CODAPPS

Coding Cheatsheet

Clément Levallois

Version 1.0, last modified

Table of Contents

Variables and objects

General rules:

- The type of the variable (String, Long, Boolean…) start with a capital letter.
- variable names start without a capital letter.

String variables store text

Creating a String variable and giving it a value

```
String title = "Welcome to my app"; ①
```

① Don't forget the double quotes " "!

Integer variables store round numbers

Creating an Integer variable and giving it a value

```
Integer classSize = 32;
int anotherClassSize = 25; ①
```

① `int` is like `Integer`. It takes less memory but is sometimes less convenient to use. Also, note: no double quote!! Double quotes are just for `String`.

Float and Double variables store decimals

Double** is like **Float** but can store **decimals with a lot more precision

Creating Float and Double variables

```
Float pi = 3.14f; ①
Double piVeryPrecise = 3.14159265358979323846264338327950288419716 9d; ②
```

① Don't forget the `f` letter at the end of your number. `float` or `Float` can be used, `float` takes less memory than `Float`.

② Don't forget the `d` letter at the end of your number. `double` or `Double` can be used, `double` takes less memory than `Double`.

Long stores big round numbers

Long** is like **Integer** but can store **bigger numbers

Creating a Long variable

```
Long millisecondsSinceLastMonth = 3644340304304141; ①
```

① Don't forget the `l` letter at the end of your number. `long` or `Long` can be used, `long` takes less memory than Long but can be inconvenient to use.

Boolean stores true / false values

This seems not very useful but actually we use it quite often

Creating a Boolean variable

```
Boolean hasAStudentCard = true; ①
```

① A classic mistake is to write `"true"` (with double quotes, which is incorrect). Boolean values are `true` or `false` without double quotes `" "`. `boolean` can be used instead of `Boolean`: less memory but also less convenient to use in some cases.

`Objects`: to create and store a variety of things

A variety of objects exist - use them to create and store things

Creating an Object storing a Date

```
Date dateStartOfTheGame;  ①
dateStartOfTheGame = new Date(); ②
Date dateEndOfTheGame = new Date();  ③
```

① An object `dateStartOfTheGame` of type `Date` is declared. It is `null` at the moment.

② `dateStartOfTheGame` is instantiated: an instance of it is created.

③ Shortcut: a variable can be declared and instantiated in one line of code.

Methods

Creating (defining) a method

Creating a method adding the VAT to a price

```
private Float addFrenchVAT(Float priceWithoutVAT) { ①
   Float priceWithVAT; ②
   //the regular rate of the VAT in France is 20% so we multiply the price by 1.20 to
find the new price ③
   priceWithVAT = priceWithoutVAT * 1.20; ②
   return priceWithVAT; ②
}④
```

① title of the method you create, then the method start at the opening curly brace `{`.

② the method itself

③ an explanation, not some code! The line starts with `//` to show this is some explanations for humans like you and me, not some code in our app.

④ this closing curly brace signals the end of the definition of the method.

Creating a method which returns nothing

```
Float price = 5.99f;  ①
private void addFrenchVAT() {  ②
   price = price * 1.20; ③
} ④
```

① we have created a variable named `price`

② now we define a method like the one before in this lesson, except that:

- `Float` has been replaced by `void`, which is an English term meaning "nothing"
- it has no parameter: there is nothing in the parenthesis `( )`

③ the method does one thing: it multiplies the value of the variable `price` by 1.20

④ this is the end of the method. There is no "return" statement.

Classes

A class is just a file in your app. It contains the variables and the methods that you want. It looks like:

A simple example of what a class looks like

```
package net.clementlevallois.codapps.myfirstapp ①

public class Form1 { ②

Integer scorePlayer; ③

  public void addOneToScore() {
    scorePlayer = scorePlayer + 1;
  }

}  ④
```

① a class always starts with the name of the package where it belongs

② the name of the class (`Form1`) should have the same name as your file where it is writte (here the file would be Form1.java)

③ this is a variable which can be used anywhere in the class.

