Learning goals

Before the next day, you should have achieved the following learning goals:

- Understand the difference between simple and complex data types.
- Be able to create your own classes, and to create new instances of those clases using `new`.
- Understand what a field is, and how to access the fields in an object (to read it or to alter —write— it).
- Strengthen your understanding of loops.

Remember that star exercises are more difficult. **Do not try star-exercises unless the other ones are clear to you**.

1 Equality of floating-point numbers

Look at the following code. What does it do? What will it print on the screen?

```
double d1 = 1.255
double d2 = d1 + 7 - 4 - 3
if (d1 == d2) {
    println("1.255 is equal to 1.255 plus 7 minus 7");
} else {
    println("1.255 is NOT equal to 1.255 plus 7 minus 7");
}
```

Execute the program. Does it print what you expected it to print? Why? (Hint: print the value of `d1` and `d2`; why do they have these values?). If the program is not behaving correctly, fix it.

2 Calculator

Write a program that reads two numbers from the user and then offers a menu with the four basic operations: addition, subtraction, multiplication, and division. Once the user has selected an operation from the menu, the calculator performs the operation.

Hint: In the same way that there exists an Integer.parseInt() method to parse integers, there is a Double.parseDouble() method to parse real numbers.

3 Command-line calculator (*)

Write a program that reads a text representing a mathematical operation (one of the four basic ones) with two operands, and then execute it. For example, if the user enters "3/5" the program outputs "0.6"; if the user enters "23 * 4" the program outputs "92".

4 Distance point–to–point

Write a program that reads the X and Y coordinates of three points and then outputs which of the three are closer. Use the following class to store the data for the points:

```
class Point {
  double x;
  double y;
}
```

Hint: The distance from (x_1, y_1) to (x_2, y_2) can be calculated as

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

5 Rectangle

Write a program that reads the X and Y coordinates of two points and then outputs the area of a rectangle where both points are opposite corners. Use the following class to store the data for the points:

```
class Rectangle {
  Point upLeft;
  Point downRight;
}
```

Your program should calculate (and write on the screen) the perimeter and area of the rectangle.

Note: For exercises 5, 6, and 7 you must access (i.e. read or write) the value of the coordinates of the points through the rectangle, not directly through the point, i.e. `myRectangle.upLeft.x`, not `point.x` or `x`.

6 Inside or outside

Write a program that reads the coordinates of the two points defining a rectangle and then the coordinates of a third point. The program must then determine whether the point falls inside or outside the rectangle.

7 Overlaps

Write a program that reads the coordinates of four points, the former two defining one rectangle and the latter two defining another one. The program must read the coordinates of a fifth point and say whether the point is inside both rectangles, inside one of them only, or out of both.

8 Line to column

Write a program that reads some text from the user and then writes the same text on screen, but each letter on a different line.

Now modify your program to write each word (as defined by spaces) rather than letter on a different line.

9 Counting letters

Write a program that reads some text from the user and then says how many letters 'e' are there in that text.

Then modify the program so that it reads the text from the user and then asks for a letter. The program should then say how many times you can find the letter in the text.

10 Counting pairs of letters (*)

Write a program that reads a short string and then some longer text. The program must say how many times you can find the short string in the text. You cannot use any method of String apart from `charAt()` and `length()`.

11 Counting letters redux (*)

Write a program that reads a text from the user and then enter a loop of requesting letters and counting them. The program stops if the user asks for the same letter twice. This is an example of the output of such a program (with a rather short and boring text):

```
Please write a text: It was a dark and stormy night
Which letter would you like to count now? a
There are 6 in your text.
Which letter would you like to count now? s
There are 3 in your text.
```

```
Which letter would you like to count now? u
There are 0 in your text.
Which letter would you like to count now? a
Repeated character. Exiting the program...
Thank you for your cooperation. Good bye!
```

12 Your change, please

Write a program that reads the total cost of a purchase and an amount of money that is paid to buy it. Your program should output the correct change specifying the amount of notes (50, 20, 10, 5) and coins (2, 1, 0.50, 0.20, 0.10, 0.05, 0.02, 0.01) needed.

13 Palindrome

A palindrome is a word, phrase, number, or other sequence of units that may be read the same way in either direction. Examples of strict palindromes include "ABBA", "madam", "radar", "kayak", and "step on no pets". Write a program that reads a text and detects whether the text is a strict palindrome.

14 Palindrome creator

Write a program that reads a text and then writes on the screen a palindrome by writing the same text followed by the same text in reversed order. For example, if the user enters the text "It was a dark and stormy night" the program must output "It was a dark and stormy nightthgin ymrots dna krad a saw tI".

15 Palindrome redux (*)

A strict palindrome is difficult to see in every day language. A relaxed palindrome, a text that is a palindrome if you ignore punctuation marks such as commas or spaces, is easier to see. Examples include "A man, a plan, a canal - Panama!", "Was it a car or a cat I saw?", and "Rise to vote, sir".

Write a program that reads a text from the user and then says whether the text is a relaxed palindrome. Note that all strict palindromes are relaxed palindromes by definition.

Hint: There are two methods that will make your life easier. The first one is Character.isLetter(), that accepts a character (not a String, even of length one) and returns true if the character is a letter (e.g. 'a', 'R') and false otherwise (e.g. '.', '5'). The second one is Character.toLowerCase(), that accepts a character (not a String) and returns the lower case version of the character.

16 Text2number

Write a program that reads a number with commas and decimal dots (such as "23,419.34") and then prints a number that is half of it. Do not use `Double.parseDouble()`.

If you were writing a simple spreadsheet, you could use this code to parse the input in the cells.

17 Mail server (*)

Let's implements part of a mail server. A mail server is a program that takes your emails and then sends them to the appropriate recipient. In this exercise, you will implement a simplified version of the SMTP protocol that is used to send emails over the Internet.

When your program starts, it should provide a command prompt to the user. Then it must read the return address of the email using a command of the form "MAIL FROM: <email-address>". The program must check that the command is properly written and that the email address is valid (i.e. contains one and only one "@" sign which is neither the first nor the last character). There is no need for the email address to actually exist, it only needs to be valid. If there is an error, the program must say so and wait for a correct return address.

Once the destination is correct, the program must say "OK" and wait for the recipient. The recipient must be introduced by the user with a command of the form "RCPT FROM: <email-address>". Once again, if the user enters an invalid command or email address, the program must wait for a correct one.

Once the return address and recipient are correct, and only then, the user can enter the command "DATA". The program reads then the body of the email, line after line, until the user enters a line that consists of only a dot. At that point, the email is ready and the program must write on the screen who is sending the email, to whom, and what the email says.

If at any point the users types "QUIT" the program must terminate. If the user enters any other command, or types one of these commands at the wrong time (e.g. RCPT before MAIL), the program must say "Invalid command" on screen. See a simple example below:

```
Welcome to My Mail Server!
>>> DATA
Invalid command.
>>> MAIL FROM: me@mypc.com
OK
>>> RCPT TO: bro
Invalid email address
>>> RPCT TO: bro@hispc.com
OK
>>> DATA
Hi bro,
Call Mum asap.
Take care,
.
Sending email...
from: me@mypc.com
to: bro@hispc.com
Hi bro,
Call Mum asap.
Take care,
...done!
>>> QUIT
Bye!
```

When you send an email from Thunderbird, Outlook, or your favourite email program, it communicates with a mail server in a way very similar to what you do in this exercise. Now you know why it seems easy for spammers to fake a sender's email adress: it really is that easy.