(Prepared by Avraham Shalev, [avraham.shalev@post.idc.ac.il](mailto:avraham.shalev@post.idc.ac.il))

**Note:** highlighted text is when you have spare time \ for advanced students

**Functions workshop**

The aim of this workshop is to help students feel more comfortable with using and writing functions, while focusing on basic string operations and practicing loops.

**Methodology**

1. Recap Strings, Loops, Functions (15 mins)
   1. Strings (5 mins)
      1. Reminder
      2. Warm-up question
   2. Functions (10 mins)
      1. Warm-up question
   3. Loops will be practiced in 2 (Genes)
2. Genes (30 mins)
3. HTML HW (5 mins + questions in the second half of the workshop).
   1. This part contains additional exercises that you can give them as self work exercise for home or you can look on it and the solution together.

**Strings**

**Optional reminder (if have a time, enriching material):** (added by Ilia Kohanovski)**:**

Discuss what `System.out.print( (int) ‘A’ );` prints and why. That actually under the hood every character is just a number that represent character using UTF-16 encoding (16 bits). First 128 symbols are ASCII compatible. <https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html>

So String actually can contain smiles for example.

**Reminder (2-3 mins):**

For String s - 4 functions we saw and discussed:

s.length()

s.charAt()

s.indexOf()

s.substring()

We want to demonstrate the different uses of the overloads.

The list of them (taken from the java documentation: [String (Java SE 21 & JDK 21)](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/String.html) (update the version of the link every time) ):

int length()

char charAt​(int index)

int indexOf​(int ch)

int indexOf​(int ch, int fromIndex)

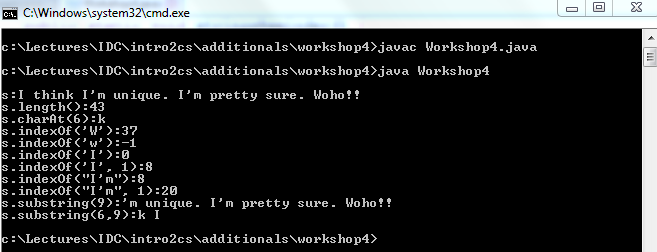
int indexOf​(String str)

int indexOf​(String str, int fromIndex)

String substring​(int beginIndex)

String substring​(int beginIndex, int endIndex)

In the associated workshop’s java file, method “stringsReminder()” demonstrates each overload:

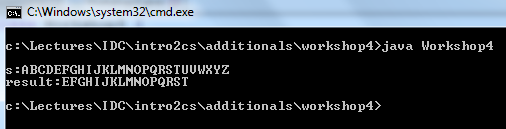


**Note:** this java file contains notes (as comments) above almost each method. You may share some\all of them with the students (and add by yourself..).

**Warm-up question (2-3 mins):**

Discuss with students how to solve the following question.

Start with the String s = "ABCD4UVWXYZ". From this String, construct a new String which contains all letters between ‘E’ and ‘T’ (including both). Students may only use String functions.

****

**Note**: s.substring(start,end) is [start,end) = [start, end+1]

Solution is In the associated workshop’s java file, method “stringsWarmup()”

**Functions**

**Warm-up question (see substring function in the code):**

Ask students to write a function which receives one String and two characters. The functions should return the substring of the input starting from the first character and terminating at the second character (including both). If one of the characters does not exist or if the last character appears only before the first character - return the empty String.

After the students solved it, when you overview the solution, tell them how you tackled this challenge (by the order you want them to use when writing functions):

1. what is my function signature?
   1. Return type - String
   2. Name - substring
   3. Arguments - 3 - String, char, char (+how should i name them?), start&end sounds good
   4. So we are going to write function: String substring(String s, char start, char end)
2. Discuss a few this good to have in mind
   1. is there such a function? If yes, use it and you are done!
   2. if not, what is there?
      1. String substring(String s, int startIndex, int endIndex)
      2. What is the difference? Maybe we can use it by doing a little extra work?
   3. Think of **edge-cases** and what you want to do (both chars are the same, maybe one of them is empty, ..)
3. Now, write a pseudo-code of the function String substring(String s, char start, char end):

startIdx = index of starting char

endIdx = index of ending char (**after startIdx**)

