



C++ - モジュール 05

繰り返しと例外

概要

このドキュメントには、C++ モジュールのモジュール05の練習問題が含まれています。

バージョン: 10

内容

I	はじめに	2
II	一般規定	3
III	練習00: ママ、大きくなったら官僚になりたいの!	5
IV	練習01: 隊列を組め、ウジ虫ども!	7
V	練習02: いいえ、必要なのは28Cではなく28Bです...	9
VI	練習03: 少なくともコーヒーマーカーよりはましだ	11
VII	提出と相互評価	13

第一章 はじめに

C++は、*Bjarne Stroustrup*（ビャルネ・ストルストラップ）により、C言語の発展形として開発された汎用プログラミング言語である（出典：[Wikipedia](#)）。

これらのモジュールの目的は、**オブジェクト指向プログラミング**を紹介することです。これはあなたのC++の旅の出発点となるでしょう。OOPを学ぶには多くの言語が推奨されています。これは複雑な言語であり、物事をシンプルに保つために、あなたのコードはC++98標準に準拠することになります。

私たちは、現代のC++が多くの面で大きく異なっていることを認識しています。ですから、もしあなたが熟練したC++開発者になりたいのであれば、42のモン・コアの後にさらに進むのはあなた次第です！

第II章 一般規定

コンパイル

- `c++`と`-Wall -Wextra -Werror`フラグでコードをコンパイルする。
- フラグ`-std=c++98`を追加しても、コードはコンパイルできるはずだ。

フォーマットと命名規則

- 演習用のディレクトリは次のように命名されます: `ex00`, `ex01`, ..., `exn`.
`exn`
- ファイル名、クラス名、関数名、メンバ関数名、属性名は、ガイドラインの要求に従ってください。
- クラス名は**UpperCamelCase**形式で記述する。クラス・コードを含むファイルは常にクラス名に従って命名されます。例えば例えば、
`ClassName.hpp/ClassName.h`、`ClassName.cpp`、`ClassName.hpp`。例えば、
`ClassName.hpp/ClassName.hpp/ClassName.cpp/ClassName.hpp`のようになります。
- 特に指定がない限り、すべての出力メッセージは改行文字で終了し、標準出力に表示されなければならない。
- さようならノルミネットC++モジュールでは、コーディング・スタイルは強制されません。あなたの好きなスタイルに従ってください。しかし、相互評価者が理解できないコードは、評価できないコードであることを心に留めておいてください。きれいで読みやすいコードを書くよう、ベストを尽くしてください。

許可/禁止

あなたはもうC言語でコーディングしていない。C++の出番だ！ だから

- 標準ライブラリのほとんどすべてを使うことが許されている。したがって、すでに知っていることに固執するのではなく、使い慣れたC関数のC++っぽいバージョンをできるだけ使うのが賢いやり方だろう。
- ただし、それ以外の外部ライブラリーは使用できない。つまり、C++11（および派生形式）とBoostライブラリは禁止されている。以下の関数も禁止されている：`*printf()`、`*alloc()`、`free()`です。これらを使用した場合、あなたの成績は0点となり、それで終わりです。

- 明示的に指定されていない限り、using 名前空間 `<ns_name>` と友達キーワードは禁止。さもないければ、あなたの成績は-42点となる。
- **モジュール 08 と 09 でのみ STL を使用できます。**つまり、それまでは**コンテナ**（vector/list/map/など）や**アルゴリズム**（`<algorithm>`ヘッダーを含む必要があるもの）を使用しないこと。そうでなければ、成績は-42点となる。

いくつかの設計要件

- メモリ・リークはC++でも発生する。メモリを確保するときに（new キーワード）、**メモリー・リーク**を避けなければならない。
- モジュール02からモジュール09まで、**特に明記されている場合を除き**、あなたのクラスは**正教会正書式**でデザインされなければなりません。
- ヘッダー・ファイルに書かれた関数の実装は（関数テンプレートを除いて）、練習にとっては0を意味する。
- それぞれのヘッダーは、他のヘッダーから独立して使えるようにしなければならない。したがって、必要な依存関係をすべてインクルードしなければならない。しかし、**インクルード・ガード**を追加することで、二重インクルードの問題を避けなければなりません。そうでなければ、あなたの成績は0点となります。

続きを読む

- 必要であれば（コードを分割するためなど）ファイルを追加しても構いません。これらの課題はプログラムによって検証されるわけではないので、必須ファイルを提出する限り、自由に行ってください。
- 時には、練習のガイドラインが短く見えても、例題を見れば、指示には明示的に書かれていない要件がわかることもある。
- 始める前に各モジュールを完全に読むこと！ 本当にそうしてください。

