


(https://profile.intra.42.fr)

SCALE FOR PROJECT PISCINE OBJECT - MODULE 00 - ENCAPSULATION (/PROJECTS/PISCINE-OBJECT-MODULE-00-ENCAPSULATION)

You should evaluate 1 student in this team



Git repository

git@vogosphere.42angouleme.fr:vogosphere/intra-uuid-683fb585- 

Introduction

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well-being of the community depends on it.
- Identify with the person (or the group) evaluated the eventual dysfunctions of the work. Take the time to discuss and debate the problems you have identified.
- You must consider that there might be some difference in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade him/her as honestly as 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 for the relevant 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 other than the content of the official repository.
- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.
- If the evaluating student has not completed that particular project yet, it is mandatory for this student to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, non-functioning program, a norm error, cheating 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 (even if you have not finished it) in order to identify any issues that may have caused this failure and avoid repeating the same mistake in the future.

- Remember that for the duration of the defence, no segfault, no other unexpected, premature, uncontrolled or 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 sure both of you are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must be properly freed before the end of execution.

You are allowed to use any of the different tools available on the computer, such as leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

Attachments

 [subject.pdf \(https://cdn.intra.42.fr/pdf/pdf/97710/en.subject.pdf\)](https://cdn.intra.42.fr/pdf/pdf/97710/en.subject.pdf)

 [DivideAndRule.cpp \(https://cdn.intra.42.fr/document/document/19205/DivideAndRule.cpp\)](https://cdn.intra.42.fr/document/document/19205/DivideAndRule.cpp)

Exercice 00 - Divide and conquer

In this section, you should evaluate the student's understanding of encapsulation. Feel free to ask questions about the decisions the graded student made: he/she should be able to explain each one. You should find a main.cpp in the student repositories, so feel free to edit it to test out things you may want to test.

Structures composition and encapsulation

In this section, you should be interested in the composition of the Account and Bank classes.

The idea is to check if there is any way for you to modify the internal data of these classes from outside, without going through the class itself.

Check, for example, that the following code does not compile:

```
Account tmpAccount = Account();
```

```
tmpAccount.value += 150;
```

Don't hesitate to modify the main in order to check that nothing illogical is possible.

If you manage to find a flaw in the reasoning of the corrected student, you can put 0 to this exercise, and stop the correction here

☒ Yes☐ No

Getter and Setter

In this section, you should be interested in the getter and setter of the corrected student's classes.

Did the student create any getters? any const getters? any setters? Can he justify to you if he did or not?

Be more interested in the reasons for the choice than in the choice itself. You should, at least, find the following things :

- a method returning an account (only allowed return type: `const Account&`) in the Bank class
- a method returning an ID in the Account class
- a method returning the value of the account in the Account class

☒ Yes☐ No

Account ID - Not duplicate

In this section, you must make sure that no two IDs are identical when the account is created.

This operation can be done directly in the Account class or in the Bank class, but the result must be the same: no duplicates must be allowed!

If you manage to produce a duplicate ID, by any means, you can put 0 to this exercise and stop the correction here.

☒ Yes☐ No

Bank - Bank charges

In this section, you have to check that the only way to give money to an account is through the bank, and that the bank takes 5% of the amount that should be given to the account.

If you manage to give money to an account without going through the bank, you can put 0 to this exercise, and stop the correction here.

Obviously, an Account created outside of the bank did no count: we only look at account created and/or added to the bank data base.

☒ Yes☐ No

Bank - Account creation

In this section, you must be able to request the creation of an account via the Bank class.

The created account must be retrievable, either by having directly retrieved a `const Account&` at the exit of the method having created the account, or by having retrieved a long representing the ID of the account, at the exit of this same method.

If you have no way to know the newly created account, then you can put 0 to this exercise, and stop the exercise here

☒ Yes☐ No

Bank - Obtaining a loan

In this section, you should be able to request the bank to provide a loan to an account, via the account ID.

If you have a way to give a loan from outside the class, or if there is no way to make the bank give such a loan to a specific account, then you can put 0 to this exercise, and stop the correction here.

☒ Yes☐ No

Exercise 01 - I don't know what i'm doing !

In this part, you will have to deal with the choices of the corrected student about the creation of a Graph structure and a Vector2 structure. Unlike the previous exercise, here the student has more freedom to create. The Vector2 class could be a simple structure, with these two attributes in public, if the student is able to explain the reason other than "because it's more convenient to have everything in public": All these decisions must make sense. You might not agree with him, but as long as he has a reasoning, you are not allowed to give him 0: This mark should only be given in case of lack of reason to a decision. In short, be flexible.

Structures composition and encapsulation

You should find at least two classes in the exercise proposed by the corrected student : A Vector2 class (or structure) and a Graph class (Or structure).

These structures must have attributes, and these attributes must have an encapsulation reflected by the student: Do not hesitate to ask why such or such attributes are private or public. Why this or that getter exists or conversely why it does not exist.

Make sure that the graded student has thought through everything they have to create in the exercise, and that nothing has been left to chance or guesswork.

☒ Yes☐ No

Graph - storing Vector2

In this part, you will have to check that we can add in the Graph class a Vector2, representing a point in the 2D space of the graph.

You must be able to retrieve this vector in some way, via an operator or via a method, either via a const Vector2&, or a Vector2&, or a simple Vector2.

Don't hesitate to ask why the student chose this or that type of return.

☒ Yes☐ No

Graph - Output

In this part, you will have to check that the content of the table can be displayed in console. The latter can be re-scale to fit in a predefined line size or set by an attribute in the class, or not.

The important thing is to be able to have a visual feedback on the Vector2 that have been given to the graph.

If you have no way of displaying on screen the content of the graph, or if the points in the graph are not the ones given to it, or if some are missing, you can put 0 to this exercise, and stop the correction here.

✓ Yes

✗ No

Bonuses

Additional requirement

You must add one point to the slider for each additional requirement completed by the student



Rate it from 0 (failed) through 5 (excellent)

Ratings

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

✓ Ok

★ Outstanding project

Empty work



Incomplete work



Invalid compilation



Norme



Cheat



Crash



Incomplete group



Concerning situation



Leaks



Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation

Declaration on the use of cookies (<https://profile.intra.42.fr/legal/terms/2>)

Privacy policy (<https://profile.intra.42.fr/legal/terms/5>)

General term of use of the site (<https://profile.intra.42.fr/legal/terms/6>)

Règlement Intérieur (<https://profile.intra.42.fr/legal/terms/9>)

Terms of use for video surveillance (<https://profile.intra.42.fr/legal/terms/1>)

Legal notices (<https://profile.intra.42.fr/legal/terms/3>)