Part 3

Using strings

Learning Objectives

- Revising reading, printing and comparing Strings
- Knowing how to split a string into several pieces

Let's first revise what we already know about Strings and see how to split them. Below we create a String variable `magicWord`, that contains value `"abracadabra"`.

```
String magicWord = "abracadabra";
```

Passing a String as a parameter to a print command (or, for that matter, any method that takes a String parameter) happens in the familiar way:

```
String magicWord = "abracadabra";
System.out.println(magicWord);
```

Sample output

```
abracadabra
```

Reading and Printing Strings

You can read a string using the nextLine-method offered by Scanner. The program below reads the name of the user and prints it:

```
Scanner reader = new Scanner(System.in);
```

```
System.out.print("What's your name? ");
// reading a line from the user and assigning it to the name variable
String name = reader.nextLine();

System.out.println(name);
```

Sample output

```
What's your name? Vicky
Vicky
```

Strings can also be concatenated. If you place a +-operator between two strings, you get a new string that's a combination of those two strings. Be mindful of any white spaces in your variables!

```
String greeting = "Hi ";
String name = "Lily";
String goodbye = " and see you later!";

String phrase = greeting + name + goodbye;

System.out.println(phrase);
```

Sample output

```
Hi Lily and see you later!
```

Programming exercise:

Print thrice

Points 1/1

Write a program, that reads a string from the user and then prints it three times.

Sample output

```
Give a word: cake

cakecakecake
```

NB! The program should ask for only one string. Don't use a loop here.

Exercise submission instructions

How to see the solution

String Comparisons And "Equals"

Strings can't be compared with with the equals operator - `==`. For strings, there exists a separate `equals`-command, which is always appended to the end of the string that we want to compare.

```
String text = "course";

if (text.equals("marzipan")) {
    System.out.println("The text variable contains the text marzipan.");
} else {
    System.out.println("The text variable does not contain the text marzipan.")
}
```

The `equals` command is always appended to the end of the string that we want to compare, "string variable dot equals some text". You can also compare a string variable to another string variable.

```
String text = "course";
String anotherText = "horse";

if (text.equals(anotherText)) {
    System.out.println("The two texts are equal!");
} else {
    System.out.println("The two texts are not equal!");
}
```

When comparing strings, you should make sure the string variable has some value assigned to it. If it doesn't have a value, the program will produce a *NullPointerException* error, which means that no value has been assigned to the variable, or that it is empty (*null*).

As we've come to know, a boolean value can be inverted through negation - **!** .

```
System.out.println("Make sure the text is not 'cake'");
String text = "pie";

if (!(text.equals("cake"))) {  // true if the condition text.equals("cake") is
    System.out.println("it wasn't!");
} else {
    System.out.println("it was!");
}
```

Sample output

```
it wasn't!
```

Programming exercise:

Is it true

Points 1/1

Write a program that asks the user for a string. If the user writes the string "true", the program prints "You got it right!", otherwise it prints "Try again!".

Sample output

```
Give a string: true
You got it right!
```

Sample output

```
Give a string: trueish
Try again!
```

Exercise submission instructions

How to see the solution

Programming exercise:

Login

Points 1/1

Write a program that recognizes the following users:

username	password
alex	sunshine
emma	haskell

The program either shows a personal message or informs of an incorrect username or password.

Sample output

```
Enter username: alex
Enter password: sunshine
You have successfully logged in!
```

Sample output

```
Enter username: emma
Enter password: haskell
You have successfully logged in!
```

Sample output

Enter username: alex
Enter password: thunderstorm
Incorrect username or password!

NB! You can't compare strings with `==`!

NB! Logins should not be implemented like this in real life! You can become familiar with safer ways to implement logins on courses focusing on web programming.

Exercise submission instructions

How to see the solution

Splitting a String

You can split a string to multiple pieces with the `split`-method of the String class. The method takes as a parameter a string denoting the place around which the string should be split. The `split` method returns an array of the resulting sub-parts. In the example below, the string has been split around a space.

```
String text = "first second third fourth";
String[] pieces = text.split(" ");
System.out.println(pieces[0]);
System.out.println(pieces[1]);
System.out.println(pieces[2]);
System.out.println(pieces[3]);

System.out.println();

for (int i = 0; i < pieces.length; i++) {
    System.out.println(pieces[i]);
}
```

Sample output

first
second
third
fourth

first
second
third
fourth

Programming exercise:

Line by line

Points
1/1

Write a program that reads strings from the user. If the input is empty, the program stops reading input and halts. For each non-empty input it splits the string input by whitespaces and prints each part of the string on a new line.

Sample output

once upon a time
once
upon
a
time
a little program
a
little
program
halted
halted

Exercise submission instructions

How to see the solution

Programming exercise:

AV Club

Points
1/1

Write a program that reads user input until an empty line. For each non-empty string, the program splits the string by spaces and then prints the pieces that contain `av`, each on a new line.

Sample output

```
java is a programming language
java
navy blue shirt
navy
```

Sample output

```
Do you have a favorite flavor
have
favorite
flavor
was it a cat?
```

Tip! Strings have a `contains`-method, which tells if a string contains another string. It works like this:

```
String text = "volcanologist";

if (text.contains("can")) {
    System.out.println("can was found");
}

if (!text.contains("tin")) {
    System.out.println("tin wasn't found");
}
```

Sample output

can was found
tin wasn't found

Exercise submission instructions

How to see the solution

Data of Fixed Format

Splitting strings is used particularly when the data is of a fixed format. This refers to data that adheres to some predefined format. An example of this of this is the comma-separated values (`csv`) format, where commas are used to separate values. Below you'll find an example of data in csv form containing names and ages. The first column contains names and the second one ages. The columns are separated by a comma.

