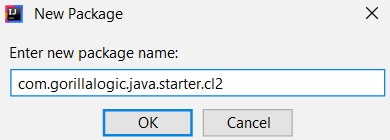
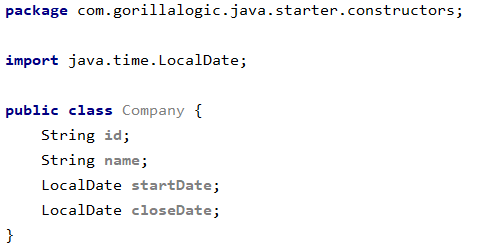
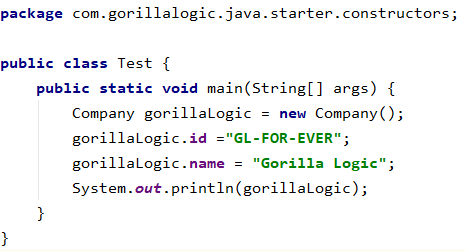
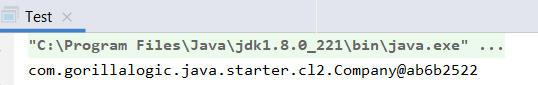
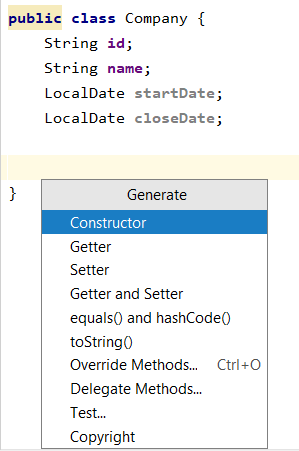
## What should I know?

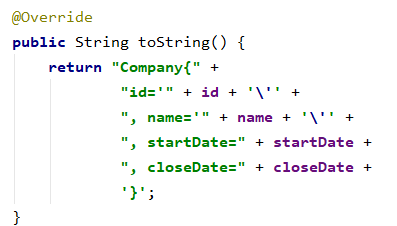
* Pre-requisites
  + CodeLab #1 completed

## Objectives

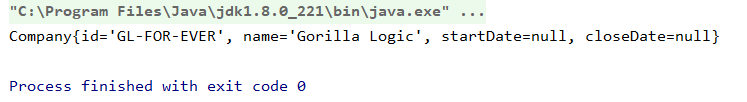
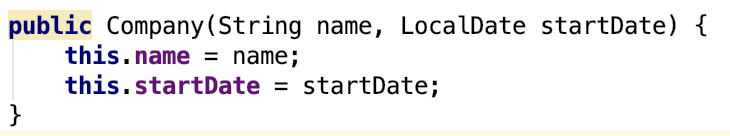
* Create a Java class with attributes
* Review default and constructor with arguments
* Review toString method
* Review equals and hashcode
* Review impact of the encapsulation: public, private and package.
* Review getters and setters.

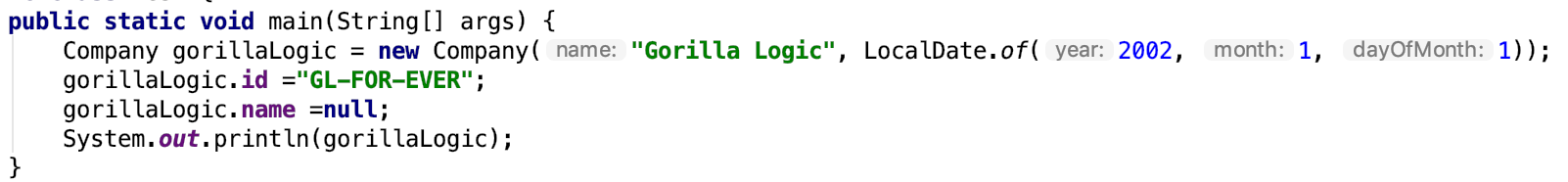
Steps:

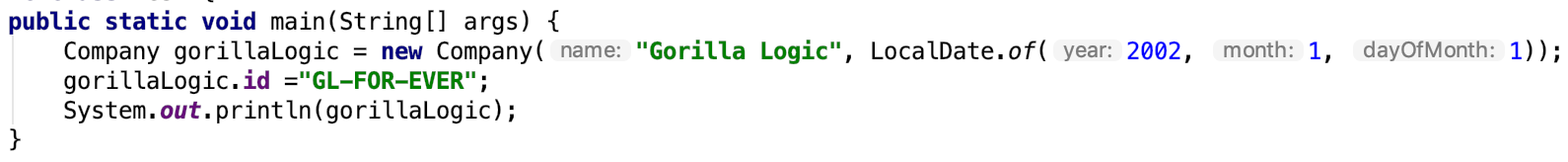
1. Open IntelliJ and the project created in the first CodeLab.
2. Create a new package:  
   
