CODAPPS

Making the cookie click

Clément Levallois

Version 1.0, last modified

Table of Contents

1. Storing the numbers about the price and productivity of cookie production tools

In the following, we are going to work on adding the auto-click, grandma, and farms tools in the game.

- the autoclick clicks on the cookie once every 10 seconds
- grandmas bake 1 cookie per second
- farms produce 3 cookies per second

Autoclick, grandmas and farms have a price: you need to spend your cookies to buy them. So if a grandma costs 100 cookies and your score is 100, you go back to zero cookie when you buy a grandma.

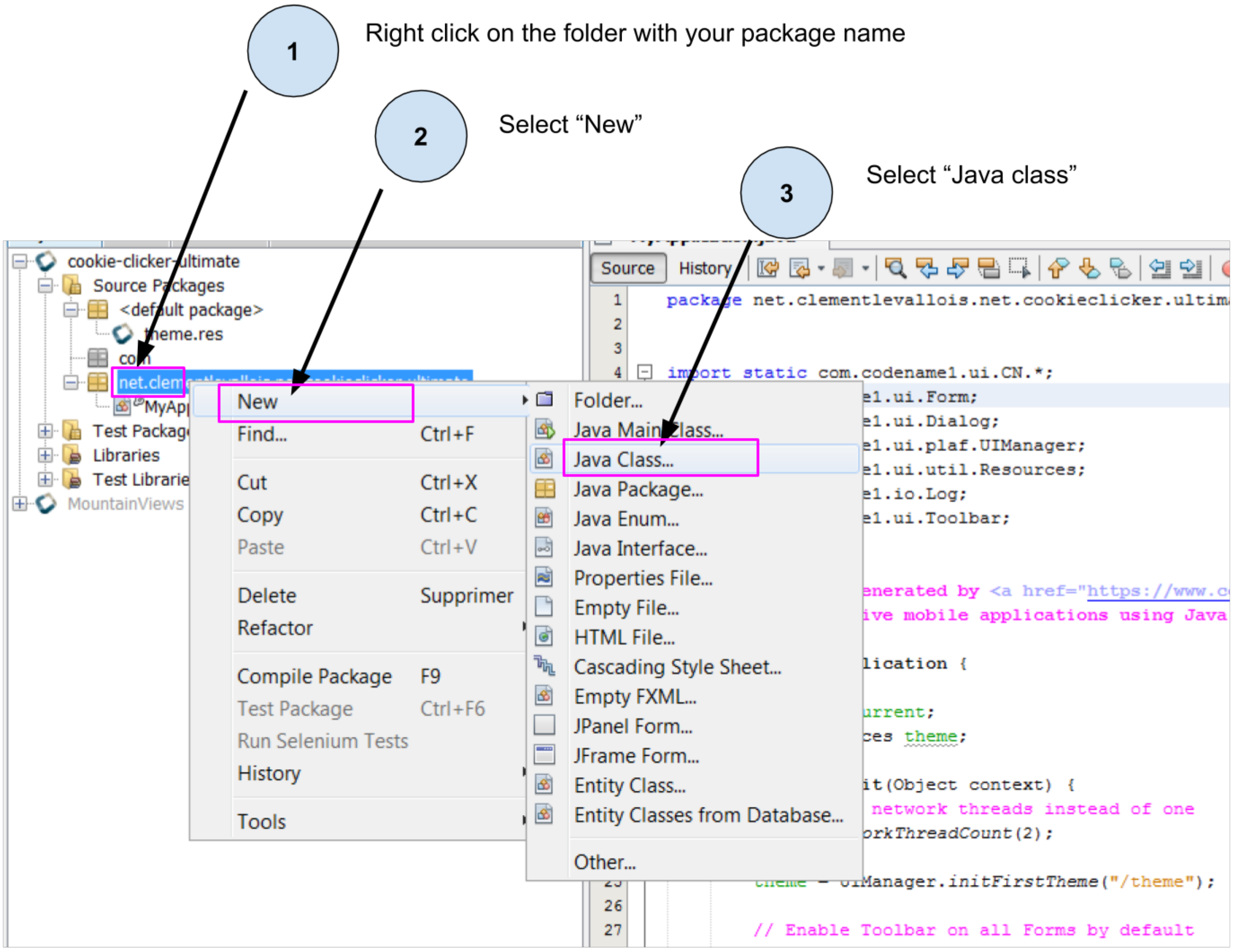
Also: **the price and productivity of these tools is evolving**: the more grandmas you buy, the more expensive they get. Each new grandma is also more productive than the preceding one you bought.

