

CODAPPS
Essential vocabulary and preliminary notions

Clément Levallois

2017-12-26

Table of Contents

1. A note on components and actions	1
a. Components	1
b. Actions	1
2. The difficulty of naming things	1
3. Two approaches to work on mobile apps	2
a. DIY or the Ikea way?	2
b. An example: creating a form by writing code, or with a GUI.	3
The end	5



1. A note on components and actions

a. Components

A mobile app can **include** many things. These are like the Lego blocks of our app:

- pictures
- buttons to press
- videos
- places where the user is supposed to write text
- a menu to navigate the app
- etc.



Things included in an app are called **Components**

b. Actions

And the app is also supposed to **do** many things, for example:

- open a web page when the user clicks a button
- change the picture when the user swaps left
- zoom in when the user double taps on the screen
- make Angry Bird fly when the user plays the game
- etc...



All an app can "do", we will call these **Actions**

2. The difficulty of naming things

When building software like mobile apps, it is important to give precise names to the tools we use.

Often, these names don't make intuitive sense, and this creates an impression of technical difficulty:

It is not to be annoying, but to make sure we actually don't confuse things and get the proper tool when we need it. Every domain of specialty, not just programmers, do this.

For instance, the screen of our mobile application is not simply called a screen... why?

1. To build a mobile app, the first Component we need is an empty region, the size of the screen of the phone, where we will place all other Components.
2. This Component is not called a **Screen**, why ??
3. Because **screen** is a name already taken, to name the phone's actual physical screen (the one that breaks when your phone falls on the ground)
4. So, what name should be given instead to the area an app takes on the screen of the phone: "ScreenApp"? "Region"? "FullSizeAppRegion"? "EmptyScreenOfTheAppWhereToPutThings"? None of this.

→ The designers of the tool chose the name **Form**.



A Form is the first Component we need when creating a mobile app.

It is the empty region of the app where we will place all other Components.

Because of this need for precise names, you will see that:

- adding text to the app, we will use a Component named **Label** (not Text)
- adding pictures to the app, we will use a Component named **Image** (not Picture)
- adding buttons to the app, we will use a **Button** Component (this one is intuitive!)

3. Two approaches to work on mobile apps

To sum up what we have seen so far:

- when we start designing an app, we need to create a **Form** first.
- Then we will place Components in the Form: Buttons, Labels, Images, etc.
- Then we can attach actions to each of these components (a "click action" on a button, etc)

How will you do that in practice? There are two ways.

a. DIY or the Ikea way?

Building a new mobile app is like building a new piece of furniture : you can build it yourself from scratch, or buy it at Ikea and just assemble it.

Codename One, the framework we use, gives you these two options: Do It Yourself (DYI) or Ikea.

- DYI: write the code to create everything in the app.
- IKEA: use a "**Graphical User Interface**" (GUI), which means you drag and drop things to build the app with a minimum of code to write.

There are different benefits to both approaches:

	Do It Yourself / writing code	Ikea approach / using a Graphical User Interface (GUI)
Benefits	Very flexible! You control every parameter since you write everything yourself	Quick and easy! You just drag and drop things, click and point with the mouse, no need to learn how to code.
Inconvenients	You need to learn how to code. Slow since you write everything yourself.	You get stuck at some point: to add features to your mobile app, not everything is in the GUI. Writing code is going to be necessary.

b. An example: creating a form by writing code, or with a GUI.

You write the code in NetBeans, in the files of your app. These files are visible here:

