CPP_05

Exercise 00: Mommy, when I grow up, I want to be a bureaucrat!

Мамочка, когда я вырасту, я хочу быть бюрократом!

	Exercise : 00
Mommy, when I grow up, I want to be a bureaucrat!	
Turn-in directory : ex00/	
Files to turn in : Makefile, main.cpp, Bureaucrat.{h, hpp}, Bureaucrat.cpp	
Forbidden functions : None	

Пожалуйста, обратите внимание, что классы исключений не обязательно должны быть разработаны в ортодоксальной канонической форме.
Но все остальные классы должны это делать.

Давайте создадим искусственный кошмар из офисов, коридоров, бланков и очередей ожидания. Звучит забавно? Нет? Слишком плохой.

Во-первых, начните с самого маленького винтика в этой огромной бюрократической машине: the **Bureaucrat**.
Бюрократ должен иметь:
• Постоянное имя (constant name).
• И **grade - уровень (оценка, ранг)** в диапазоне от **1** (максимально возможная оценка) до **150** (минимально возможная оценка).
Любая попытка создать экземпляр бюрократа с использованием недопустимой оценки должна вызывать **исключение**:
либо **Bureaucrat::GradeTooHighException** либо **Bureaucrat::GradeTooLowException**.

Вы предоставите средства получения для обоих этих атрибутов: **getName()** и **getGrade()**.

Реализуйте также **две функции-члена для увеличения или уменьшения ранга бюрократа**. Если класс находится вне диапазона, оба они будут выдавать те же исключения, что и конструктор.

Примечание: Помните. Поскольку 1-й класс является самым высоким, а 150 - самым низким, увеличение 3-го класса должно дать бюрократу 2-й класс.

Создаваемые исключения должны быть уловимы с помощью блоков try и catch:

```
try
{
}
catch (std::exception & e) {
  /* handle exception */ (обработка исключения)
}
```

Вы реализуете перегрузку оператора insertion («), чтобы напечатать что-то вроде (без угловых скобок):

<name>, bureaucrat grade <grade>. // пример: Bob, bureaucrat grade 12

Как обычно, проведите несколько тестов, чтобы доказать, что все работает так, как ожидалось.

Exercise 01: Form up, maggots! (Постройтесь, черви или личинки!)

Form up, maggots!	
Turn-in directory : ex01/	
Files to turn in : Files from previous exercise + Form.{h, hpp}, Form.cpp	
Forbidden functions : None	

Теперь, когда у вас есть бюрократы, давайте дадим им какое-нибудь занятие. Что может быть лучше, чем заполнение стопки бланков?

Затем давайте создадим класс **Form**. Он имеет:

• Постоянное имя.

• Логическое значение, указывающее, подписано ли оно (при построении это не так - видимо при создании объекта значение = не подписано/false).

• Постоянный grade (уровень), необходимый для его подписания. (grade - уровень, оценка, ранг) (типа _signGrade)

• И постоянный grade (уровень), необходимый для его выполнения. (типа _execGrade)

Все эти атрибуты являются **private**, а не protected.

Уровни (grades) **Form** соответствуют тем же правилам, которые применяются к Bureaucrat.

Таким образом, следующие исключения будут выдаваться, если уровень (grade) формы находится за пределами границ:

Form::GradeTooHighException and Form::GradeTooLowException.

Как и раньше, **напишите геттеры для всех атрибутов и перегрузку оператора insertion («)**, который печатает всю информацию формы.

Добавьте также функцию-член **beSigned()** в форму (Form), которая принимает бюрократа (Bureaucrat) в качестве параметра.

Он изменяет статус формы на подписанный, если grade бюрократа достаточно высок (выше или равен требуемому).

Помните, что 1-й grade выше, чем 2-й.

Если уровень слишком низкий, бросьте исключение **Form::GradeTooLowException**.

Наконец, добавьте функцию-член **signForm()** к бюрократу. Если форма была подписана, она напечатает что-то вроде:

<bureaucrat> signed <form>

В противном случае он напечатает что-то вроде:

<bureaucrat> couldn't sign <form> because <reason>.

Внедрите и включите некоторые тесты, чтобы убедиться, что все работает так, как ожидалось.

Exercise 02: No, you need form 28B, not 28C...
(Нет, вам нужна форма 28B, а не 28C…)

No, you need form 28B, not 28C...
Turn-in directory : ex02/
Files to turn in : Files from previous exercises + ShrubberyCreationForm.[{h, hpp}.cpp], RobotomyRequestForm.[{h, hpp}.cpp], PresidentialPardonForm.[{h, hpp}.cpp]
Forbidden functions : None

Поскольку теперь у вас есть базовые формы, пришло время создать еще несколько, которые действительно что-то делают.
Во всех случаях форма базового класса должна быть абстрактным классом. Имейте в виду, что атрибуты формы должны оставаться закрытыми и что они находятся в базовом классе.

Добавьте следующие конкретные классы:
• **ShrubberyCreationForm**: (Форма создания кустарника) Обязательные grades: sign 145, exec 137
Создайте файл **<target>_shrubbery** в рабочем каталоге и запишите в него деревья ASCII.

• **RobotomyRequestForm**: (Форма запроса на роботехнику) Обязательные grades: sign 72, exec 45
Издает какие-то сверлящие звуки. Затем сообщает, что <target> была успешно роботозирована в 50% случаев. В противном случае сообщает, что роботомия не удалась.
"<target> has been robotomized successfully 50% of the time". Otherwise, informs that "the robotomy failed"

• **PresidentialPardonForm:** (Бланк президентского помилования) Обязательные grades: sign 25, exec 5
Сообщает, что <target> помилована Зафодом Библброксом.
"Informs that <target> has been pardoned by Zaphod Beeblebrox"

Все они принимают только один параметр в своем конструкторе: цель формы. Например, "home" (дом), если вы хотите посадить кустарник у себя дома.

Теперь добавьте функцию-член **execute(Bureaucrat const & executor) const** в базовую форму и реализуйте функцию для выполнения действия формы конкретных классов. Вы должны убедиться, что форма подписана и что grade бюрократа, пытающегося заполнить форму, достаточно высок. В противном случае выдайте соответствующее исключение.

Хотите ли вы проверить требования в каждом конкретном классе или в базовом классе (затем вызвать другую функцию для выполнения формы), зависит от вас. Однако один способ красивее другого.

Наконец, добавьте функцию-член **executeForm(Form const & form)** в Bureaucrat. Он должен попытаться выполнить форму. Если он пройдет успешно, выведите что-то вроде:
<bureaucrat> executed <form>
Если нет, выведите явное сообщение об ошибке.

Внедрите и включите некоторые тесты, чтобы убедиться, что все работает так, как ожидалось.