Practical 5: Command Line Input/Output

You have so far learned (by experience mostly) how to send data from your program to the screen using the `System.out.println()` method. In this practical you will learn how to **input** data to your program during runtime.

This practical also intends to get you acquainted with online Java documentation and more specifically with the extremely useful Applications Programming Interfaces (APIs) of Java classes.

Task 1

Please consider the following program:

```
import java.io.*;

public class Program
{
    public static void main(String[] args) throws IOException
    {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);

        System.out.print("Enter some text: ");
        String text = br.readLine();

        System.out.println(text.length());
        System.out.println(text.toUpperCase());
        System.out.println(text.toLowerCase());
        System.out.println(text.charAt(0));
        System.out.println(text.charAt(text.length() - 1));
    }
}
```

The program creates a `BufferedReader` object called `br` using an `InputStreamReader` object called `isr` that in turn uses the `System.in` object. The `System.in` object is the standard input stream in Java that is associated with the keyboard by default.

If all this sounds overwhelming just remember that the first two lines of the `main` method result in an object (arbitrarily named `br` in the example above) that can be used to read input from the keyboard.

Apart from reading the input stream, there are several other new things to learn:

What do you think is the difference between the, by now familiar to you, `System.out.println()` and `System.out.print()` methods?

Have a look at the various methods of the `String` class used in the program above. Can you provide an explanation of each output along with the result? For example you could change the line:

```
System.out.println(text.length());
```

to

```
System.out.println("The text contains " + text.length() +
" characters.");
```

Have a look at the **`String` class API** at:

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

for information about the above `String` class methods.

Task 2

Note in the example program above that the `readLine()` method of the `BufferedReader` class returns data of type `String`. Verify this by looking up the `BufferedReader` class in the official Java documentation on the web (hint: try searching for "`Java BufferedReader class`" using your favourite search engine).

Task 3

What if you want to read **numeric values** from the user in order to use them in numeric expressions?

Have a look at the (first occurrence of the) `parseInt` method in the API of a class called `Integer`.

You should be able to infer that, for example, the following code segment:

```
...
String someText = "15";
int number = Integer.parseInt(someText);
...
```

...converts a string containing characters (that can be converted into an `int` data type) into an `int` data type.

Write a program that reads an integer from the user (through the keyboard), adds 100 to it and displays the result.

Task 4

What if you want to read a `float` from the user?

Change your program in the task above so that it accepts a `float` from the user, adds the number `3.14` to it and displays the result.

This task will require you to infer what to do from your experience in the previous task. The information that you will need has not been given to you explicitly.

Ask a demonstrator to help you if you are stuck.

Task 5

Write a program that asks the user for their first name, a second name and the year of their birth and then displays a message with their full name and age, for example:

```
Fred Bloggs, you are 21 years old.
```