Sample data

sebastian,2
lucas,2
lily,1

Let's assume the user enters the data above row by row, ending with an empty line.

A program to print the names and ages looks like the following:

```
Scanner reader = new Scanner(System.in);

while (true) {
    String input = reader.nextLine();
    if (input.equals("")) {
        break;
    }

    String[] pieces = input.split(",");
```

```
    System.out.println("Name: " + pieces[0] + ", age: " + pieces[1]);
}
```

Sample output

```
sebastian,2
Name: sebastian, age: 2
lucas,2
Name: lucas, age: 2
lily,1
Name: lily, age: 1
```

Programming exercise:

First words

Points

1/1

Write a program that reads user input until an empty line is entered. For each non-empty line the program splits the string by spaces and prints the first part of the string.

Sample output

```
one two three four
one
this is a very important message
this
```

Exercise submission instructions

How to see the solution

Programming exercise:

LastWords

Points
1/1

Write a program that reads user input until an empty line is entered. For each non-empty line the program splits the string by spaces and prints the last part of the string.

Sample output

one two three four
four
this is a very important message
message

Tip! You can find out the length of the array like this:

```
String[] parts = {"one", "two", "three"};
System.out.println("Number of parts: " + parts.length);
```

Sample output

Number of parts: 3

Exercise submission instructions

How to see the solution

Secret messages

In the exercises above, we actually implemented a very simple decrypting method for secret messages. One variant of these hidden messages consists of the first character of each line. For example, the

secret message "program" is hidden in the (somewhat cryptic) text below.

Sample data

```
Polymorphous computations elaborate.
Real calculators honour.
Older desktops deliver.
Great mainframes link.
Reversed devices install.
Additional workstations modem.
Many microcomputers letter.
```

Let's continue along the same theme! You can get a character at a specified index of a string with the `charAt` method.

```
String text = "Hello world!";
char character = text.charAt(0);
System.out.println(character);
```

Sample output

```
H
```

Using Diverse Text

We've printed strings in the examples above. Some of the data contained in a fixed-format string can be numerical. In the previous data we used that contained names and ages, the ages were integers.

Sample data

```
sebastian,2
lucas,2
lily,1
```

Splitting a string always produces an array of strings. If the text is of fixed format, we can assume the data in a specific index to always be of the a

specific type — e.g., in the example above, age at index 1 is an integer.

The program below computes the sum of ages in this fixed format data. In order to compute the sum, the age must first be converted to a number (the familiar command `Integer.valueOf()`)

```
Scanner reader = new Scanner(System.in);
int sum = 0;

while (true) {
    String input = reader.nextLine();
    if (input.equals("")) {
        break;
    }

    String[] parts = input.split(",");
    sum = sum + Integer.valueOf(parts[1]);
}

System.out.println("Sum of the ages is " + sum);
```

Sample output

```
sebastian,2
lucas,2
lily,1

Sum of the ages is 5
```

We can write a program to compute the average of the ages in the same way:

```
Scanner reader = new Scanner(System.in);
int sum = 0;
int count = 0;

while (true) {
    String input = reader.nextLine();
    if (input.equals("")) {
        break;
    }

    String[] parts = input.split(",");
    sum = sum + Integer.valueOf(parts[1]);
```

```
    count = count + 1;
}

if (count > 0) {
    System.out.println("Age average: " + (1.0 * sum / count));
} else {
    System.out.println("No input.");
}
```

Sample output

```
sebastian,2
lucas,2
lily,1

Age average: 1.666
```

Programming exercise:

Age of the oldest

Points
1/1

Write a program that reads names and ages from the user until an empty line is entered. The name and age are separated by a comma.

After reading all user input, the program prints the age of the oldest person. You can assume that the user enters at least one person, and that one of the users is older than the others.

Sample output

```
sebastian,2
lucas,2
lily,1
hanna,5
gabriel,10
Age of the oldest: 10
```

Exercise submission instructions

How to see the solution

Programming exercise:

Name of the oldest

Points 1/1

Write a program that reads names and ages from the user until an empty line is entered. The name and age are separed by a comma.

After reading all user input, the program prints the name of the oldest person. You can assume that the user enters at least one person, and the that one of the users is older than the others.

Sample output

```
sebastian,2
lucas,2
lily,1
hanna,5
gabriel,10

Name of the oldest: gabriel
```

Exercise submission instructions

How to see the solution

Length of string

In the next exercise you'll be asked for the length of the names. You can find out the length of a string with `length()`-method:

```
String word = "equisterian";
int length = word.length();
System.out.println("The length of the word" + word + " is " + length);
```

Sample output

```
The length of the word equisterian is 11
```

Programming exercise:

Personal details

Points 1/1

Write a program that reads names and birth years from the user until an empty line is entered. The name and birth year are separated by a comma.

After that the program prints the longest name and the average of the birth years. If multiple names are equally longest, you can print any of them. You can assume that the user enters at least one person.

Sample output

```
sebastian,2017
lucas,2017
lily,2017
hanna,2014
gabriel,2009

Longest name: sebastian
Average of the birth years: 2014.8
```

Sample output

```
sauli,1948
tarja,1943
martti,1936
```

mauno,1923
urho,1900

Longest name: martti
Average of the birth years: 1930.0

Exercise submission instructions

How to see the solution

You have reached the end of this section! Continue to the next section:

→ 5. Summary

Remember to check your points from the ball on the bottom-right corner of the material!

In this part:

Source code of the material

This course is created by the Agile Education Research -research group of the University of Helsinki.

Credits and about the material.