④ don't forget the closing curly brace of the class!

Instantiating a Form in MyApplication.java

```
public void start() {
    Form1 myForm1 = new Form1(); ①
    myForm1.show(); ②
}
```

① We instantiate our Form1

② **And now we can use methods of this Form1**. Here, we use the method `show()` which has for effect to display the `Form` on screen.

Another common way to instantiate an object is this one:

Getting the present time and storing it in a variable

```
public void start() {
  LocalTime timeNow = LocalTime.now(); ①
}
```

① This stores the time at the moment when this line of code is executed, in the variable `timeNow`

If... conditional statements

conditional statements for numbers

Different kinds of conditional statements about numbers

```
Float priceItemInEuros;
priceItemInEuros = 5.99f;
Label productLabel = new Label();
if (priceItemInEuros < 6) {
  productLabel.setText("cheap product!");
}
if (priceItemInEuros == 5.99) { ①
  productLabel.setText("the price is exactly 5.99");
}
if (priceItemInEuros != 5.99) { ②
  productLabel.setText("the price is different from 5.99");
}
if (priceItemInEuros =< 6) {
  productLabel.setText("the price is under or equal to 6!");
}
if (priceItemInEuros >= 7) {
  productLabel.setText("the price is above or equal to 6!");
}
```

conditional statements about text

It would be a **mistake** to write:

Mistake! Don't do this!

```
String playerName1 = "Tristan";
String playerName2 = "Tristan";

if (playerName1 == playerName2) { ①
  messageLabel.setText("the two players have the same name!");
}
```

① Using `==` to compare two Strings is incorrect.

- Your build will not fail, but even if the two players have the same name it might say it's false!
- when comparing two String, you should do like below:

Different kinds of conditional statements about text

```
String playerName1 = "Tristan";
String playerName2 = "Touni";

if (playerName1.equals(playerName2)) {
  messageLabel.setText("the two players have the same name!");
}
if (!playerName1.equals(playerName2)) { ①
  messageLabel.setText("the two players have different names!");
}
```

① note the `!` in front

conditional statements about several items

A statement with two conditions which need both to be true

```
Float priceItemInEuros;
priceItemInEuros = 5.99f;
Label productLabel = new Label();
if (priceItemInEuros < 6 & priceItem > 2) { ①
  productLabel.setText("relatively cheap product!");
}
```

① the `&` means "and". The two conditions: `priceItemInEuros < 6` **and** `priceItem > 2` both need to be true for the statement `productLabel.setText("relatively cheap product!");` to be executed.

A statement with two conditions where just either one of the two needs to be true

```
Float priceItemInEuros;
priceItemInEuros = 5.99f;
Label productLabel = new Label();
if (priceItemInEuros < 6 | priceItem > 2) { ①
  productLabel.setText("relatively cheap product!");
}
```

① the `|` means "or". Just one of the two conditions: `priceItemInEuros < 6` **or** `priceItem > 2` needs to be true for the statement `productLabel.setText("relatively cheap product!");` to be executed.

Loops

Writing a loop

```
for (int i = 0; i<100; i = i+1){
  System.out.println("I looped " + i) +" times.";
}
```

A loop with several ifs inside

```
for (int i = 0; i<100; i = i+1){
  System.out.println("I looped " + i + " times.");
  if (i == 0) {
    System.out.println("We just started the loops. This is going to be a long
journey.");
  }
  if (i == 50) {
    System.out.println("Half way already!");
  }
  if (i == 99) {
    System.out.println("This was the last loop. Bye!");
  }
}
```

ArrayLists

Creating a list and adding objects to it

```
ArrayList<Balloon> balloons = new ArrayList();
for (int i = 0; i<20000;i = i+1){
  Balloon myBalloon = new Balloon();
  balloons.add(myBalloon);
}
```

Looping through a list to show the names of all players

```
for (String player: playerNames) { ①
  Label myLabel = new Label();
  myLabel.setText(player);
  myForm.add(myLabel)
}
```

① here we assume that we had created before an ArrayList of player names.