

CODAPPS

Coding Cheatsheet

Clément Levallois

Version 1.0, last modified

Table of Contents

Variables and objects.....	1
String variables store text	1
Integer variables store round numbers	1
Float and Double variables store decimals	1
Long stores big round numbers	2
Boolean stores true / false values	2
Objects: to create and store a variety of things	2
If... conditional statements.....	2
conditional statements for numbers	2
conditional statements about text	3
conditional statements about several items	4
Loops	4
ArrayLists	5



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 = 3.141592653589793238462643383279502884197169d; 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.

If... conditional statements

conditional statements for 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.