- オーディンによって、ソーによって！頭を使い！




たくさんのクラスを実装しなければならない。これは、お気に入りのテキストエディタでスクリプトを書くことができない限り、退屈に思えるかもしれない。



練習をこなすにはある程度の自由が与えられている。しかし、義務的なルールは守り、怠けてはいけません。多くの有益な情報を見逃すことになります！理論的な概念を読むことをためらわないでください。

第三章

練習00： ママ、大きくなったら官僚になりたいの！

	エクササイズ： 00
ママ、大きくなったら官僚になりたいの！	
ターンイン・ディレクトリ： <i>ex00/</i>	
提出ファイル： <i>Makefile, main.cpp, Bureaucrat.{h, hpp}, Bureaucrat.cpp</i>	
禁止されている関数： なし	



なお、例外クラスは正統的なカノニカル・フォームでデザインする必要はない。
しかし、他のすべてのクラスはそうしなければならない。

オフィス、廊下、フォーム、待ち行列といった人工的な悪夢をデザインしよう。
面白そう？ ダメ？ 残念。

まず、この巨大な官僚機構の最も小さな歯車、**官僚から始めよう。官僚はこ**

うでなければならない：

- 一定の名前。
- そして、1（可能な限り高い評点）から**150**（可能な限り低い評点）までの評点。

無効なグレードを使用して官僚をインスタンス化しようとする、例外がスロー

される:

Bureaucrat::GradeTooHighExceptionまたはBureaucrat::GradeTooLowExceptionのいずれか。

これらの属性には、getName() と getGrade() というゲッターを用意します。また、官僚の等級を増減させる2つのメンバ関数も実装します。評点が範囲外の場合は、どちらもコンストラクタと同じ例外をスローします。



覚えておいてほしい。等級1が最高で150が最低なので、等級3を上げると官僚は等級2になるはずだ。

スローされた例外は、tryブロックとcatchブロックを使ってキャッチできなければならない：

```
試す
{
    /* 官僚と何かする */
}
キャッチ (std::exception & e)
{
    /* 例外処理 */
}
```


挿入(")演算子のオーバーロードを実装して、次のように表示する（角括弧は使わない）：

<名前>、官僚グレード<等級>。

いつものように、すべてが期待通りに機能することを証明するために、いくつかのテストを提出する。

第四章

練習01： 隊列を組め、ウジ虫ども！

	エクササイズ： 01
隊列を組め、ウジ虫ども！	
ターンイン・ディレクトリ： <i>ex01/</i>	
提出ファイル： 前回演習のファイル + Form.{h, hpp}, Form.cpp	
禁止されている関数： なし	

官僚がいるのだから、何か仕事を与えよう。書類の山に記入すること以上に良い活動があるだろうか？

では、**Form**クラスを作りましょう。これには

- 一定の名前。
- 符号付きかどうかを示すブール値（構築時は符号なし）。
- 署名には一定の成績が必要。
- そして、それを実行するために必要な一定の成績。

これらの属性はすべて**プライベート**であり、保護されているわけではない。

フォームの評点はビューロクラットに適用されるのと同じ規則に従う。したがって、フォームの評点が範囲外の場合、以下の例外がスローされます：

Form::GradeTooHighException と Form::GradeTooLowException です。

前と同じように、すべての属性のゲッターと、フォームのすべての情報を表示す

る挿入 (") 演算子のオーバーロードを書いてください。

また、官僚をパラメータとするbeSigned()メンバ関数をフォームに追加する。この関数は、官僚の等級が十分に高い（要求された等級よりも高いか、または同等である）場合に、フォームのステータスを署名付きに変更します。等級1は等級2より高いことを覚えておいてください。評点が低すぎる場合は、Form::GradeTooLowException をスローします。

最後に、ビューロクラットにsignForm()メンバ関数を追加する。フォームに署名があれば、次のように表示される：

<bureaucrat>は<form>に署名した。


そうでなければ、次のように表示される：

<官僚>は<理由>のために<フォーム>に署名できなかった。

テストを実施し、すべてが期待通りに機能することを確認する。

第五章

練習02：いいえ、必要なのは28Cではなく28Bです。

	エクササイズ：02
いいえ、必要なのは28Cではなく28Bです。	
ターンイン・ディレクトリ：ex02/	
提出ファイル： Makefile, main.cpp, Bureaucrat.{h, hpp}.cpp, Bureaucrat.cpp + AForm.{h, hpp}.cpp、ShrubberyCreationForm.{h, hpp}.cpp、 + RobotomyRequestForm.{h, hpp}.cpp、PresidentialPardonForm.{h, hpp}.cpp。	
禁止されている関数： なし	