Same for auto-clicks and farms.

So that makes a long list of variables to track!

→ price and productivity for each tool, and number of tools owned.

We could create all these variables in our class `Form1`, but it will be more convenient to store all this in a separate class (a separate file). Let’s call it "Parameters" and create it:



1 Right click on the folder with your package name

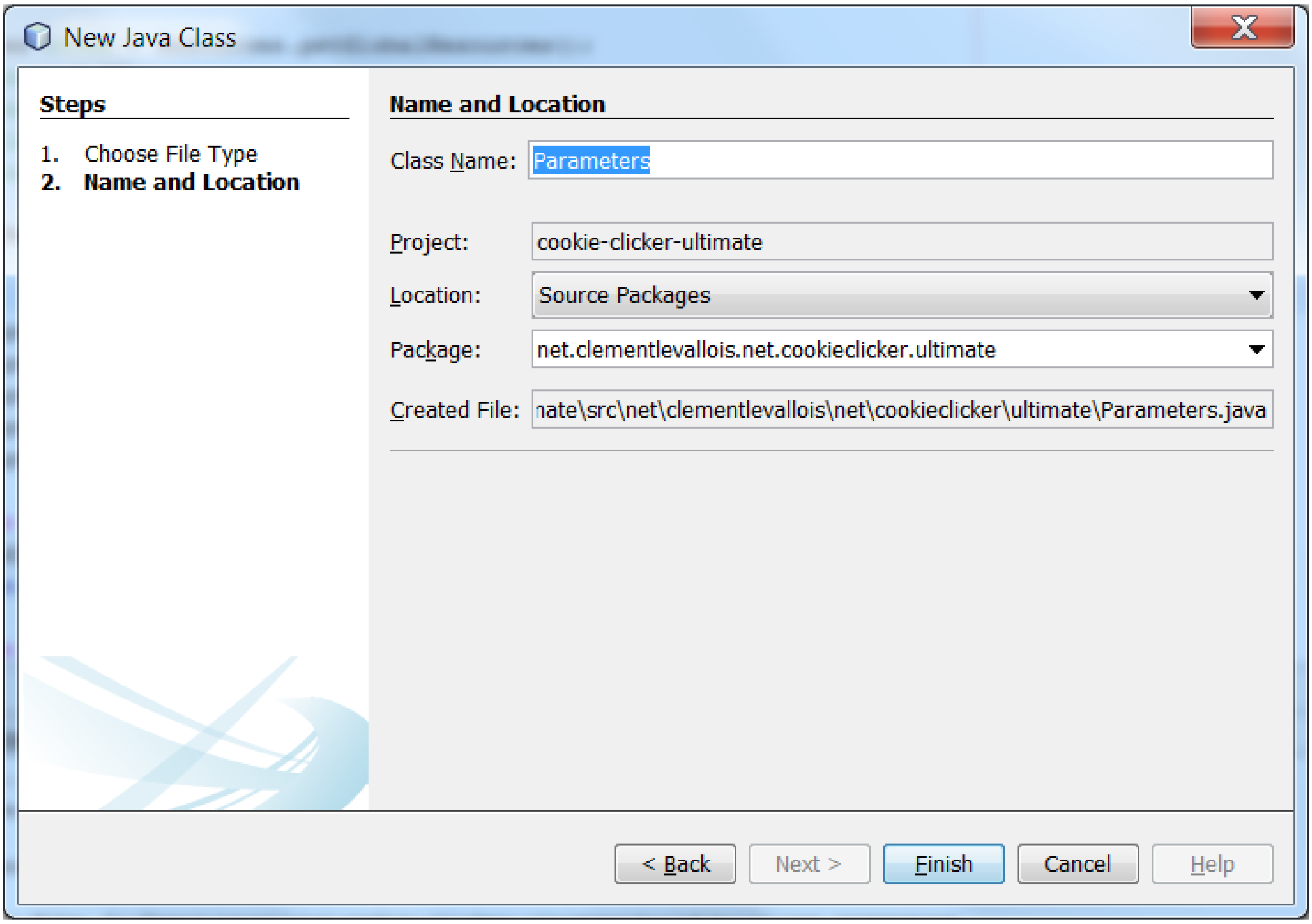
2 Select “New”

