SCALE FOR PROJECT PISCINE CPP (/PROJECTS/PISCINE-CPP) / DAY 05 (/PROJECTS/42-PISCINE-C-FORMATION-PISCINE-CPP-DAY-05)

Introduction

The subject of this project is rather vague and leaves a lot to the user's choice. This is INTENDED. The questions in this grading scale, however, are very focused and concentrate on what we think is the core of each exercise, what we want you to grasp. So we would like you to do the same : You can and should tolerate moderate deviations in filenames, function names, etc ... as long as the exercise basically works as intended. Of course, in case the student you are grading really strayed too far, you should not grade the exercise in question at all. We leave it to your good judgement to determine what constitutes "straying too far".

The usual obvious rules apply : Only grade what's on the git repository of the student, don't be a dick, and basically be the grader you would like to have grading you.

Do NOT stop grading when an exercise is wrong.

Guidelines

You must compile with clang++, with -Wall -Wextra -Werror

Any of these means you must not grade the exercise in question:
- A function is implemented in a header (except in a template)
- A Makefile compiles without flags and/or with something other than clang++
- A non-interface class is not in Coplien's form

Any of these means that you must flag the project as Cheat:
- Use of a "C" function (*alloc, *printf, free)
- Use of a function not allowed in the subject
- Use of "using namespace" or "friend" (Unless explictly allowed in the subject)
- Use of an external library, or C++11 features (Unless explictly allowed in the subject)

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/2053/d05.en.pdf)

ex00

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise. If any non-interface class is not in Coplien's form, do not grade this exercise.

ex00

There is a Bureaucrat class
It has a constant name
It has a grade that ranges from 1 (Highest) to 150 (Lowest)
Exceptions are thrown when trying to create a Bureaucrat with a grade too high/low
There are getters for the attributes
There are functions to increment / decrement the grade, they throw exceptions when appropriate. Remember that incrementing a grade 3 gives you a grade 2 since 1 is the highest...
The exceptions used inherit from std::exception, or from something derived from std::exception (i.e. they are catchable as "std::exception & e")
There is a << operator to ostream overload that outputs the info of the Bureaucrat.

☑ Yes ☒ No

ex01

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise. If any non-interface class is not in Coplien's form, do not grade this exercise.

ex01

There is a Form class
It has a name, a bool that indicates whether is it signed (At the beginning it's not), a grade required to sign it, and a grade required to execute it.
The name and grades are constant.
All these attributes are private and not protected.
The grades have the same constraints as in the Bureaucrat (Exceptions, 1 = highest 150 = lowest, etc...)
There are getters for the attributes and a << operator to ostream overload that displays the complete state of the Form.
There is a Form::beSigned member function that works as described by the subject.
There is a Bureaucrat::signForm function that works as described by the subject.

☑ Yes ☒ No

ex02

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise. If any non-interface class is not in Coplien's form, do not grade this exercise.

ex02

There are concrete forms that are conform to the specifications of the subject (Required grades, names and actions).
They take only one parameter in their constructor, which is the target.
There is a Form::execute(Bureaucrat const & executor) method that works as specified by the subject.
Either this method is pure and the grade checks are implemented in each subclass, or this method does the checks then calls another method that only runs the action and is pure in the base class, both of these techniques are valid.
There is a Bureaucrat::executeForm(Form const & form) that works as specified by the subject.

☑ Yes ☒ No

ex03

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise. If any non-interface class is not in Coplien's form, do not grade this exercise.

Good dispatching

The makeForm function should really use some kind of array of pointers to member functions to handle the creation of Forms.
If it's using a worse method, like if/elseif/elseif/else branchings, or some other ugly stuff like this, please count this as Wrong.

☑ Yes ☒ No

ex03

There is an Intern class.
It has a makeForm function that works as specified by the subject.

☑ Yes ☒ No

ex04

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise. If any non-interface class is not in Coplien's form, do not grade this exercise.

ex04

There is an OfficeBlock class.

It has pointers to one Intern, one Bureaucrat that's the \"signing\" one, and one Bureaucrat that's the \"executing\" one.

It can be constructed either with all three or with nothing.
It has functions to set a new intern or new bureaucrats.
It has a doBureaucracy function that works as specified by the subject.
If the three members are not all set, the doBureaucracy function can not work.

☑ Yes ☒ No

Good exceptions

Rate the specificity of the exceptions thrown when using doBureaucracy here.
0 when there are no exceptions at all, 5 when there is one exception class by error type.

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

---

ex05

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise. If any non-interface class is not in Coplien's form, do not grade this exercise.

---

ex05

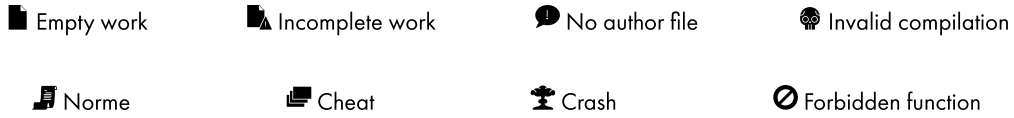
There is a CentralBureaucracy class.
It has 20 office blocks.
It can be created without parameters.
You can "feed" Bureaucrats to it, and they are used to fill the office blocks. If all the blocks are filled, new Bureaucrats are either rejected or stored in a waiting list of some sort.
Interns required to fill the blocks are generated automatically.
It is possible to queue target names in the object.
There is a doBureaucracy function that does some random bureaucracy to each target that was queued up, using the officeblocks it has created.

☑ Yes ☒ No

Ratings

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

✔ Ok



Conclusion

Leave a comment on this evaluation

Preview!!!