. (Simply use Copy/Paste from the Java documentation)

Look on class **String**
4. What can be done with the **split(String regex)** method?
5. Which method can be used to lowercase all letters of a string?

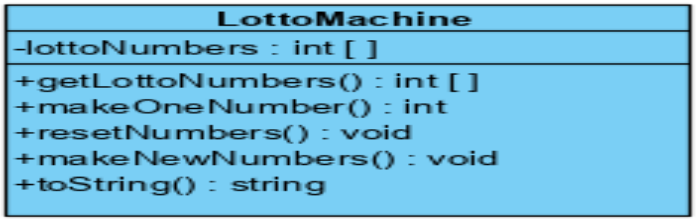
Look on class **BigInteger**
6. Which method can we use to check whether the number is likely to be a prime number?

Look on class **Thread**
7. Which method do you use to let your program sleep for a certain number of milliseconds?

General Java research
8. From which class all classes in Java inherit or which class is the mother of all classes in Java?

Task 3: Programming a small Lotto machine

You now program a class **`LottoMachine`**



You also program a second class **`LottoMachineTester`** that tests **`LottoMachine`**

LottoMachine has a private Array of 6 int numbers. These are the 6 numbers that a gamer would like to play. They are called lottoNumbers[0] to lottoNumbers[5]
(We always count from 0 beginning in Java!!)

The class has a few methods to do its job:

getLottoNumbers() returns the array lottoNumbers to the caller of the method.

resetNumbers() sets all 6 numbers in lottoNumbers back to 0

makeOneNumber() creates one random number between 1…49 and puts it into array lottoNumbers:
It searches for the next free place (where the number is just 0) and saves the new number there, if this numbers is not already existing in one of the 5 other places.

makeNewNumbers() repeats calling makeOneNumber, until the whole array is filled with 6 different numbers from 1 to 49.

toString() generates and returns to the caller a string with the 6 numbers in ascending order.
Let us assume that we have these numbers:
lottoNumbers[0] is 43
lottoNumbers[1] is 11
lottoNumbers[2] is 31
lottoNumbers[3] is 13
lottoNumbers[4] is 0 (not set)
lottoNumbers[5] is 0 (not set)

In this situation toString() is programmed so that the following is produced:
If you would program in method main() of LottoMachineTester (see below) the code:
`System.out.printl("Actual Lotto numbers are: "+myLotto);`

This would generate the followings output on screen:
`Actual Lotto numbers are: 11 13 31 43`

In class LottoMachineTester you only program the main method that creates one object of class LottoMachine: **`LottoMachine myLotto = new LottoMachine();`**

Then it calls makeNewNumbers() of the object: **`myLotto.makeNewNumbers();`**
And finally it prints (see above) the objects number values. You should always see 6 numbers because you call **`makeNewNumbers`** () that generates all 6 numbers.
Then you reset the array **`myLotto.resetNumbers();`**
and repeat the test by calling again makeNewNumbers() and print it out again.