3 Select “Java class”

```
package net.clementlevallois.net.cookieclicker.ultim


import static com.codename1.ui.CN.*;
e1.ui.Form;
e1.ui.Dialog;
e1.ui.plaf.UIManager;
e1.ui.util.Resources;
e1.io.Log;
e1.ui.Toolbar;

enerated by <a href="https://www.c
ive mobile applications using Java

lication {

urrent;
ces theme;

it(Object context) {
 network threads instead of one
orkThreadCount(2);

theme = UIManager.initFirstTheme("/theme");

// Enable Toolbar on all Forms by default
```

Figure 1. Creating a Form with code

In the window which opens, leave every parameter unchanged **except for the <u>"Class Name"</u> where you write** `Parameters`, and click `Finish`



Name and Location

Class Name: Parameters

Project: cookie-clicker-ultimate

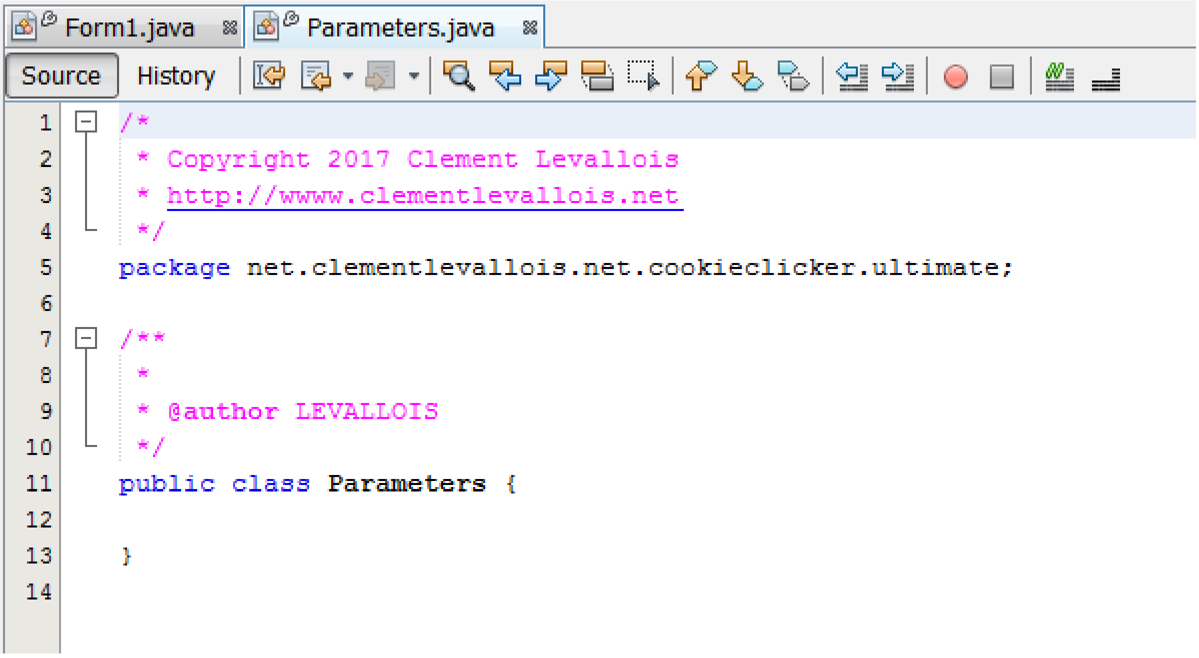
Location: Source Packages

Package: net.clementlevallois.net.cookieclicker.ultimate

Created File: nate\src\net\clementlevallois\net\cookieclicker\ultimate\Parameters.java