3. Create a new class inside this package called Company with some attributes:  
     
   **Note**: add the dependency to java.time.LocalDate in order to make it compilable.
4. Create a new class in the same package called Test and add to it the main method.
5. Inside the main method, create an instance of the Company class, set values for the attributes and print it using the System.out   
   
6. Run the Test class. The exit will be something like this:  
     
   As you can see the output shows the default toString result (a weird String, it has the full name of the class @ and the memory reference).
7. Let’s implement the toString method in the class, to do so press ⌘N and select toString method:  
   

Select the fields that you want to show of your class, the generated method should look like this:  
  
Consider that you can edit the toString as needed, the only recommendations are:

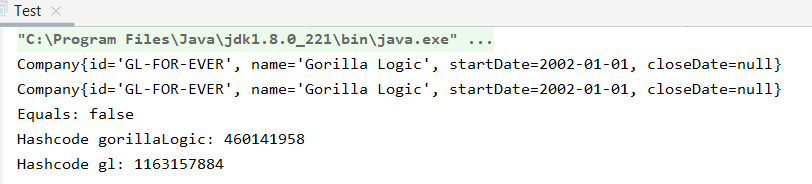
* Always show the class name.
* Try to be consistent between classes.

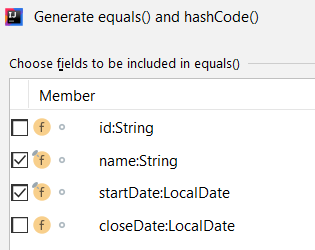
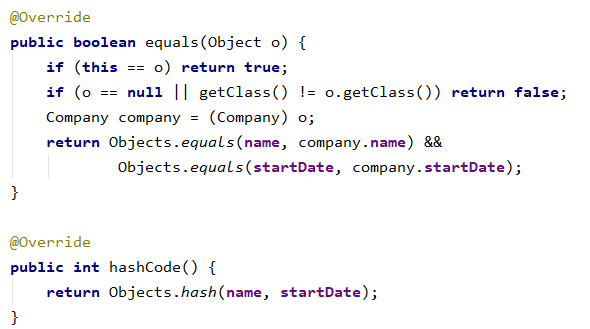
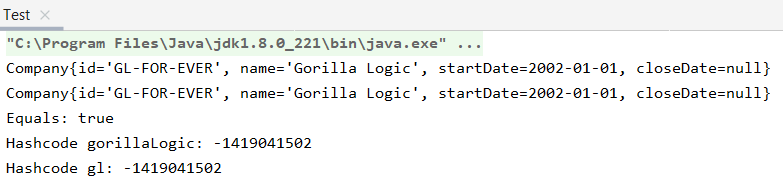
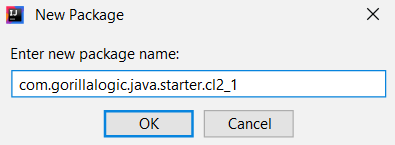
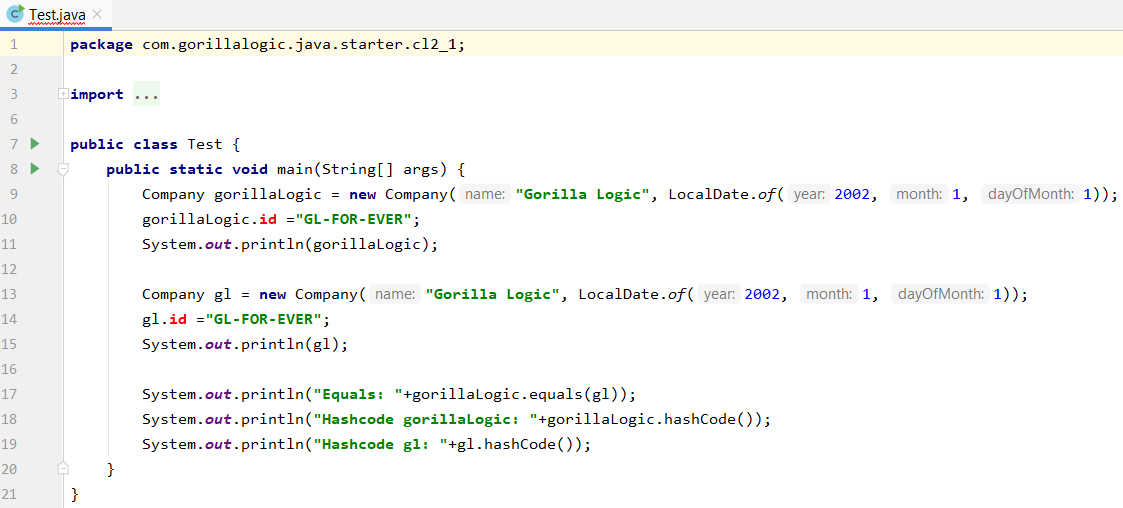
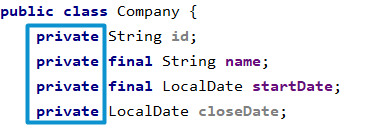
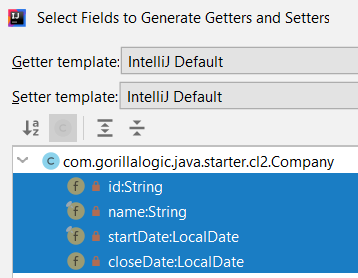
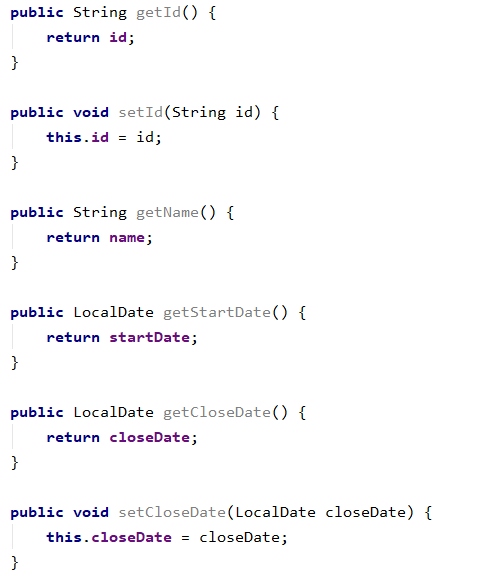
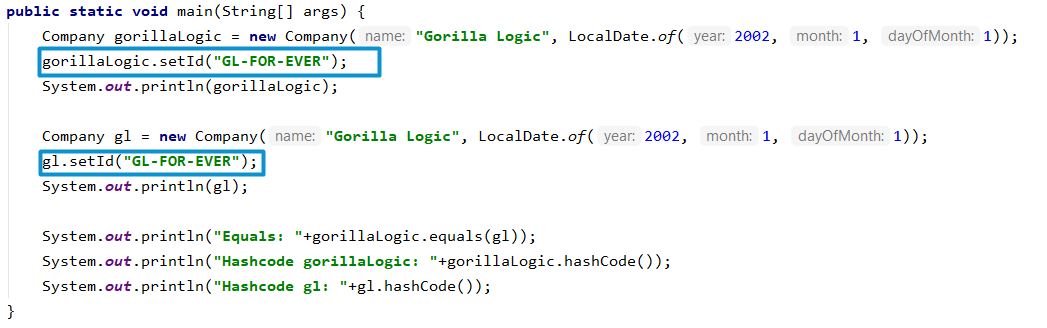
1. Now, let’s run again our Test class:  
     
