SCALE FOR PROJECT PISCINE OBJECT - MC 04 - DESIGN PATTERN (/PROJECTS/PISCI OBJECT-MODULE-04-DESIGN-PATTERN

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.42angouleme.fr:vogsphere/intra-uuid-53a5671e-4dc3
```

Introduction

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well community depends on it.

- Identify with the person (or the group) evaluated the eventual dysfunctions of the work. Take the ti and debate the problems you have identified.

- You must consider that there might be some difference in how your peers might have understood instructions and the scope of its functionalities. Always keep an open mind and grade him/her as hc possible. The pedagogy is valid only and only if peer-evaluation is conducted seriously.

Guidelines

- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository belongs to the student or the group. Ensure that the work is project and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you and make you evaluate something content of the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have r possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for this stude entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, non-functioning program, a nor etc. In these cases, the grading is over and the final grade is 0 (or -42 in case of cheating).
However, with the exception of cheating, you are encouraged to continue to discuss your work (eve finished it) in order to identify any issues that may have caused this failure and avoid repeating the the future.

- Remember that for the duration of the defence, no segfault, no other unexpected, premature, unc unexpected termination of the program, else the final grade is 0. Use the appropriate flag.
You should never have to edit any file except the configuration file if it exists.
If you want to edit a file, take the time to explicit the reasons with the evaluated student and make s are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must be pr before the end of execution.
You are allowed to use any of the different tools available on the computer, such as leaks, valgrind, case of memory leaks, tick the appropriate flag.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/101072/en.subject.pdf)

datas.hpp (https://cdn.intra.42.fr/document/document/20078/datas.hpp)

Exercice 00 - Preparation

Structures organisation

This section isn't realy blocking nor give points for the module : it's only an introduction for the follov
The evaluated student must explain to you the choice that were made to dispach the structures into

Yes No

Exercice 01 - Singleton

Singleton class

In this section, the evaluated student must have created a singleton.hpp file.
The evaluator must check that this class indeed follow the singleton definition.

Definition:
Singleton is a creational design pattern that lets you ensure that an object has only one instance, w
global access point to this instance.

It should end up with a template class, allowing the user to create instances via a method and to ac
same method or via a second one.

It's up to your jugement to validate that what the evaluated student provided is indeed a singleton b

Warning :
As writed inside the subject, students aren't allow to repeat there code, so if you find the singleton c
multiples times, you may concidere give this exercice a 0 and stop the evaluation here.

Yes No

Lists creation

Using the Singleton base class, the evaluated student must have create four singletons lists :

- StudentList
- StaffList
- CourseList
- RoomList

The evaluated student must have provided a main.cpp to try the code.
It's time to edit the main, and try stuff !

If you find something that shouldn't be possible, you may concidere give this exercice a 0, and stop
here

Yes No

Exercice 02 - Factory and Command

Form classes

In this section, the evaluated student must have completed the four differents form of Form creatab
secretary.

If you do not find the 4 edited Form class, you may concidere give this exercice a 0 and stop the ev

Yes No

Form factory

In this section, the evaluated student must have created a Form Factory, in the form of the Secretar
As so, you must find a way to request creation of the four concrete Form classes to the Secretary c

If you do not find the way to create them thought the Secretary, you may concidere give this exercic
the evaluation here.

Yes No

Form completion

Now that the Forms are created, you must ensure that they do stuffs.

For this, the evaluated student must have provided a main.cpp, where you will be able to test the cl

Try to create forms, and try to execute them with and without having them signed by the head mast

If you find anything that isn't working as expected inside the subject, you may concidere give this ex stop the evaluation here.

Yes No

Exercice 03 - Mediator

Mediator creation

In this section, the evaluated student must have transform the Headmaster class into a mediator : it the central node of everything that happen in the program.

In this program, he's suppose to allow to centralize the event concerning students and professors. Every action must pass by him, and he will be able to perform action for everyone.

To verifiy that, you must check that :

- When given no course, a professor must ask a course creation form to the factory created ea
- When given no course, a student must ask a form to join course to the factory created earlier
- When a student finished it course, the course's professor must ask a graduation form to the f earlier.
- When given no classroom, a professor must ask for a classroom creation form to the factory

In those 4 cases scenarios, they must fullfill the form with necessary informations, and must return headmaster, to be signed and executed by it.

If any of those conditions aren't fullfilled, you may concidere give this exercice a 0 and stop the eva

Yes No

Headmaster sign and execute form

In this section, you will validate that the evaluated student have provided a way for the headmaster execute forms it may have received.

You must check that, when signing and execute forms, they do have the desired effect.

For this, the evaluated student must have provided a main.cpp, where you can check out that the h sign and execute forms.

Yes No

Exercice 04 - Observer

Observer creation

In this section, the evaluated student must have create, for the headmaster, a way to notify student that they have the opportunity to go into recreation.
The headmaster also have the opportunity to notify them that it's time to back to class.

Those interaction must be created by using a class, contained inside the Observer.hpp file.

You must verify that this interaction isn't made by the professors and students who will verify that th over, but by the bell indicating to every students and professors that they need to go back to class.
The interaction must also not be implemented by the headmaster iterating thought every professors and sending them back to class.

If any of those conditions aren't fullfilled, you may concidere give this exercice a 0 and stop the eva

Yes No

Exercice 05 - Facade

Facade creation

In this section, the evaluated student must have created a class School, containing everythings they previously created.

This class must contain the following methods, allow the users to control the school without having behind every action :

- void runDayRoutine() Execute the school day routine.
- void launchClasses() Make the headmaster ask the students and professors to attend them c
- void requestRingBell() Make the headmaster ring the bell
- void recruteProfessor() Add a new professor to the school
- void recruteStudent() Add a new student to the school
- Course* getCourse(std::string p_name) return a course with a given name
- std::vector<Student*> getStudents() return a vector containing every student in school
- std::vector<Professor*> getProfessors() return a vector containing every professor in school
- Headmaster getHeadmaster() return the headmaster of the school
- Secretary getSecretary() return the secretary of the school
- void graduationCeremony() launch the graduation of every student that can graduate at a de

You must check that you can indeed write a main that will control the school without have to create than the school inside the main.

If any of those conditions aren't fullfilled, you may concidere give this exercice a 0 and stop the eva

Yes No

Ratings

Don’t forget to check the flag corresponding to the defense

Ok Outstanding project

Empty work Incomplete work Invalid compilation Norme Cheat

Incomplete group Concerning situation Leaks Forbidd

Conclusion

Leave a comment on this evaluation (2048 chars max)

Finish evaluation