Figure 2. Creating a new class called Parameters

This creates an empty class like this one:



```
/*
 * Copyright 2017 Clement Levallois
 * http://wwww.clementlevallois.net
 */
package net.clementlevallois.net.cookieclicker.ultimate;

/**
 *
 * @author LEVALLOIS
 */
public class Parameters {

}
```

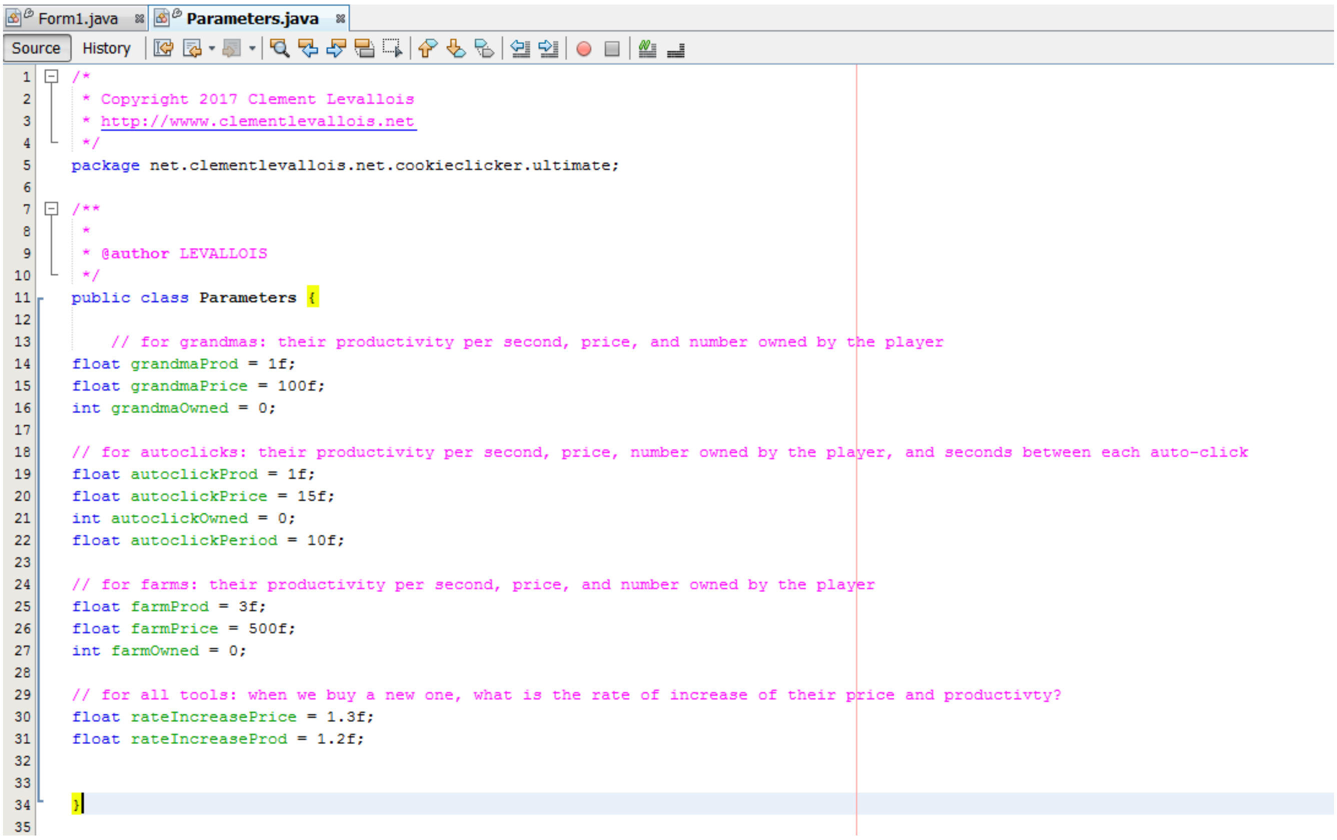
Figure 3. The class Parameters, still empty

