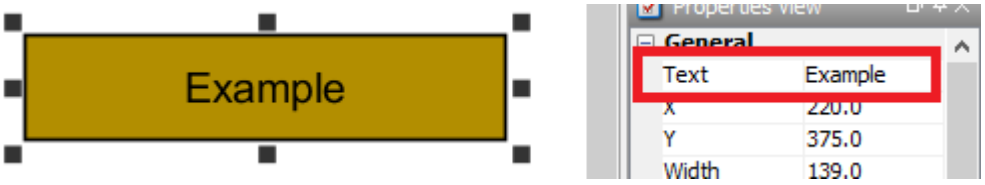
Exercise 1: Model a simple (very simple!) Drinking Machine on yEd Graph Editor.

For example, imagine this sequence. The user arrives and **chooses a drink**. Then he/she **inserts coins** until value inserted matches the price of the beverage. When this happens the machine **releases the requested product**, **calculates the change** and **returns it** to the user. Once the operation is over, the machine **remains on standby** until the next client arrives.

Every node and edge you create **must have** the text field (red square on the image below) with some value.



You must also respect the following rules (otherwise it won't work):

- **The model must have one, and only one, node with the text value "Start"!**
- **The Start vertex must have one, and only one, out-edge**;
- **The Start vertex must have no in-edges;**
- **The text field**(red square in the image above) **must not have whitespaces in it;**

Tip 1: Make a very simple model. If it's too complex, it will come to bite you later.

Tip 2: You just need to use nodes and edges.

Tip 3: Fill the text fields with meaningful values. Otherwise you won't know what means what, which spells trouble for the following exercises *wink* *wink*.

Exercise 2: Had variables and conditions to your model.

2.1: Create a variable to store the total value of the inserted coins.

2.2: You can insert 50 cents coins.

2.3: You can insert 20 cents coins.

2.4: "… **inserts coins** until value inserted matches the price of the beverage…", Make sure this happens. The drink costs 1.00 euro.

2.5: Once all drinks are sold, the machine turns off. Assume that the machine carries 20 drinks.

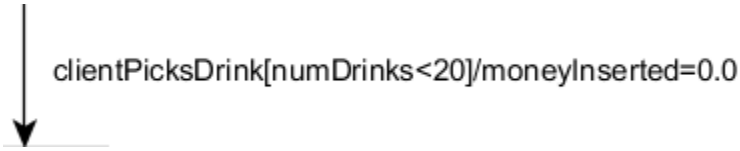
You must also respect the following rules (otherwise it won't work):

- **No whitespaces in conditions or variables;**
- **The conditions and variables are written like in Java;**
- **Conditions (also called Guards) and variables (also called actions) can should be created on the edges text field (see below how to).**

To define a variable or condition you use the following rule:

```
Label [Guard1;Guard2] / Action1;Action2;
```

Example:



Tip 1: If you couldn't finish the last exercise, use the provided solution!

Tip 2: Exercise 2.2 and 2.3 can be solved by adding a new edges. "insertCoin" from the provided solution seems to be a good edge to repeat ☺

Tip 3: Exercise 2.5 is mix of exercise 2.1 and 2.2.

Tip 4: Feel free to add edges and nodes.

Exercise 3: Convert your model to code!

This one is a little tricky so we'll give you a walkthrough:

1. If you couldn't finish the last exercise, use the provided solution!

2. Create a Folder.
   Name it and place it wherever you see fit. Keep in mind that you will have to access it through the console / terminal.

3. Inside the created folder paste the following files (they are in the attachments sent to you): graphwalker-2.6.3.jar, ModelAPI.template and the model you made in the last exercise;

4. Open the console/terminal and navigate to the folder you created on 2.

5. Execute the following command: "*java -jar graphwalker-2.6.3.jar source -f YOURGRAPHNAME.graphml -t ModelAPI.template*". Change YOURGRAPHNAME to the name of the model you placed in the folder. Example: "*java -jar graphwalker-2.6.3.jar source -f exercise.graphml -t ModelAPI.template*".

6. If Java functions were printed in the screen than your model is error-free and can be transformed. If you got an error, than your model is not transformable and must be corrected. The error might give you clue (by the way, the errors are from a Java compiler, so if you know and programmed in Java you are must certainly able to solved them).

7. If you were able to perform 5, it's time you save the conversion. To do this simply insert the following command "*java -jar graphwalker-2.6.3.jar source -f YOURGRAPHNAME.graphml -t ModelAPI.template > Exercise.java*". If the command was successful, you should have an extra file on your folder called "*Exercise.java*".

8. And it's done. Onward to the next and final exercise!

Exercise 4: Time to TQSO!

A little walkthrough before the exercises:

1. Create a Java project in Eclipse. Name it however you want.
2. Add JUnit 4 Library to your project.
3. Add “graphwalker-2.6.3.jar” to your project. To this simple choose “Add JARs…” on the same menu you added Junit 4. You don’t need to import it to the project.
4. Import the model from which you generated the code.
5. Import the generated code from the last exercise. You’ll probably need to create a package.
6. Correct any errors that show in the imported code. No errors should show up but if you gave your file a different name from the recommended in the last exercise some errors about class mismatching should show up.
7. Import “Test.java” into the same package. “Test.java” should be on the attachments given to you.
8. Uncomment one of the test and watch the results. Try to understand what the test does.
9. Try every test. Some may not work on your model.

Extra task: If you wish to experiment on another model you can import the “Login Example” project (a simulation of Spotify’s login) directly to eclipse and experiment with it.

Are you not satisfied? Do you crave more knowledge?

Even if you don't here are some links that you can consult:

- http://graphwalker.org/documentation/use-cases-or-how-tos (Use cases or How-tos, to learn more about the tool used today);
- http://graphwalker.org:8080/job/graphwalker/site/apidocs/index.html (GraphWalker API);
- http://graphwalker.org/ (GraphWalker homepage);

Thank you very much

Manuel Seixas, ei09011

Vítor Castro, ei09131