



## C++ - モジュール 05

### 繰り返しと例外

#### 概要

本書は、C++モジュールのうち、Module 05の演習を収録したものです。

バージョン : 9

# 内容

I	はじめに	2
II	一般規定	3
III	運動00：ママ、大きくなったらお局様になりたいの！？	5
IV	練習問題01：隊列を組め、ウジ虫ども！？	7
V	Exercise 02: いいえ、28Cではなく28Bのフォームが必要です...	9
VI	練習問題03：少なくともコーヒーメーカーよりはましだ	11

# 第一章 はじめに

C++は、*Bjarne Stroustrup*が「C with Classes」(出典：[Wikipedia](#))というプログラミング言語の元祖として作った汎用プログラミング言語である。

これらのモジュールの目標は、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。オブジェクト指向を学ぶには、多くの言語が推奨されます。この言語は複雑な言語であり、物事を単純化するために、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++が多くの側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のコモンコアの後にさらに進むのはあなた次第なのです

## 第II章 総則

### コンパイル

- `c++`と`-Wall -Wextra -Werror`フラグでコンパイルしてください。
- フラグ `-std=c++98` を追加しても、あなたのコードはコンパイルできるはずです。

### フォーマットと命名規則

- 演習用のディレクトリは、`ex00`, `ex01`, ...のように命名されます。 `exn`
- ファイル、クラス、関数、メンバ関数、属性にガイドラインに必要な名前を付ける。
- クラス名は、**UpperCamelCase** 形式で記述します。クラスコードを含むファイルには、常にクラス名に従った名前が付けられます。例えば例えば、`ClassName.hpp/ClassName.h`, `ClassName.cpp`, または `ClassName.tpp`.例えば、レンガの壁を表すクラス "`BrickWall` "の定義を含むヘッダーファイルがあれば、そのファイル名は`BrickWall.hpp`となります。
- 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。あなたの好きなものに従えばいいのです。しかし、同僚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ために、最善を尽くしてください。

### 許可/禁止

あなたはもうCでコーディングしていない。C++の時代だ!というわけで。

- 標準ライブラリのほとんどすべてを使用することが許されています。したがって、すでに知っているものに固執するのではなく、使い慣れたC関数のC++的なバージョンをできるだけ使うのが賢い方法でしょう。
- ただし、それ以外の外部ライブラリは使用できません。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ます。以下の関数も禁止されています。`*printf()`、`*alloc()`、`free()`。これらを使用した場合、成績は0点、それで終わりです。

- 明示的に指定しない限り、`using`名前空間<ns\_name>と友人キーワードは禁止です。そうでない場合、あなたの成績は-42となります。
- **STL の使用はモジュール 08 のみ許可されています。**つまり、コンテナ (`vector/list/map/` など) とアルゴリズム (<`algorithm`> ヘッダを含む必要があるもの) はそれまで使わないでください、ということです。さもなければ、あなたの成績は -42 になります。

### いくつかのデザイン要件

- C++でもメモリリークは発生します。メモリを確保するときに (`new` キーワード)を使用する場合は、メモリリークを回避する必要があります。
- モジュール02からモジュール08までは、特に明記されている場合を除き、正教会の正書法で授業を設計する必要があります。
- 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）、演習では0を意味します。
- それぞれのヘッダーは、他のヘッダーと独立して使用できるようにする必要があります。従って、必要な依存関係はすべてインクルードしなければなりません。しかし、インクルードガードを追加することによって、二重インクルードの問題を回避し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。

### リードミー

- 必要であれば、いくつかのファイルを追加することができます（コードを分割するためなど）。これらの課題は、プログラムによる検証を行わないので、必須ファイルを提出する限り、自由に行ってください。
- 演習のガイドラインは短く見えても、例題には明示的に書かれていない要件が示されていることがあります。
- 始める前に、各モジュールを完全に読んでください本当に、そうしてください。
- オーディンによって、ツールによって!頭を使い!!!




多くのクラスを実装する必要があります。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る人でなければ、退屈に思えるかもしれません。



演習をこなすには、ある程度の自由度が与えられています。ただし、必須のルールは守り、怠慢は禁物です。多くの有益な情報を見逃すこととなりますよ。理論的な概念については、ためらわずに読んでください。

## 第三章

# 運動00：ママ、大きくなったらお局様になりたいの！？

	演習：00
ママ、大きくなったら、お局様になりたいの！」 。	
ターンインディレクトリ： <i>ex00/</i>	
提出するファイル: <i>Makefile, main.cpp, Bureaucrat.{h, hpp}, Bureaucrat.cpp</i> 。	
禁止されている機能: なし	



例外クラスは、正統派正典形式でデザインされる必要はないことに注意してください。しかし、他のすべてのクラスはそうしなければなりません。

オフィス、廊下、帳票、待ち行列など、人工的な悪夢をデザインしてみよう。  
楽しそう？ダメ？残念だな

まず、この巨大な官僚機構の最も小さな歯車である「**官僚**」から始めましょう。

官僚はこうでなければならない。

- 一定の名称。
- そして、**1**（最高点）から**150**（最低点）までの評点。

無効なグレードを使用して `Bureaucrat` をインスタンス化しようとする、例外がスローされる必要があります。

`Bureaucrat::GradeTooHighException` または `Bureaucrat::GradeTooLowException` のどちらかです。

