Exercises module 7 – Arrays and loops

1 – Sum, Max and Min

1. Write a program that outputs the **sum**, **maximum** and **minimum** value of this array using for loop:

```
int[] input = new int[]{1, 4, 5, 7,20000, -511, 100, -200, 400};
```

2. The **for-each** loop is another loop construct that is useful to iterate over all the items in an array or collection of information.

   It looks like: `for (int item : input) {  …  }`

   Do lookup on the web how it works and then modify the previous exercise so that it uses for-each construct instead.

2 – Loops and arrays

1. You are given this code below and you have to implement the logic to find all the *even* numbers in the **input array** and put them into the **result array.**

```
int[] input = new int[]{1, 4, 5, 7,20000, -511, 100, -201, 400};
int[] result;

// implement code here to put the even numbers into result

for (int item : result) {
    System.out.println(item);
}
```

Think of:

- How big should the result array be?
- The result array should only be as big as the number of results

1. Write a new method named **binaryStringToInteger(string input)** that returns an integer

3 – Binary to integer converter

In this exercise, we will write a simple binary to integer converter

2. Create a **new project** in IntelliJ
3. Write a new method named **binaryStringToInteger(string input)** that returns an integer

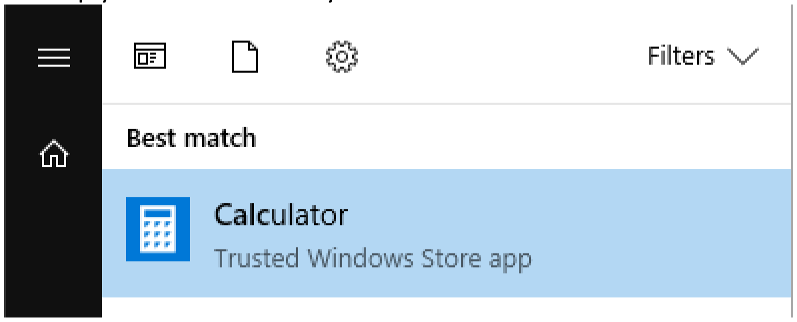
   When calling the method, the following result should be returned:

```
System.out.println(binaryStringToInteger("10000000"));  //128
System.out.println(binaryStringToInteger("00000001"));  //1
System.out.println(binaryStringToInteger("00001111"));  //15
System.out.println(binaryStringToInteger("11110000"));  //240
System.out.println(binaryStringToInteger("01111111"));  //127
System.out.println(binaryStringToInteger("11111111"));  //255
```

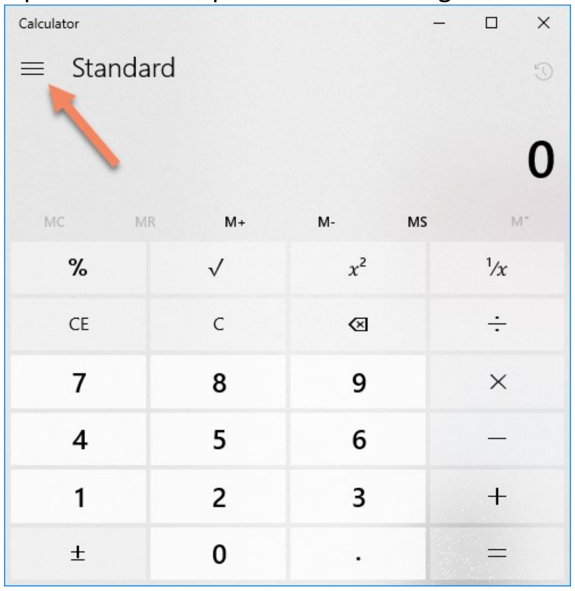
   (We only accept input strings that contains exactly 8 numbers)

4. If you are not familiar with binary math, then this video is a good introduction:
   Introduction to number systems and binary
   https://tinyurl.com/gnmxhm3

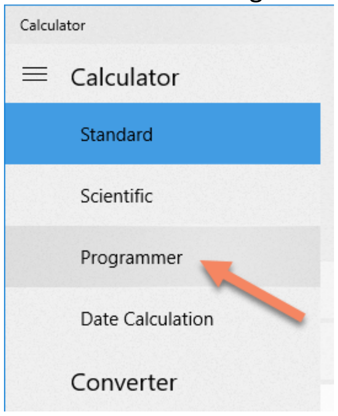
5. To help you with the math you can use the built in Calculator in Windows