   Now it has sense!!!
2. Now, let's imagine that we can’t have an enterprise without name and start date and those fields can’t be changed.
   1. Create a constructor with those fields:  
      
   2. Modify the Test class to make it compilable again.



* 1. Modify the Company test to ensure that we cannot modify the fields:  
     
  2. Modify the Test class to make it compilable again:  
     
  3. Now we need to edit the Test class because it is not compiling anymore.
  4. Now, even when we have forced to set the name and start date at the creation of the object we need to ensure that it cannot be edited after. We will do this setting those attributes as final:  
     

1. Now lets create a second company with the same data and print the comparison between both companies and the hashcode of both objects:  
   

The output will be false for the comparison:  


1. Let's imagine that we want to ensure that any companies that have the same start date and name are the same.
   1. Go to the Company class and press ⌘N and select equals() and hashcode() and select those fields:  
      
   2. Press next until finish:  
      
   3. Run again the Test class and check that now both companies are the same:  
      
2. Let’s create a new package at the same level that this package:  
   
3. Copy the test class to this package:  
     
   You will see that it doesn't compile, the reason is that the attribute id of the class Company has a default encapsulation that makes it accessible from the same package but no from others.
4. To solve this a possible solution would be to set it as public:  
     
   Just by doing this the new class will compile because now that attribute can be accessed from anywhere.
5. However, this is a bad practice and can lead us to further encapsulation problems, so the real solution is to make all the attributes private and create their getters and setters:
   1. Go to the company class and put all the attributes as private:  
      
   2. Now press ⌘N and generate Getters and Setters for those fields:  
        
        
      Note that no Setters are generated for the name and startDate, this is because both attributes are final and their values can’t be changed after initialization, this is also a reason to create make the attributes private and create getters and setters, in this way no one will try to set a second value on a final attribute and will have to face compilation problems by doing so.
   3. Modify both Test classes to use the setter of the id attribute.  
      
   4. Run the code and check that everything is working :)