基本的なフォームができたので、次は実際に何かをするフォームをいくつか作ってみよう。

どのような場合でも、基底クラス Form は抽象クラスでなければならないので、AForm という名前に変更しなければなりません。フォームの属性はプライベートのままである必要があり、それらは基底クラスにあることに留意してください。

以下の具象クラスを追加する：

- **ShrubberyCreationForm**： 必要な等級： 符号145, exec 137
作業ディレクトリに<target>_shrubberyファイルを作成し、その中にASCIIツリーを書き込む。

- **RobotomyRequestForm**: 必要な成績: サイン 72, exec 45

ドリルで穴をあける音がする。その後、<ターゲット>が50%の確率でロボットミーに成功したことを知らせる。そうでない場合は、ロボットミーが失敗したことを知らせる。

- **PresidentialPardonForm**: 必要なグレード: sign 25, exec 5 <

ターゲット>がザフォード・ビーブルブロックスによって恩赦されたことを伝える。

これらはすべて、コンストラクターに1つのパラメーターだけを取る。例えば、自宅に低木を植えたい場合は "home "となる。

ここで、`execute(Bureaucrat const & executor) const` メンバ関数を基本フォームに追加し、具象クラスのフォームのアクションを実行する関数を実装する。フォームが署名されていることと、フォームを実行しようとする官僚の等級が十分高いことをチェックしなければならない。そうでなければ、適切な例外を投げる。

すべての具象クラスで要件をチェックするか、ベース・クラスでチェックするか（それから別の関数を呼び出してフォームを実行するか）は、あなた次第だ。しかし、どちらか一方を選ぶ方が、きれいである。

最後に、`executeForm(Form const & form)` メンバ関数をビューロクラットに追加する。この関数はフォームの実行を試みます。成功したら、次のように表示します：


<bureaucrat>が<form>を実行した。

そうでない場合は、明示的なエラーメッセージを表示する。

テストを実施し、すべてが期待通りに機能することを確認する。

第六章

練習03：少なくともコーヒーメーカーよりはましだ

	練習：03
少なくともコーヒーを淹れるよりはましだ	
ターンイン・ディレクトリ：ex03/	
提出ファイル：提出ファイル：前回の演習で使ったファイル + Intern.{h, hpp}, Intern.cpp	
禁止されている関数：なし	

書類の記入だけでも十分面倒なのに、それを一日中官僚にやらせるのは残酷だ。幸い、インターンは存在する。この演習では、**インターン・クラス**を実装しなければならない。インターンには名前も成績も個性もない。官僚が気にかけるのは、彼らが仕事をする事だけだ。

しかし、インターンにはmakeForm()関数という重要な機能がある。この関数は2つの文字列を受け取ります。最初の文字列はフォームの名前で、2番目の文字列はフォームのターゲットです。パラメータとして渡された名前を持つ**フォームオブジェクトへのポインタ**を返し、そのターゲットは2番目のパラメータで初期化されます。

のように表示される：

インターンは<form>を作成する。

パラメータとして渡されたフォーム名が存在しない場合、明示的なエラー・メッセージを表示する。

if/elseif/elseフォレストを使うような、読みにくく醜い解決策は避けなければなりません。このようなものは評価プロセスで受け入れられません。あなたはもうPiscine（プール）にはいないのです。いつものように、すべてが期待通りに動くことをテストしなければならない。

例えば、以下のコードは "Ben-der " をターゲットとした **RobotomyRequestForm**を作成する：

```
{  
    Intern someRandomIntern;  
    Form*   rrf;  
  
    rrf = someRandomIntern.makeForm("robotomy request", "Bender");  
}
```


第七章

提出と相互評価

通常通り、Gitリポジトリに課題を提出してください。あなたのリポジトリ内の作品だけが、ディフェンス中に評価されます。フォルダ名やファイル名が正しいかどうか、ためらわずに再確認してください。



16D85ACC441674FBA2DF65190663F9373230CEAB1E4A0818611C0E39F5B26E4D774F1
74620A16827E1B16612137E59ECD492E468A92DCB17BF16988114B98587594D12810
E67D173222A

