Exercises module 8 – Strings

8.1 – Chars and strings

1. Write a program that prints out this string backwards on the screen using a for-loop that is counting down.

```
char[] a = {'J', 'a', 'v', 'a', 'R', 'u', 'l', 'e', 's'};
```

2. Write a program that prints out every second character of the char array above to the screen, the output should be “JvRls”. Do not use the modulus operator, just do it with a single for-loop

3. Write a program that converts the char array in the previous step to a **string** and then prints out the **string** to the console.

   Try it first without using **StringBuilder** and then using **StringBuilder**.

4. Write a program that converts this string to an array and then print out the array to the screen:

```
String name ="Java Svensson";
```

5. Write a program that converts and splits this string to an array of integers and print out the array to the screen:

```
String input ="1,2,4,9,8,7,6,4,1";
```

8.2 – ASCII table

1. Write a program that prints out the ASCII-table in a nice way on the screen. Google to find out what an ASCII table is and how they can be formatted. Explore what additional characters we can send to the Console.

   Some tips

- What characters are part of the ASCII table? 0-127 or 0-255?
- An ASCII table can be represented in many different ways, choose one!
- Read more about ASCII here https://en.wikipedia.org/wiki/ASCII

Bonus if you can include the **HEX** (hexadecimal) value of each character. Display it either as a nice list or if you want to be more advanced as a grid. If you are unsure about hex math do look that up!

8.3 – Input validation

1. We usually need to do Input validation when we receive data from the user.

   When we do input validation we check the input against various rules and conditions to make sure it's correct and within limits.

   First write an application that prints out "**Input error**" if the input string is too long (longer than 10 characters), otherwise it prints "OK".
   Sample:

```
Input:  ShortStr
Output: OK

Input: ThisISALongString
Output: Input error
```

2. Now try to implement the following validation rules, try one at a time. Run only one at a time. Print out **OK** if the rule is satisfied and **Input error** if the rule is broken.

Input string length must be larger than 5
Input string length must be between 4-8 characters
Input string must start with 2 numeric characters (Hint google for Char.IsDigit)
String must contain "borg", like "Helsingborg"
String must end with a colon ':' character
All characters in the input must be all upper-case
The input string must contain the characters '(' and ')'. And the ')' must come after the '('

Important:

- Do test your logic so that it works for empty strings and strings with one character!
- To get more examples of strings that can cause trouble visit:
  - big-list-of-naughty-strings
    https://github.com/minimaxir/big-list-of-naughty-strings

8.4 - String manipulations

The following mini-exercises requires you to sanitize the Input, to remove various parts from the user provided input.

Each step is independent and you don't need to combine them.

If the input starts with +46 then remove it, like if the input is +4642123456 it should return 42123456.
Remove all dot '.' Characters from the input, like "127.0.0.1" should return 127001.
This string of comma-separated integers sometimes contains negative values. Like "123, 45,-34,231, 0,-1,-12312 , 12312,23423" Make a function that replaces all negative values with zero. It should return a new comma-separated string like: "123,45,0,231,0,0,0,12312,23423" (all spaces should also be removed if there are any present)
If the input does not end with "." Then add a "." At the end of the string.

8.5 StringBuilder #1

We often need to generate HTML from Java and in this exercise, we need to generate a complete HTML table using a **StringBuilder** and output the result to the screen.

```
<table>
  <tr>
     <td>Row1</td>
     <td>xxxxx<td>
  </tr>
  <tr>
     <td>Row2</td>
     <td>xxxxx<td>
  </tr>
  <tr>
     <td>Row3</td>
     <td>xxxxx<td>
  </tr>
   . . . . .
  <tr>
     <td>Row9</td>
     <td>xxxxx<td>
  </tr>
</table>
```

Using **StringBuilder** write the code necessary to generate the HTML table above that contains 10 rows. And where the first **<td>** in each row contains the row number.

Print out the result to the screen.

8.6 StringBuilder #2

Modify the code in #1 so that the output uses different background color for odd and even rows. Like this output: (hint: use the modulus operator)

```
<table>
  <tr bgcolor="#cccccc">
     <td>Row1</td>
     <td>xxxxx<td>
  </tr>
  <tr bgcolor="#ffffff">
     <td>Row2</td>
     <td>xxxxx<td>
  </tr>
  <tr bgcolor="#cccccc">
     <td>Row3</td>
     <td>xxxxx<td>
  </tr>
  <tr bgcolor="#ffffff">
     <td>Row4</td>
     <td>xxxxx<td>
  </tr>
    . . . . .
</table>
```

8.6 – Random

1. Read up on the **Java Random** documentation first, so you understand how it works.
   https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

2. Look at the code below and try to reason about it before you run it.
3. Then try to answer:
   - What does it do?
   - Will the output be the same if you run it multiple times?
   - What is the output like?

```
Random rand = new Random(-229985452);
for(int i=0;i<5;i++)
{
    int r=96 + rand.nextInt(27);
    char c = (char)r;
    System.out.print(c);
}

rand = new Random(-147909649);
for(int i=0;i<5;i++)
{
    int r=96 + rand.nextInt(27);
    char c = (char)r;
    System.out.print(c);
}
```

A more in-depth discussion about this can be found here:
http://stackoverflow.com/questions/15182496

Also try the code found here:
http://ideone.com/RnDYd2

Exercise 8.7 - Receipt

In this exercise, you'll write your own console application which will ask the user for some details about a product, and print a receipt for a purchase of that product.

We will ask for the product's name, price, quantity, and whether it is a food item or not. If it's a food item, we will apply a tax rate of 10%, otherwise 25%.

A test run might look like:

```
Enter the product name: AAA Batteries
Price per unit: 5
Quantity bought: 4
Food item? (y/n): n

--- RECEIPT ---
 Product: AAA Batteries

 Total amount to pay, excluding tax: 20
 Total amount to pay, including tax: 25
 Of which, tax is: 5
---------------
```

Main Task

1. Create a new **Console project**.
2. Find the entry point in the **main** method in the **Main.java** file.
3. In the main method, add code which will ask the user to input the product's name.
4. After that, add code which will ask the user the price of the product.
5. Then, add code which will ask the user the quantity bought.
6. Finally, ask the user whether this is a food item or not.
7. Calculate the price pre-tax, the amount of tax, and then the total price including tax.

Tip: We cannot compare strings in Java using ==. Instead, you have to write the following:

```
if (foodItem.equals("y"))
```

Print out this data in a format similar to that above.

Questions and concepts to study further on your own:

- A string is immutable, what does that mean?
- When should you use StringBuilder?
- Understand how to compare two strings
  https://stackoverflow.com/questions/513832/how-do-i-compare-strings-in-java
- Understand null
  - What does null mean?
    https://en.wikipedia.org/wiki/Null_pointer
  - Why is null bad?
    http://www.yegor256.com/2014/05/13/why-null-is-bad.html
  - Is it a bad practice to use null in Java?
    https://www.quora.com/Is-it-a-bad-practice-to-use-null-in-Java
  - Avoiding != null statements
    https://stackoverflow.com/questions/271526

- A Brief History of Unicode
  https://www.infoq.com/presentations/unicode-history?utm_term=Java
- The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)
  http://www.joelonsoftware.com/articles/Unicode.html
- Random values and the random seed value