In this class, we will define all the variables that will store the numbers we need:

Parameters for the cookie clicker app

```
// for grandmas: their productivity per second, price, and number owned by the player
float grandmaProd = 1f;
float grandmaPrice = 100f;
int grandmaOwned = 0;

// for autoclicks: their productivity per second, price, number owned by the player, and seconds between each auto-click
float autoclickProd = 1f;
float autoclickPrice = 15f;
int autoclickOwned = 0;
float autoclickPeriod = 10f;

// for farms: their productivity per second, price, and number owned by the player
float farmProd = 3f;
float farmPrice = 500f;
int farmOwned = 0;

// for all tools: when the player buys a new one, what is the rate of increase of their price and productivity?
float rateIncreasePrice = 1.3f;
float rateIncreaseProd = 1.2f;
```

The entire file will look like this:



```
/*
 * Copyright 2017 Clement Levallois
 * http://wwww.clementlevallois.net
 */
package net.clementlevallois.net.cookieclicker.ultimate;

/**
 *
 * @author LEVALLOIS
 */
public class Parameters {

        // for grandmas: their productivity per second, price, and number owned by the player
    float grandmaProd = 1f;
    float grandmaPrice = 100f;
    int grandmaOwned = 0;

    // for autoclicks: their productivity per second, price, number owned by the player, and seconds between each auto-click
    float autoclickProd = 1f;
    float autoclickPrice = 15f;
    int autoclickOwned = 0;
    float autoclickPeriod = 10f;

    // for farms: their productivity per second, price, and number owned by the player
    float farmProd = 3f;
    float farmPrice = 500f;
    int farmOwned = 0;

    // for all tools: when we buy a new one, what is the rate of increase of their price and productivty?
    float rateIncreasePrice = 1.3f;
    float rateIncreaseProd = 1.2f;


}
```

Figure 4. variables added to the class Parameters

Nice! How can our code in the file `Form1` use these variables? Very simply:

1. We go back to our class `Form1`
2. We define a variable `myParameters` of type `Parameters`, and we instantiate it.
3. That’s it. If we need to use the number of farms owned by the players, we will simply write `myParameters.farmOwned`. Done!

See below:

Using parameters in Form1

```
public class Form1 extends com.codename1.ui.Form { ①
    Resources theme = Resources.getGlobalResources(); ①
    float score = 0f; ①
    Parameters myParameters = new Parameters(); ②
```