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ための 2 つのメンバ関数も実装します。等級が範囲外である場合、それらの両方はコンストラクタと同じ例外を投げます。



覚えておいてください。 等級1が最高で、150が最低ですから、等級3をインクリメントすると、官僚は等級2になるはずです。

投げられた例外はtryブロックとcatchブロックを使ってキャッチ可能でなければなりません。

ここがける


```
{
    /* ビューロクラットで何かする */
}
catch (std::exception & e)
{
    /* 例外を処理する */
}
```

をす

いつものように、期待通りに動作することを証明するために、いくつかのテストを提出します。

## 第四章

### 練習問題01：隊列を組め！ウジ虫たちよ。

	演習：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>	
提出するファイル:前回演習のファイル+ <code>Form.{h, hpp}</code> , <code>Form.cpp</code>	
禁止されている機能:なし	

さて、官僚ができたからには、何か仕事を与えよう。書類の束に記入すること以上に良い活動があるだろうか？

では、**Form** クラスを作ってみましょう。これには

- 一定の名称。
- 符号化されているかどうかを示すブーリアン（構築時は符号化されていない）。
- サインに必要な定数グレード。
- そして、それを実行するために必要な定数

グレード。これらの属性はすべて私的なもの

であり、保護されているわけではありません

。

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ルールに従います。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記述してください。





`Bureaucrat` をパラメータとして受け取る `beSigned()` メンバ関数もフォームに追加する。これは、ビュロクラットの等級が十分に高い (必要な等級より高い、または同等) 場合に、フォームの状態を署名済みに変更します。等級 1 は等級 2 よりも高いことを忘れないように。

評点が低すぎる場合、`Form::GradeTooLowException` を投げる。

最後に、`Bureaucrat` に `signForm()` メンバ関数を追加してください。フォームに署名があれば、次のように表示されます。

<bureaucrat>のサイン入り<form>。


そうでない場合は、次のように表示されます。

<bureaucrat>が<form>にサインできなかったのは<reason>のせいだ。

期待通りに動作することを確認するために、いくつかのテストを実施し、提出する。

## 第五章

### Exercise 02: いいえ、28Cではなく、28Bのフォームが必要です...

	演習 : 02
いいえ、28Cではなく、28Bが必要です...	
ターンインディレクトリ : <i>ex02/</i>	
提出するファイル:過去の演習のファイル+ $\alpha$ ShrubberyCreationForm.[{h, hpp},cpp], RobotomyRequestForm.[{h, hpp},cpp], PresidentialPardonForm.[{h, hpp},cpp]。	
禁止されている機能 :なし	

基本的な形ができたので、次は実際に何かをするための形をいくつか作ってみましょう。

いずれの場合も、基底クラスである **Form** は抽象クラスである必要があります。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を念頭に置いてください。

以下の具象クラスを追加します。

- **ShrubberyCreationForm**。必要なグレード : 符号145、実行137  
作業ディレクトリに&lttarget>\_shrubberyというファイルを作成し、その中にASCIIツリーを書き込む。
- **RobotomyRequestForm**です。必要な成績 : 符号72、実行45  
ドリルで穴を開ける音がする。その後、&lttarget>が50%の確率でロボット化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に失敗したことを知らせる。
- **PresidentialPardonForm**です。必要な成績 : sign 25, exec 5  
Zaphod Beeblebroxによって&lttarget>が恩赦されたことを通知します。

これらはすべて、コンストラクタに1つのパラメータを取るだけです : フォームのターゲットです。例えば、自宅に低木を植えたい場合は "home" とする。

さて、ベースフォームに `execute(Bureaucrat const & executor) const` メンバ関数を追加し、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します。フォームに署名があることと、フォームを実行しようとする官僚の等級が十分に高いことを確認する必要があります。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確認するか、ベースクラスで確認するか（それから別の関数を呼び出してフォームを実行するか）、それはあなた次第です。しかし、どちらかの方法の方がきれいです。

最後に、`executeForm(Form const & form)` メンバ関数をビューロクラットに追加してください。これは、フォームの実行を試みなければなりません。もし成功したら、次のような内容を表示します。


<bureaucrat>が<form>を実行しました。

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。

期待通りに動作することを確認するために、いくつかのテストを実施し、提出する。

## 第六章

### 練習問題03：少なくともコーヒーマーカーよりはましだ

	演習：03
少なくとも、コーヒーマーカーよりはましだ	
ターンインディレクトリ： <i>ex03/</i>	
提出するファイル:過去の演習のファイル + Intern.{h, hpp}, Intern.cpp	
禁止されている機能:なし	

書類の記入だけでも十分迷惑なのに、官僚に一日中これをやらせるのは酷な話です。幸いなことに、インターンは存在します。この演習では、**インターン**クラスを実装する必要があります。インターンには名前も等級も固有特性ありません。官僚が気にするのは、彼らが仕事をするだけです。

しかし、このインターンには、**makeForm()**関数という重要な能力があります。これは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、2番目の文字列はフォームのターゲットです。この関数は**フォームオブジェクト**へのポインタを返し、そのオブジェクトの名前はパラメータとして渡されたものです。

というような印字がされます。

インターンが<form>を作成

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します。

if/elseif/elseの森を使うような、読めない、醜い解答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かどうかテストしなければなりません。

例えば、以下のコードでは、"Ben-der " をターゲットとした **RobotomyRequestForm** を作成しています。

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