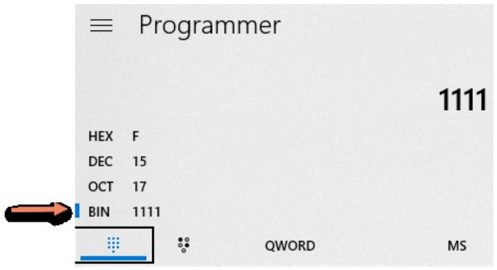
Open it and then press the "Hamburger" menu



Then choose the Programmer mode:



In the programming mode, you can select BIN and enter for example 1111 and you see the decimal value of 15.



Now let's try to implement our own binary to integer converter.

6. In the method create an **array** named **mapping** that can hold **8 integers**
7. Write a **for-loop** that fills the array with the following values:

Position	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Value	1	2	4	8	16	32	64	128

The array will help us with the conversion in the next step.

8. Now we need to write the logic that will iterate over the input string and for each **one('1')** found in the string will look-up the corresponding value in the mapping array and summarize the result.

   As an example, if we have the input of "**1100000001**", we find **128 + 64 + 1 = 193**

   The rightmost one is always one.

4 – Chess-board

1. Our job is to write the code to generate a chess board pattern on the screen, something similar to:

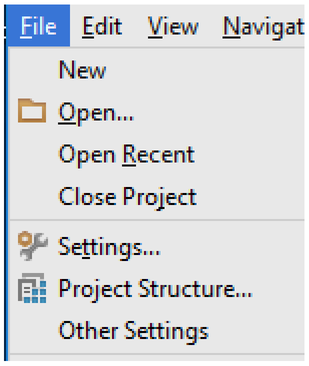
```
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
```

   The template for the code looks like this:

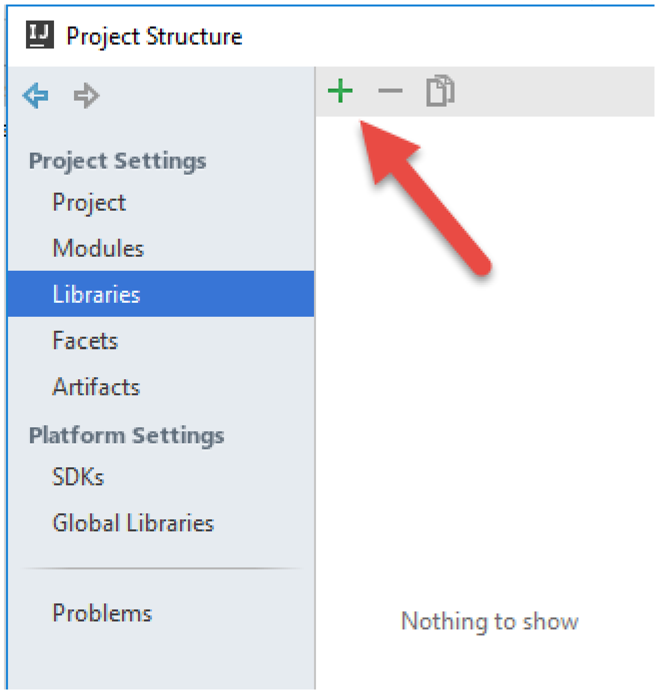
```
boolean [][] board = new boolean[8][8];

//Put logic here to fill the board array with the chess pattern

//Put logic here to output the board to the screen
```

5 – A nicer chess-board

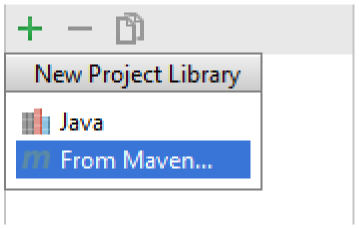
1. This time we are going to modify the previous code to display the chess board on a real console window instead of the built in console in IntelliJ.
2. Click on ***File -> Project Structure***



3. Click on Libraries and then on the + icon at the top



4. Select "From Maven"



5. Search for "**`com.googlecode.lanterna`**" and press the arrow own and choose version **2.1.9**

   then press OK and go back to the code.