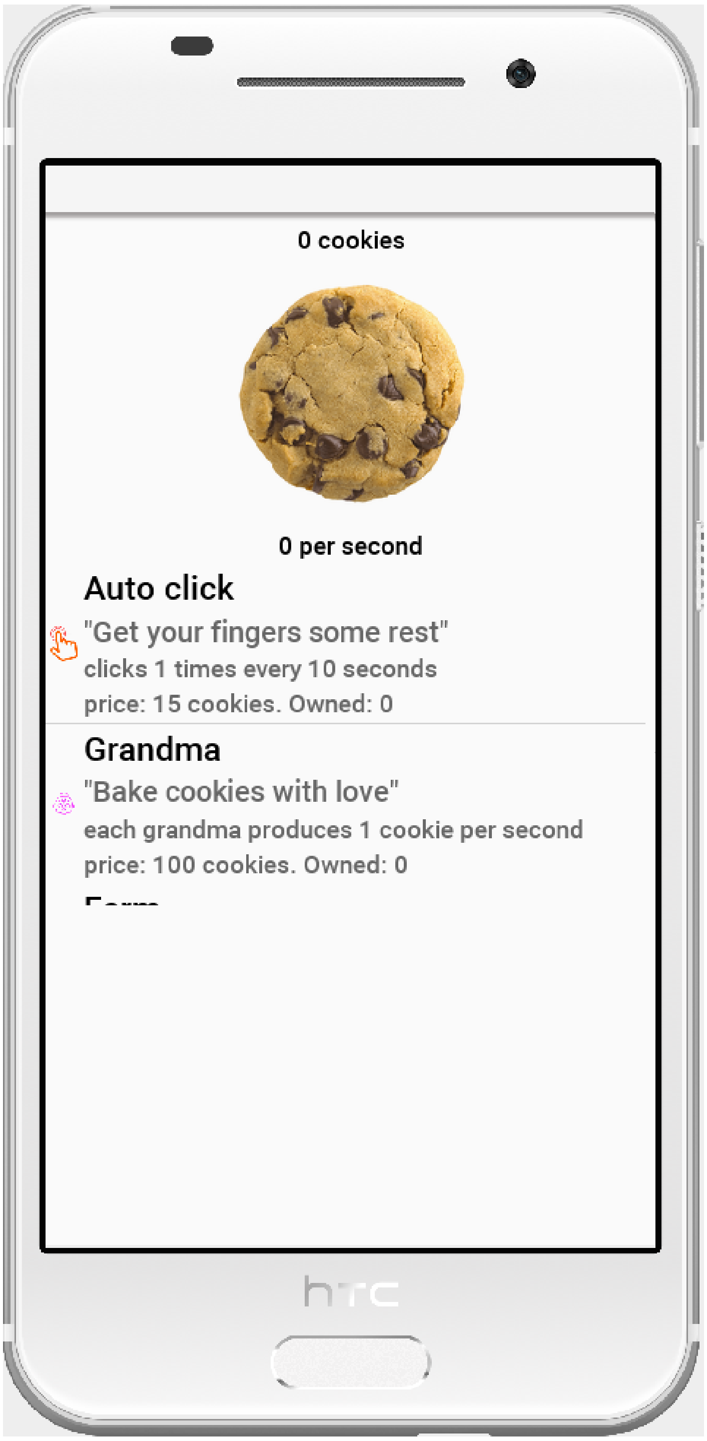
① the lines we have already written at the beginning of the file Form1

② * the line we add*

With all these parameters defined and available to use in `Form1`, we can now add some code to make our buttons click and do stuff:

2. Making the Auto click, the grandma and the farm buttons "clickable"

Just to recap, this is our app so far:



0 cookies

0 per second

Auto click

"Get your fingers some rest"

clicks 1 times every 10 seconds

price: 15 cookies. Owned: 0

Grandma

"Bake cookies with love"

each grandma produces 1 cookie per second

price: 100 cookies. Owned: 0

Farm

Figure 5. Our app so far

The player can click on the cookie and this will increase the score by one. **But when the player clicks on any of the tools (auto-click, grandma, farm), nothing happens**.

To make the cookie clickable, we had added an addActionListener method to the `ScaleImageButton` of the cookie. **We'll do the same for each of the tools.**

The difference is that tools have more features:

1. the player must have enough cookies to buy them.
2. since the player buys one tool, we must increase by one the number of this tool owned by the

player

3. the score of the player must be decreased when the tool is bought (if the player has 100 cookies and the grandma costs 100, the score must go to zero).
4. the price and productivity of **the next tool** that the player will purchase must increase
5. we must be update the label showing how many tools the player owns and the price of the next copy of the tool
6. we must update the label showing the score

Let’s do this for the Auto click tool first:

a. Making the AutoClick button work

All of this can be done with these lines of code:

Adding an action listener to the autoclick button

```
cookieCentral.addActionListener( ①
        (event) -> { ①
            score = score + 1; ①
            scoreLabel.setText(score + " cookies"); ①
        } ①
); ①

autoclickButton.addActionListener( ②
        (event) -> {
            if (score >= myParameters.autoclickPrice) {
                myParameters.autoclickOwned = myParameters.autoclickOwned + 1;
                score = score - myParameters.autoclickPrice;
                myParameters.autoclickProd = myParameters.autoclickProd *
myParameters.rateIncreaseProd;
                myParameters.autoclickPrice = myParameters.autoclickPrice *
myParameters.rateIncreasePrice;
                autoclickButton.setTextLine4("price: " + myParameters.autoclickPrice +
" cookies. Owned: " + myParameters.autoclickOwned + ".");
                scoreLabel.setText(score + " cookies");
            }
        }
); ②
```

① the code we had already written to make the cookie clickable, for reference (don’t write it again!)