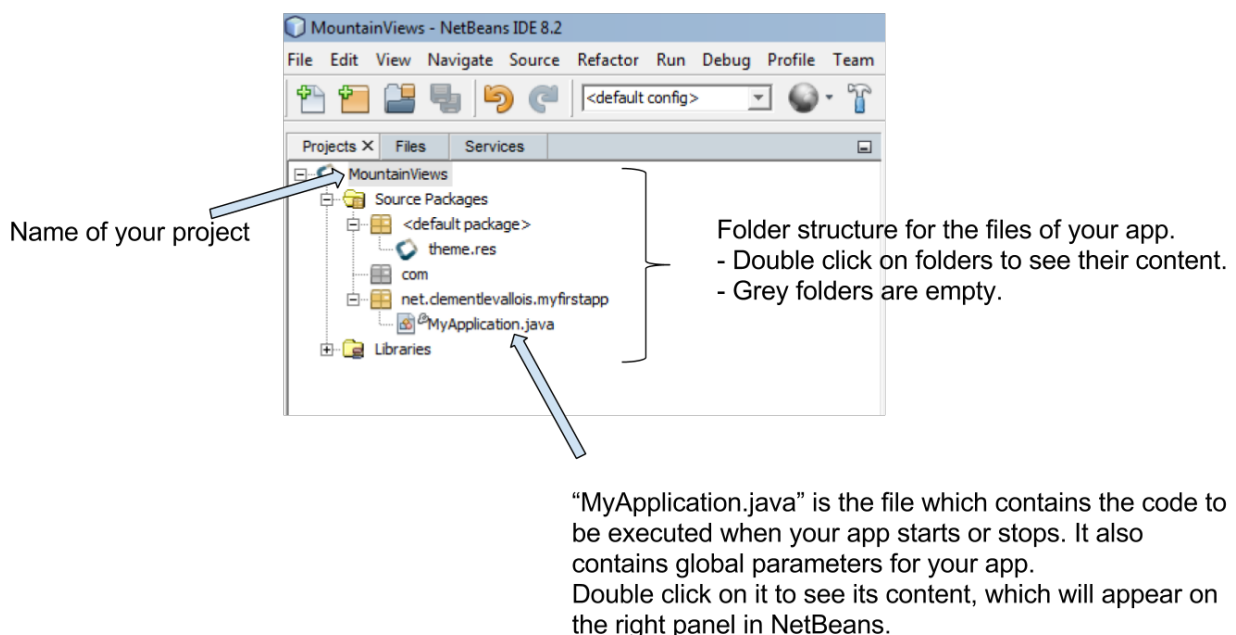


Figure 1. The files of your app visible in NetBeans

Double click on the file `MyApplication.java` to see its content on the right panel in NetBeans:

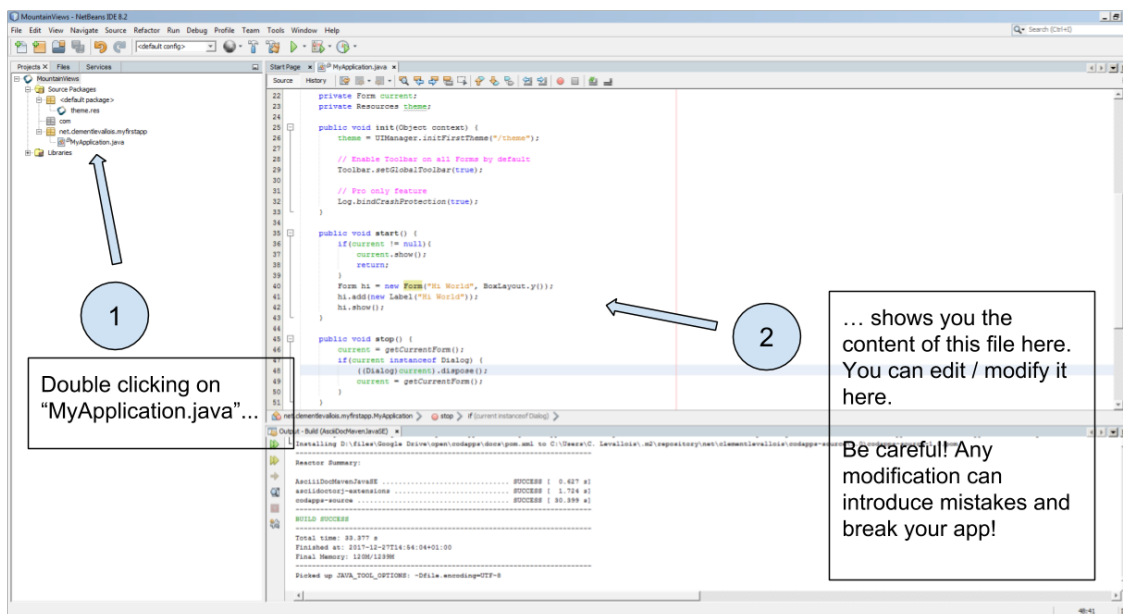


Figure 2. Opening the content of *MyApplication.java* for editing

When we created the project, an empty screen was added to it by default.

You can see this empty screen by previewing your app (click on the green arrow in NetBeans, or get back to the lesson [here](#))

How was this empty screen added to the app?

Simply with these 2 lines of code in the file *MyApplication.java* (scroll down a bit in NetBeans, as they are in the middle of the file):

MyApplication.java

```
public void start() {
    if(current != null){
        current.show();
        return;
    }
    Form hi = new Form("Hi World", BoxLayout.y()); ①
    hi.add(new Label("Hi World")); ②
    hi.show(); ③
}
```

- ① This line of code creates the screen with a title "Hi World" and an horizontal layout
- ② This line adds a piece of text saying ("Hi World"). It could have been removed.
- ③ This other line of code causes the screen to appear (otherwise it would remain hidden)

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](#)