## **CODAPPS**

### Variables and objects

Clément Levallois

Version 1.0, last modified

## **Table of Contents**

The interactive tool we are using to learn the basics of code	1
Variables	2
a. <b>String</b> variables: to store text	2
b. <b>Integer</b> variables: to store round numbers	4
c. Float, Double and Long variables: to store decimal numbers and big numbers	5
d. <b>Boolean</b> variables: to store true / false information	7
Objects	8
a. Creating and defining objects	8
b. Always the same logic: creating Buttons, Labels	9
e end	10



# 1. The interactive tool we are using to learn the basics of code

It is useful to learn about coding by reading a lesson like this one, but it is more efficient if you can **practice while you read**.

Indeed, a key competency in coding is to learn the simple discipline of writing text carefully:

- not forgetting a; at the end of the line
- not forgetting to put a capitalized letter when it is necessary
- learning how to spot opening accolades { and closing accolades }
- not confusing when to use commas, and semi-colons; knowning wen to put double quotes ""
  or not...

It is all very trivial, right?

And yet this is what needs the most practice, at the start.

So in this lesson, you will see plenty of interactive screens like this one:

if you follow this lesson on a pdf, you will not see the interactive screen in your document.



You can open a web browser and do the interactive exercise here:

https://repl.exploreyourdata.com/ui/console.html

or here:

http://www.javarepl.com/term.html

"Interactive screen" means that **you can write directly on it** and see your code in action - yes, right here: click next to "java", a cursor will blink to show that you can write:

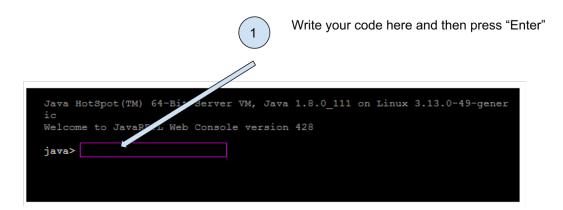


Figure 1. How to use the interactive tool

Last note before we start: this lesson is on the <u>essentials</u> of coding, so we go fast and discuss just the most important notions of coding.

If you are interested in (much) longer, more thorough approaches, you can have a look at these two interactive courses:

- "Think Java" by Tinklet (Java is the same programmming language as the one we use here)
- "Python for Everybody" also by Tinklet (Python is the most popular programming language for data science)

### 2. Variables

If we compare coding to cooking, you could say that variables are the ingredients of the recipe.

→ variables are a way to create, define and store all the information we need in our mobile app.

What kind of information do we need to store?

- if you create a **gaming app**, you might need to store the name of the player, the score, the energy level, etc...
- if you create an **app selling a service**, you might need to store a login name, a password, a list of products and their prices...

As you see, variables can be of different types: text, numbers, numbers with decimals, ...



You can create all the kinds of variables you want, but to help you, a number of them are already predefined because they are so common:

#### a. String variables: to store text

A variable of type "String" specializes in storing textual information. It is created and defined this

way:

Creating a String variable and giving it a value

```
String playerName; ①
playerName = "Bernard B"; ②
```

- ① We create the variable (this needs to be **done just once**)
  - String The capital **S** is mandatory. It means this variable **specializes in storing text**. This is the **type** of the variable and it always **stands in front** of the name of the variable.
  - playerName this is the name I chose for the variable, it could have been anything else. By convention it always starts without a capitalized letter and it has no space in it.
  - The ; shows the end of this instruction and **is mandatory** (if you don't put it, the program tries to read the next line as the direct continuation of this one, and it gets confused).
- ② A value is assigned to the variable (this can be done as many times as we need)
  - playerName: our variable, created just above.
  - =: the right of the equal sign will be the value of the variable.
  - "Bernard B": this is the value I store in the variable playerName. Textual values should be put between quotes " ".
  - The ; shows the end of this instruction and **is mandatory**.

There is a shortcut if you want to create a variable and give it a value, just in one line of code:

A shortcut

```
String playerName = "Bernard B";
```

So this single line of code creates a variable called playerName, and I immediately give it a value: Bernard B.

This will be handy when we need to show the player's name on the screen of the app: we will just use the variable player, and what ever value in it (the player's name) will be shown.



Create a variable called "favoriteMovie" and store the name of your favorite movie in it

Here is the solution:

- Declaring the variable
- Assigning a value to the variable
- 3 Just an automatic feedback by the computer: the variable has been recorded.