② **all the lines we add to make the autoclick button working**

Notes: of course these lines are intimidating.

But have a careful look at them, taking note of where the curly braces open and close: you will see that they do exactly what the points 1. to 4. above have described:

Adding an action listener to the autoclick button - with comments

```
autoclickButton.addActionListener(
        (event) -> {
            if (score >= myParameters.autoclickPrice) {  ①
                myParameters.autoclickOwned = myParameters.autoclickOwned + 1;  ②
                score = score - myParameters.autoclickPrice;  // <3>                 }
                myParameters.autoclickProd = myParameters.autoclickProd *
myParameters.rateIncreaseProd;  ④
                myParameters.autoclickPrice = myParameters.autoclickPrice *
myParameters.rateIncreasePrice;  ④
                autoclickButton.setTextLine4("price: " + myParameters.autoclickPrice +
" cookies. Owned: " + myParameters.autoclickOwned + ".");  ⑤
                scoreLabel.setText(score + " cookies"); ⑥
            }
        }
);
```

① the player must have enough cookies to buy them. If not, the button does nothing (the lines of code following the { will not be executed).

② since the player buys one tool, we must increase by one the number of this tool owned by the player

③ the score of the player must be decreased when the tool is bought (if the player has 100 cookies and the grandma costs 100, the score must go to zero).

④ we increase the price and productivity of **the next tool** that the player will purchase

⑤ we must be update the label showing how many tools the player owns and the price of the next copy of the tool

⑥ we must update the label showing the score

Launch the preview, and test the app: the player can now buy an auto-clicker for the price of 15 cookies:

▶ https://www.youtube.com/watch?v=sjH6PZ5HcqQ *(YouTube video)*

Exactly the same logic applies for the grandma button and the farm button.

b. Making the Grandma and Farm buttons work

Just add these lines of code:

Adding an action listener to the grandma button and the farm button

```
grandmaButton.addActionListener(
        (event) -> {
            if (score >= myParameters.grandmaPrice) {
                myParameters.grandmaOwned = myParameters.grandmaOwned + 1;
                score = score - myParameters.grandmaPrice;
                myParameters.grandmaProd = myParameters.grandmaProd * myParameters
.rateIncreaseProd;
                myParameters.grandmaPrice = myParameters.grandmaPrice * myParameters
.rateIncreasePrice;
                grandmaButton.setTextLine4("price: " + myParameters.grandmaPrice + "
cookies. Owned: " + myParameters.grandmaOwned + ".");
                scoreLabel.setText(score + " cookies");
            }
        }
);

farmButton.addActionListener(
        (event) -> {
            if (score >= myParameters.farmPrice) {
                myParameters.farmOwned = myParameters.farmOwned + 1;
                score = score - myParameters.farmPrice;
                myParameters.farmProd = myParameters.farmProd * myParameters
.rateIncreaseProd;
                myParameters.farmPrice = myParameters.farmPrice * myParameters
.rateIncreasePrice;
                farmButton.setTextLine4("price: " + myParameters.farmPrice + "
cookies. Owned: " + myParameters.farmOwned + ".");
                scoreLabel.setText(score + " cookies");
            }
        }
);
```

c. The result so far

The result is a game app where the player can buy three tools to increase the cookie production - if they have enough cookies!

▶ https://www.youtube.com/watch?v=424CS-oUt38 *(YouTube video)*

Congratulations, all the buttons of your app are now working and "do something"!

In the next and last lesson, we will see how the score of the player can update on screen.

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.

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