If (startIdx == -1) OR (endIdx == -1) OR (endIdx<startIdx)

result = “”

else

result = substring(startIdx, endIdx)

1. Transform that pseudo-code to java code and you are done!
   1. Consider little changes from pseudo-code to java code
      1. substring() method in this case should get endIdx**+1**
      2. Need to check string boundaries, .. (we are not doing this in this stage)

Solution is In the associated workshop’s java file, method “functionsWarmup()”

**Note to tell students:** In this way of thinking and challenge solving, no matter in what language you will code in the future, all 3 first steps will remain the same. The only change will be the last (4) stage, which is pretty minor.

Note for you only: edge-cases and boundaries were not included in the code, to focus what matters.

**Genes exercise**

The aim of this exercise is to help students feel more comfortable using\writing functions (with loops, dealing strings, ..) while solving real-life problems (by converting them to algorithms).

**What are Genes and how to identify them**

For our purposes a DNA strand is considered to be a String in which all characters are one of the four possibilities: ‘A’, ‘C’, ‘G’, ‘T’ (in reality every such letter represents one of four possible biological compounds).

Every three letters compose a single word, or codon (in reality, it codes for amino acids).

Thus, a valid DNA strand should have length divisible by three.

**Note:** Make sure students know how to check that (len%3==0).

A DNA strand can contain many genes. A gene is a sequence of codons inside the strand which is enclosed between two special codos, a start codon and an end codon.

Today, we will only consider one possible start and end codons. The start codon is given by “ATG” and the stop codon is given by “TAG”.

For example if the DNA is "GGTCCGATGCCTGGGCTCTAGGAA", it contains a single gene "ATGCCTGGGCTCTAG".

Another example if given with the DNA "ATGTTCAAATAGCCCTTA" and the gene "ATGTTCAAATAG".

We are going to write a function that takes a DNA strand as an input and returns genes contained inside it. We will do it step-by-step.

**Questions:**

Before each question, discuss briefly the possible challenge:

Before Q1: What happens if “TAG” appears before “ATG”.

Before Q2: What happens if the String looks like this “ATGGTAG”, which is not a valid gene?

Before Q3: Maybe there are additional “TAG”s that make a valid gene? Use loops.

Before Q4: I guess you wrote it all in one function. Well, that’s not good practice.

Before Q6:Duplicating code is BAD!!! consider loop! with X iterations as parameter from the outside (maxIterations = 2,10,.. While -1 can be interpeted unlimited)

1. Find a gene candidate (starts with “ATG” and ends with “TAG”) in a given DNA strand, if there is one. Otherwise return “”.
2. Now check if the gene is valid (divided by 3) and only in this case return it.
3. Try to find a valid gene even if the first ending-codon is not a real ending codon.
4. Break your solution into functions. You can show them the solution.
5. Find 2 genes in a DNA strand.
6. Find all genes in a DNA strand.

Possible discussion:

* What should we change if we want to allow for the input String to be with lower case letters as well?
  + Just one additional call in the start using:

| dna = dna.[**toUpperCase**](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/String.html%23toUpperCase())() |
| --- |

**HTML exercise (HW)**

Discuss students all what we saw with Genes.

Explain them that this can be done to any data we can refer as string…!

It doesn’t matter whether what we should find:

* A single gene from a DNA strain
* An url from a HTML’s source code
* A sentence from an article \ a news
* A word from a sentence
* Few letters from a word

In all that cases, we (as programmers) should act the same:

* Understand what is the pattern of what we search
* Understand how to express (in code) the logic to validate it
* Search what we want (using indexOf() on the pattern)
* Extract only it from the whole data (using substring)

Dan wrote the exercise and solution files, so you can look at them:

Exercise file: FindURLS HW.docx

Solution (code) file: FindURLs.java