Figure 2. Creating a variable for a textual information

#### b. Integer variables: to store round numbers

A variable of type "Integer" specializes in storing round numbers (like 1, 2, 3...). It is created and defined this way:

Creating an Integer variable and giving it a value

```
Integer playerAge; ①
playerAge = 22; ②
```

- ① We create the variable (this needs to be **done just once**)
  - Integer The capital I is mandatory. It means this variable specializes in storing round numbers. This is the type of the variable.
  - playerAge this is the name I chose for the variable, it could have been anything else. By convention it always starts **without** a capitalized letter and it has **no space** in it.
  - The ; shows the end of this instruction and **is mandatory** (if you don't put it, the program tries to read the next line as the direct continuation of this one, and it gets confused).
- ② A value is assigned to the variable (this can be done as many times as we need)
  - playerAge is our variable, created just above.
  - = the right of the equal sign will be the value of the variable.
  - 22 is the value I store in the variable player Age. **Don't use double quote around the value**.
  - The ; shows the end of this instruction and is mandatory.

To repeat: we did not put double quotes around 22! Double quotes " " are only for String variables!

There is the same shortcut as usual if you want to create a variable and give it a value, just in one

line of code:

A shortcut

```
Integer playerName = 18;
```

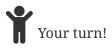
So this single line of code creates a variable called playerAge, and I immediately give it a value: 22.

There is another way to use variables for round numbers:

Another way

```
int playerName = 18; ①
```

1 Integer has been replaced by int. Both are the same, except that int takes even less space in the memory of your program.



- 1. Create an Integer variable called score and store 999999 in it.
- 2. *Create a variable called* purchasedItems, *using* int *instead of* Integer. *Store* 4 *in it.*

Here is the solution:

```
Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_111 on Linux 3.13.0-49-generic Welcome to JavaREPL Web Console version 428

java> Integer score;
java> score = 999999;
java.lang.Integer score = 999999
java> int purchasedItems;
java> purchasedItems = 4;
java.lang.Integer purchasedItems = 4
java>
```

Figure 3. Creating 2 variables to store numbers

# c. Float, Double and Long variables: to store decimal numbers and big numbers

Variables of type "Float" and "Double" specialize in storing decimal numbers (like 1.4533).

The difference between Float and Double? Double can store even more decimals than Float.

Variable of type "Long" specializes in storing loooong numbers (like 9395353439449039035353). It is useful when you need to count milliseconds, for example.

These are created and defined this way:

Using Float, Double and Long variables

```
Float averagePrice;
averagePrice = 15.34f; ①
Double piValue;
piValue = 3.14159265358979323846d; ②
Long timeOfPlayinMilliSeconds;
timeOfPlayinMilliSeconds = 725853353505351; 3
```

- ① We added the letter f at the end of our number, to signal that this is a **Float**.
- ② We added the letter d at the end of our number, to signal that this is a **Double**.
- 3 We added the letter 1 at the end of our number, to signal that this is a **Long**.

There is the same shortcut as usual if you want to create a variable and give it a value, just in one line of code:

A shortcut

```
Float averagePrice = 15.34f;
Double piValue = 3.14159265358979323846d;
Long timeOfPlayinMilliSeconds = 725853353505351;
```

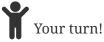
There is another way to use these types of variables, again for a gain in memory:

Another way

```
float averagePrice = 15.34f;
double piValue = 3.14159265358979323846d;
long timeOfPlayinMilliSeconds = 725853353505351;
```

Float is replaced by float, Double is replaced by double, and Long is replaced by long.

This looks like just a small change, but the smaller cap version takes even less space in the memory of your program.



- 1. Create a variable called discountPercentage and store 0.33 in it.
- 2. Create a variable called dollarToEuro, using double. Store 0.80240500000000003 in it.
- 3. Create a variable called milliseconds, using long. Store 25343353530285753 in it.

Here is the solution:

```
Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_111 on Linux 3.13.0-49-generic Welcome to JavaREPL Web Console version 428

java> float discountPercentage = 0.33f;
float discountPercentage = 0.33
java> double dollarToEuro = 0.80240500000000003d;
double dollarToEuro = 0.802405
java> long milliseconds = 253433535302857531;
long milliseconds = 25343353530285753
java>
```

Figure 4. Creating 3 variables to store decimal or long numbers

#### d. Boolean variables: to store true / false information

If you have never programmed before, this type of variable might be surprising to you. Why do we need a type of variable that would just store 2 possible values: true or false?

