Design Patterns - TP5

https://github.com/IUT-Blagnac/cpoa-tp4-template

1. TP5 initial code

This is a template for the students' assignments.

Course material: 📱📱💻 http://bit.ly/jmb-cpoa

2. Assignment info

LAST NAME

BRUEL

First Name

Jean-Michel

Group #

[X] Teachers

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ Innopolis

3. Requirements

You'll need:

- ☑ A GitHub account
- ☐ A Git Bash terminal (if you use Window$)

Try the following command in your terminal to check your `git` environment:

```
git config --global -l
```

4. Initial tasks

- ☑ Click on the Github Classroom link provided by your teacher (in fact, this should be done if you

read this).

- ☐ Clone on your machine the Github project generated by Github Classroom.
- ☐ Modify the README file to add your last name, first name and group number.
- ☐ Update the badge URL so that it reflects your repo's result
- ☐ Commit and push using the following message:

commit/push

```
fix #0 Initial task done
```

In the following, every time you'll see à `fix #⋯` text, make sure all your files are committed, and then push your modifications in the distant repo, making sure you used the corresponding message (`fix #⋯`) in one of the `commit` messages.

- If you want to check that you're really ready for `fix #0`, you can run the command in your shell: `make check`.
- If you want to list the ToDos of the day, run `make todos`.

Initial code

1. Get the file: ObserverInitialCode.zip.
2. Read the content and compare the applications `observer.nonpattern` and `observer.pattern`.
3. Launch the applications (look for the 2 classes with `main()`) and observe (haha) the results.

Exercises

Reverse Engineering

- Obtain and compare the 2 class diagrams for the 2 packages ()
- Place them as `TP5.plantuml` and `TP5-observer.plantuml` in the `docs` folder

You should obtain something like that:

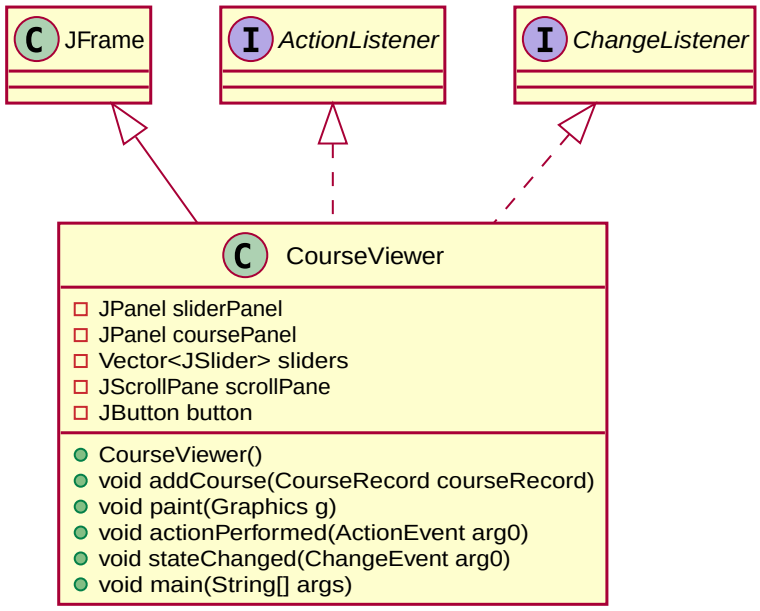

Figure 1. `observer.nonpattern`

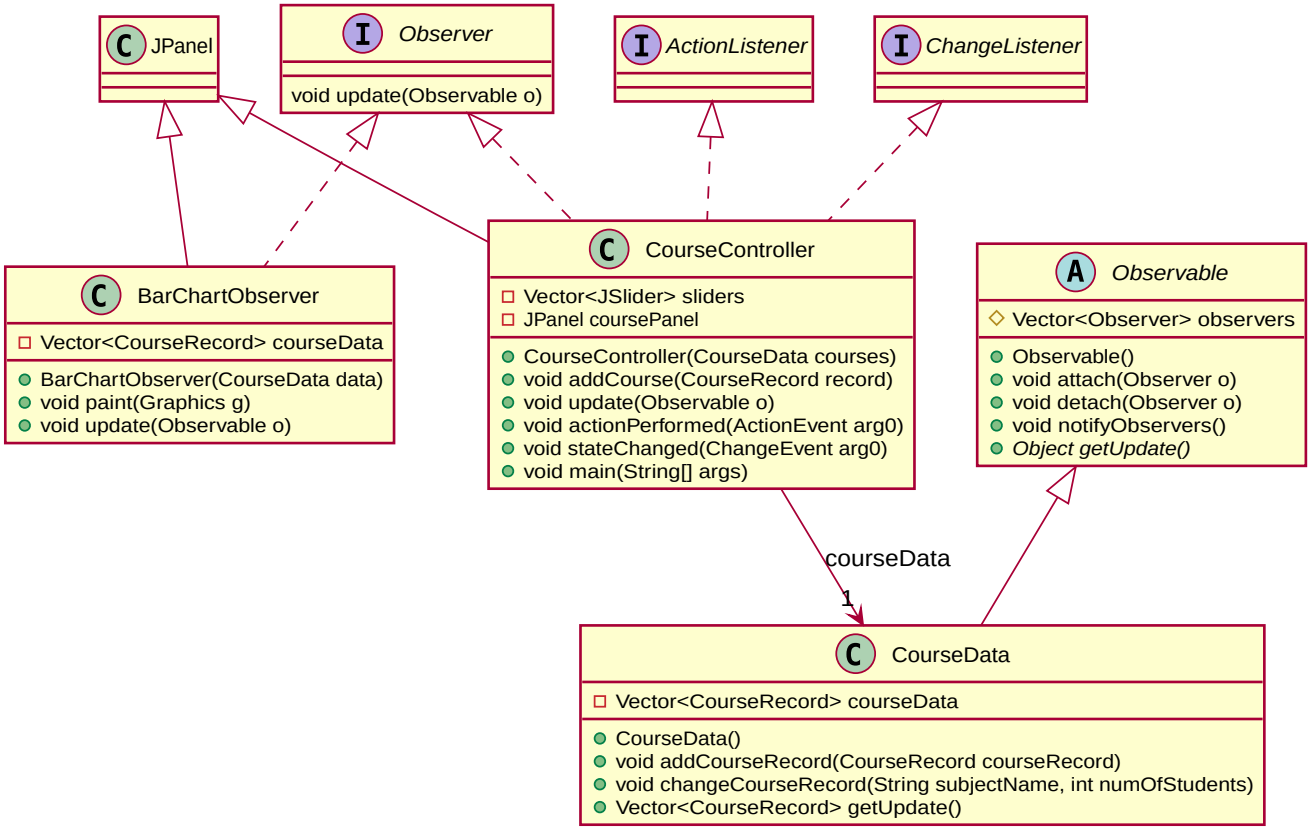

Figure 2. `observer.pattern`

commit/push

```
fix #1.1: Add Class Diagrams
```

`observer.nonpattern` code evolution

Upgrade the `observer.nonpattern` code to display the vector `CourseRecords` as a *pie chart*, in addition to the actual display (*bar chart*). For details on Swing pie charts, cf. final section.

commit/push

```
fix #1.2: Add Pie Chart in nonPattern
```

`observer.pattern` code evolution

Do the same with the `observer.pattern` code.

What do you think in terms of differences between the 2 (in terms of coding efforts and results)?

commit/push

```
fix #1.3: Add Pie Chart in Pattern
```

Push vs. Pull

You might have observed that the *Observer* pattern was using the *pull* method (`notifyObservers()` do not send any information).

Realize a `push` version.

commit/push

```
fix #1.4: Add push version
```

Problems with the *push*

In this new version, if the programs have more than 1000 courses, and if only one evolves, `notifyObservers ()` push all the information on all the observers!

Improve your *push* model so that it only pushes relevant data.

For this exercise, you can ignore the changes to `New Course` and continue to use the *pull* model for this type of change.

commit/push

```
fix #1.5: Change notify for smart push
```

Selecting the *updates*

You might have observed that `CourseController` is only interested in the changes of `New Course`, while `BarChart` and `PieChart` need to be aware of the `JSlider` changes. Extend the registration interface of `Observable` (the` attach` method) so that `CourseController` no longer receives updates from` updates` that are not of interest to it.

commit/push

```
fix #1.6: Add Smart attach()
```

How to draw a pie chart ?

Here is a code segment that draws a pie chart given a Graphics object and an Array containing Integers to be represented in the pie chart. It is drawn at location (xOffset, yOffset) and with the radius specified to be of size 100.

```
public void paint(Graphics g, Integer[] data) {
    super.paint(g);
    int radius = 100;

    //first compute the total number of students
    double total = 0.0;
    for (int i = 0; i < data.length; i++) {
        total += data[i];
    }
    //if total == 0 nothing to draw
    if (total != 0) {
        double startAngle = 0.0;
        for (int i = 0; i < data.length; i++) {
            double ratio = (data[i] / total) * 360.0;
            //draw the arc
            g.setColor(LayoutConstants.subjectColors[i%LayoutConstants.subjectColors
.length]);
            g.fillArc(LayoutConstants.xOffset, LayoutConstants.yOffset + 300, 2 *
radius, 2 * radius, (int) startAngle, (int) ratio);
            startAngle += ratio;
        }
    }
}
```

Contributors

- Jean-Michel Bruel

About...