   One of the features of Maven is a way to download dependencies and libraries from central repositories like: https://search.maven.org/

6. Add the following import statements at the top of your code:

```
package com.company;

import com.googlecode.lanterna.*;
import com.googlecode.lanterna.input.Key;
import com.googlecode.lanterna.terminal.Terminal;
import com.googlecode.lanterna.terminal.TerminalSize;

import java.nio.charset.Charset;

public class Main {
```

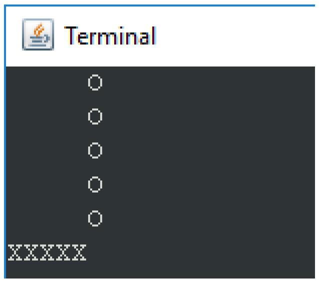
7. At the top of the main method, add these two lines:

```
Terminal terminal = TerminalFacade.createTerminal(System.in, System.out,
Charset.forName("UTF8"));

terminal.enterPrivateMode();
```

8. To write to the console window you can for example write:

```
for (int i=0;i<5;i++)
{
    terminal.moveCursor(5,i);
    terminal.putCharacter('O');
    terminal.moveCursor(i,5);
    terminal.putCharacter('X');
}
```

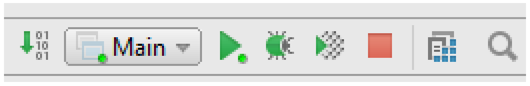
   This will result in the following pattern on the new window that opens:



```
     O
     O
     O
     O
     O
XXXXX
```

   (Make sure you understand why the output is like it is)

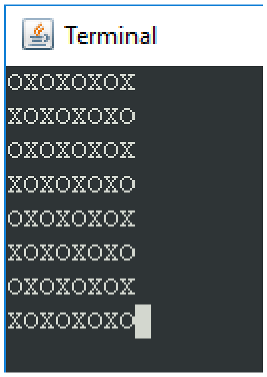
9. To close the window, you need to press the red **stop-button** in IntelliJ.



10. The reason for using this library is that we now can control the cursor and the output much better than we can in the console output inside Intellij.

    We will use this library for some of the exercises in this course.

11. Now modify your chess board code so that it outputs the chess board to this new console window instead.



```
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
```

12. To add colors, use the following:

```
terminal.applyBackgroundColor(Terminal.Color.GREEN);
terminal.applyForegroundColor(Terminal.Color.BLUE);
```

    Try it!

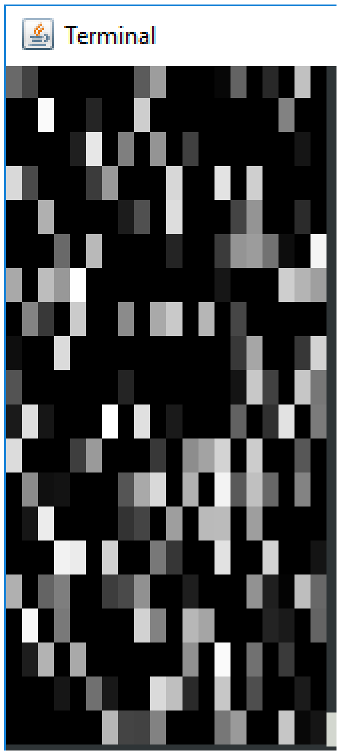
13. You can also output Unicode characters using code like:

```
terminal.putCharacter('\u263a');
```

    This one should output a smiling face: ☺

6– Effects – part 1

1. The goal of this step is to write a program that can generate this random pattern on the screen:



2. Complete the following code to get the output above:

```
package com.company;

import com.googlecode.lanterna.*;
import com.googlecode.lanterna.terminal.Terminal;
import java.nio.charset.Charset;
import java.util.Random;

public class Main {

    public static void main(String[] args) {

        Terminal terminal = TerminalFacade.createTerminal(System.in,
System.out, Charset.forName("UTF8"));
        terminal.enterPrivateMode();

        int  [][] board = new int[20][20];

        //Populate the array at random positions with
        //random values between 0-255
        //Try fill it with 100 random values

        Random rand = new Random();

        //Draw the array on the screen and each position gets
        //its color from the array

        //Draw the board on the screen
    }
}
```

3. To write a color block to the screen you can use the following piece of code:

```
terminal.moveCursor(x, y);
terminal.applyForegroundColor(color, color, color); //red, green, blue
terminal.putCharacter('\u2588');
```

Setting the three values to be the same, a value between 0-255, will give you a color range from 0 (black) to 255 (white).

