Exercises module 3 - Our-first-application

1 – Line breaks

1. Run this code:

```
System.out.print("Line 1");
System.out.print("Line 2");
System.out.print("Line 3");
```

   As you see the output becomes:

```
Line 1Line 2Line 3
```

2. How do we get the output to appear on three separate rows?

   Fix the code so each output appears on separate rows using **println**.

3. An alternative way to get each item on separate rows is to use the **printf** method instead. Update the code above and use **printf** to get the output on separate lines.

   When we want to output a new line, we can use the escape characters **\r\n** to output a new line, but this will not look good on a Linux machine, why?

   Read about the issue here and about the difference of **\r\n** and **\n**
   https://blog.codinghorror.com/the-great-newline-schism/

4. In Java, there is an option to use **%n** to generate a correct newline independent of operating system. Modify the code to use **%n** instead of **\r\n** and make sure it works.

5. Experiment and try to figure out the different between these methods:

```
System.out.print("Line 1%n");
System.out.println("Line 1%n");
System.out.printf("Line 1%n");
```

2 – Printf

1. This code does not work, please correct the bug so that it works:

```
String name="Java";
int x=42;
float f = 3.14f;

System.out.printf("%s %f %d", name,x,f);
```

2. Given these variables:

```
int x=12;
int y=345;
int z=6789;
```

   Write a printf statement that outputs:

```
  12
 345
6789
```

3. Modify the code so the output becomes:

```
0012
0345
6789
```

4. Explore the documentation and try some of the other parameters and options:
   https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

3.1 – The scanner class

The documentation for the scanner class can be found here:
https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

1. Write an application that asks for 3 integers and then prints them out on a single line:

```
Please enter 3 numbers, one per row:
123
456
78

Output: 123 456 78
```

2. The scanner can also take other sources as input, for example a string like:
(See the help link above for the solution)

```
Scanner in = new Scanner("123 232 2323");
```

Add the necessary code to the line above to it prints out the three numbers on the screen.
Should output:

```
123
232
2323
```

3.2 – The scanner class and reading from files

1. We can also read data from files using the scanner class.

Open **Notepad++** and create new text file that contains the following text and save it as **hello.txt** in **c:\temp**

```
Hello
Java!!!!
```

2. Write a program that use the scanner class to read from **hello.txt** and print out the first two lines on the screen using the **next()** method.

   You need to add these two import statements above the Main class:

```
import java.io.File;
import java.io.FileNotFoundException;
```

3. Run the application and you will receive an error like:

   Error:(13, 23) java: unreported exception java.io.FileNotFoundException; must be caught or declared to be thrown

   To solve this, you need to modify the main method declaration so it becomes:

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

   We will go into the exact reason for this error later when we deal with exception handling.

4. Run the application again and you should see the content of the file on the screen.

   It's also a best practice to close the scanner when we are dealing with files, like:

```
in.close();
```

Questions and concepts to study further on your own:

- What is string interning?
- Why can you not modify an existing string?
- What is the ASCII table?
- What can you do with JavaDoc comments?
- Difference between CR LF, LF and CR line break types?
- What is a code smell?
- What is a escape character?
    - How much commenting is better for coding? https://softwareengineering.stackexchange.com/questions/317055