After all, we could just use a String type of variable for that:

Using a String variable to store a "true" value

```
String customerLoggedIn = "true";
String customerSubscribedToNewsletter= "false";
```

That would work very well, but in practice these true / false values are so common that a special type of variable called Boolean has been created just for them:

Using Boolean variables

```
Boolean hasSuperPowers = true;
Boolean isAClubMember = false;
```

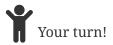
Let's notice that we did not put double quotes around the value true or false.

Just like before, there is another way to use these type of variables, again for a gain in memory:

Another way

```
boolean hasSuperPowers = true;
boolean isAClubMember = false;
```

Notice that Boolean has been replaced by boolean.



- 1. Create a variable called is Premium Member and store a true value in it.
- 2. Create a variable called isReturningVisitor and store a false value in it

Here is the solution:

```
Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_111 on Linux 3.13.0-49-generic Welcome to JavaREPL Web Console version 428

java> Boolean isPremiumMember = true;
java.lang.Boolean isPremiumMember = true
java> Boolean isReturningVisitor = false;
java.lang.Boolean isReturningVisitor = false
java>
```

Figure 5. Creating 3 variables to store true false values

### 3. Objects

We know how to create and store text, number and true / false values: these ingredients that are the basis of our mobile app.

But we surely need to create a store many thing else... like:

- some Forms, Buttons, Labels!
- a date (April, 10 2010)?
- colors, pictures, etc.

These type of variables are called **objects**. They are created and defined in the following way. Let's take the example of a Form:

#### a. Creating and defining objects

Creating and defining a Form

```
Form myForm; ①
myForm = new Form(); ②
```

- 1 We create the object (we declare it)
  - we define an object called "myForm".

- it specializes in storing Forms. Form was put in front of it to indicate this.
- at this stage the variable is empty (the technical term is null)
- ② We create and store a value in our object (we instantiate the object)
  - this line of code creates a new Form and stores it in our variable myForm.
  - the brackets () are empty, but not useless: because in some cases we can add a parameter inside them.
  - don't forget the capital F to Form, no capital to new, don't forget the () nor the ;

Just like for the variables we've seen above, we can take a shortcut: declare and instantiate an object in just one line:

Declaring and instantiating an object in one line

```
Form myForm = new Form();
```

#### b. Always the same logic: creating Buttons, Labels...

What we saw above is surely intimidating, but if you "get it", then you know how to create a huge variety of things.

(and we have a "cheatsheet" to help you memorize the essentials!)

For our mobile app, you can now create a Button:

Declaring and instantiating a Button

```
Button buttonCookieClicker = new Button();
```

Or a Label:

Declaring and instantiating a Label

```
Label welcomeMessage = new Label();
```

The Label is created in one line of code, but without text in it. That's a bit useless.

Remember when I mentioned that the empty () could sometimes include a parameter? Label is a good example:

Declaring and instantiating a Label

```
Label welcomeMessage = new Label("Welcome to my app!");
```

Now, if you read this lesson with care, this alternative could get you a "ahah":

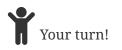
Declaring and instantiating a String, passing it to a Label

```
String welcomeMessage = "Welcome to my app!";
Label titleOfMyApp = new Label(welcomeMessage);
```

Let's now try to create an object in the interactive screen?

Form, Label and Button are objects provided by the plugin we installed in NetBeans, so they are not included in the interactive console here.

Instead, let's create another object. The object of type Date specializes in storing dates:



1. Create an object of type "Date" called dateOfSubscription and store a new Date in it.

Here is the solution:

```
Welcome to JavaREPL version null (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_131)
Type expression to evaluate, :help for more options or press tab to auto-complete.

java> Date dateOfSubscription = new Date();
java.util.Date dateOfSubscription = Sun Feb 11 19:55:39 CET 2018
java>
```

Figure 6. Declaring and instantiating an object of type Date

Look at the line in green: it gives the value stored in your variable dateOfSubscription: by default, it is the day and time when the variable is instantiated. Very convenient!

### The end

Questions? Want to open a discussion on this lesson? Visit the forum here (need a free Github account).

Find references for this lesson, and other lessons, here.

Licence: Creative Commons, Attribution 4.0 International (CC BY 4.0). You are free to:

- copy and redistribute the material in any medium or format
- Adapt remix, transform, and build upon the material
- ⇒ for any purpose, even commercially.



This course is designed by Clement Levallois.

Discover my other courses in data / tech for business: http://www.clementlevallois.net

Or get in touch via Twitter: @seinecle