You can experiment with different value combinations, like:

```
terminal.applyForegroundColor(color, 0 , 0); //red
terminal.applyForegroundColor(0, color , 0); //blue
terminal.applyForegroundColor(0, 0, color); //green
```

(All values must be between 0-255)

If you want to know more about what RGB means and how it works, then search for RGB on Wikipedia or on the web. Your computer display uses these RGB values to generate 16 million different colors.

6 – Effects – part 2

1. Let’s continue the previous exercises with adding some animations.

   At the top of your code, make sure you import the following packages and also add “**throws InterruptedException**” after the main method.

```
package com.company;

import com.googlecode.lanterna.*;
import com.googlecode.lanterna.input.Key;
import com.googlecode.lanterna.terminal.Terminal;
import java.nio.charset.Charset;
import java.util.Random;

public class Main {

    public static void main(String[] args) throws InterruptedException {
```

2. In your application add an infinite loop (a loop that never ends), like this one and insert your code to draw the screen:

```
while(true) {

    //Add one random color to the board
    int xx = rand.nextInt(20);
    int yy = rand.nextInt(20);
    board[xx][yy]= 255; // rand.nextInt(255);

    //TODO: Insert you’re your code to draw on the screen here


    //Formula to take the average value of the current cell
    //and all of its neighbors
    int  [][] tmpboard = new int[20][20];
    for (int y = 1; y < 19; y++) {
        for (int x = 1; x < 19; x++) {
            int color = 0;
            color += board[x - 1][y - 1];
            color += board[x    ][y - 1];
            color += board[x + 1][y - 1];

            color += board[x - 1][y];
            color += board[x    ][y];
            color += board[x + 1][y];

            color += board[x - 1][y + 1];
            color += board[x    ][y + 1];
            color += board[x + 1][y + 1];
```

```
                if(color>0)
                    color /= 9;

                tmpboard[x][y] = color;
            }
        }
        board = tmpboard;

        //Wait for a key to be pressed
        Key key;
        do{
            Thread.sleep(5);
            key =terminal.readInput();
        }
        while(key == null);
    }
```

Make sure you understand how the code works and then run it! Press one key will animate the screen.

If you have time feel free to play with the code, to create different effects and colors! Perhaps create fireworks like effects!

3. **Challenge - stretch task if you have time**
   The current calculation logic above does not update all the cells in the array (only cells (1,1)-(19,19) is updated. Try to update it so that the calculation loop updates all the cells in the array:

```
        //Formula to take the average value of the current cell
        //and all of its neighbors
        int  [][] tmpboard = new int[20][20];
        for (int y = 0; y < 20; y++) {
            for (int x = 0; x < 20; x++) {
```

7 – Sum of numbers

In this exercise, your task is to write an application which calculates the sum of a series of numbers, 1…n, where n is an input from the user. If the input is 5, the sum is 1+2+3+4+5 = 15.
A test run might look like:

```
Enter a number: 5
Answer: 15
```

Main Task

1. Start by creating a new **Console project**.
2. In the **main** method, add the necessary code. Here, you'll want to use some kind of loop. You can make it work with different types of loops, and the choice is entirely up to you.
3. Continue by making the program a little more verbose. Try printing the entire calculation so that the output looks like the following:

```
Enter a number: 5
1+2+3+4+5 = 15
```

Stretch Task

1. Let the user enter the number to start counting from, so that the user can count from m…n, so if the inputs are 2 and 5, the sum is 2+3+4+5 = 14.
2. Add code to your method that checks if the second number is larger than the first. Show an error message to the user if it's not.

Questions and concepts to study further on your own:

- Can you resize an array in Java?
- Jagged arrays
- How does the Java 'for each' loop work?
  https://stackoverflow.com/questions/85190
- How to initialize an array in Java?
  https://stackoverflow.com/questions/1938101/how-to-initialize-an-array-in-java
- What happens if you try to access elements outside the array?
- Binary calculations and math
- Hamburger menu:
  - https://en.wikipedia.org/wiki/Hamburger_button
  - https://apptimize.com/blog/2015/07/the-ultimate-guide-to-the-hamburger-menu/