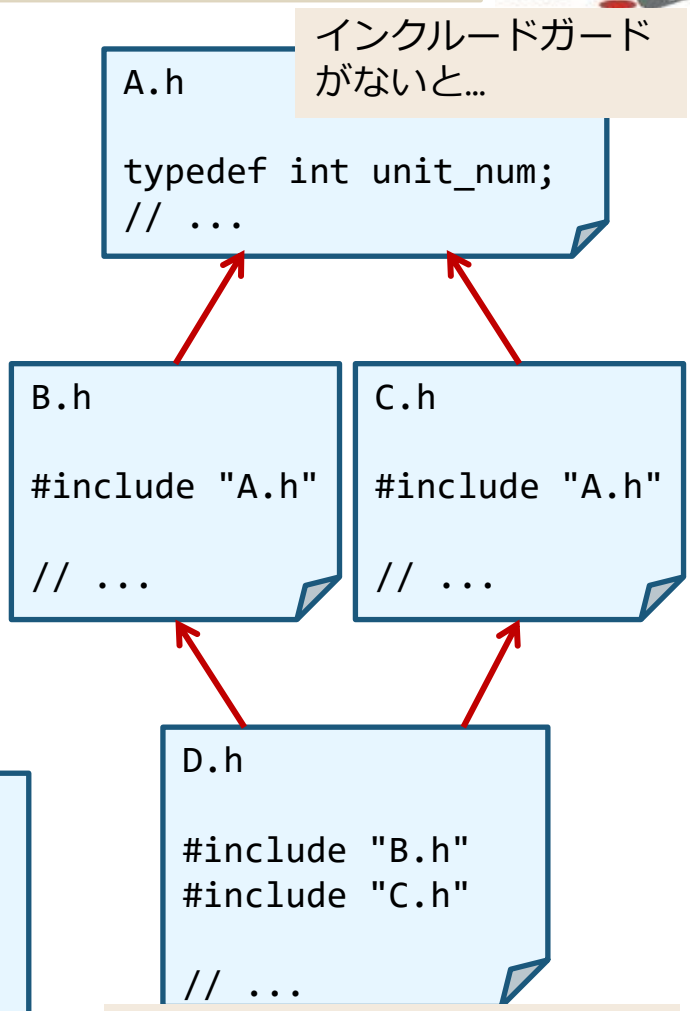
ETロボコン公式トレーニング モデリング入門

補足資料

a-1. インクルードガードについて

- 多重定義を避ける工夫です
  - 同じヘッダファイルを複数回includeすると、何度も同じ定義が見つかったとみなされ、コンパイルエラーになります
  - そこで、プリプロセッサの機能を使って、一度読み込んだヘッダファイルをスキップする仕組みに仕立てたものがインクルードガードです
  - 最近のC/C++コンパイラでは、インクルードガードのしくみを提供する
    `#pragma once`
    というマクロが使えるようになってきています




```
A.h
#ifndef HEADER_H
#define HEADER_H

typedef int unit_num;
// ...

#endif /* HEADER_H */
```

古典的なインクルードガード
（古いコンパイラでも使える）

```
A.h
#pragma once

typedef int unit_num;
// ...
```

モダンなインクルードガード
（新しいコンパイラで使える）

a-2. コーディングルールについて

- コーディングするうえで、規範となるルール
  - 組織やプロジェクトによって規定されます
    - 命名規則、コーディングスタイル、コメントの付け方、などなど
    - 何を採用するか、どこまで決めるかなど、開発規模や考え方によってルールは様々
  - ルール化することでソースコードに統一性が生まれるため、保守性や可読性が上がります

- 公開されているコーディングルールの例
  - 組込みソフトウェア開発向けコーディング作法ガイド［C言語版］、［C++言語版］
  - Google C++コーディングガイド

- 付録a-3（次ページ以降）に、このテキストのサンプルプログラムを作成するために使用したコーディングルールのうち、命名規則について紹介します

a-3. サンプルコード命名規則（１）

1. ファイル名はクラス名と同一の名前を使用する
   - 例）ClassName.cpp/ClassName.h
2. 型名（クラス名、構造体型名、列挙型名など）は大文字から始め、アンダースコアは使用せずに単語の先頭は大文字とする。
   - 例）ClassName
   - 理由）統一した書き方であればよいため、特になし
3. 関数名は小文字から始め、アンダースコアは使用せずに単語の先頭は大文字とする。
   - 例）funcName
   - 理由）統一した書き方であればよいため、特になし
4. 変数名は小文字から始め、アンダースコアは使用せずに単語の先頭は大文字とする。
   - 例）variableName
   - 理由）統一した書き方であればよいため、特になし

a-3. サンプルコード命名規則（2）

5. グローバル変数にはプレフィックスとして「g」をつける
   - 例）gGlobalData
   - 理由）グローバルデータを使用する場合は明確に他と区別されるべきである
6. クラスのメンバ変数にはプレフィックスとして「m」をつける
   - 例）mMemberData
   - 理由）クラス内でアクセスする際に、メンバ変数であることが一目で分かるべきである
7. 定数はすべて大文字とし、単語間はアンダースコアを入れる
   - 例）CONST_DATA
   - 理由）定数であることを明示し、その他の変数との区別を明確にするべきである

a-4. コンストラクタの初期化リストの使用

- インスタンスを生成するとコンストラクタはメンバ変数を初期化します
  - メンバ変数を初期化するには初期化リスト（メンバイニシャライザ）を使います
    - メンバ変数が const 修飾されているときは、初期化リストを使わないと初期化する機会がなくなってしまう
  - コンストラクタのコード中でメンバ変数へ初期値を代入するコードは、関連しているクラスや属性の初期化で不都合が生じることが多いので、初期化リストを使いましょう
    - コンパイラに警告されます
- 初期化リストを使ったコンストラクタの書式

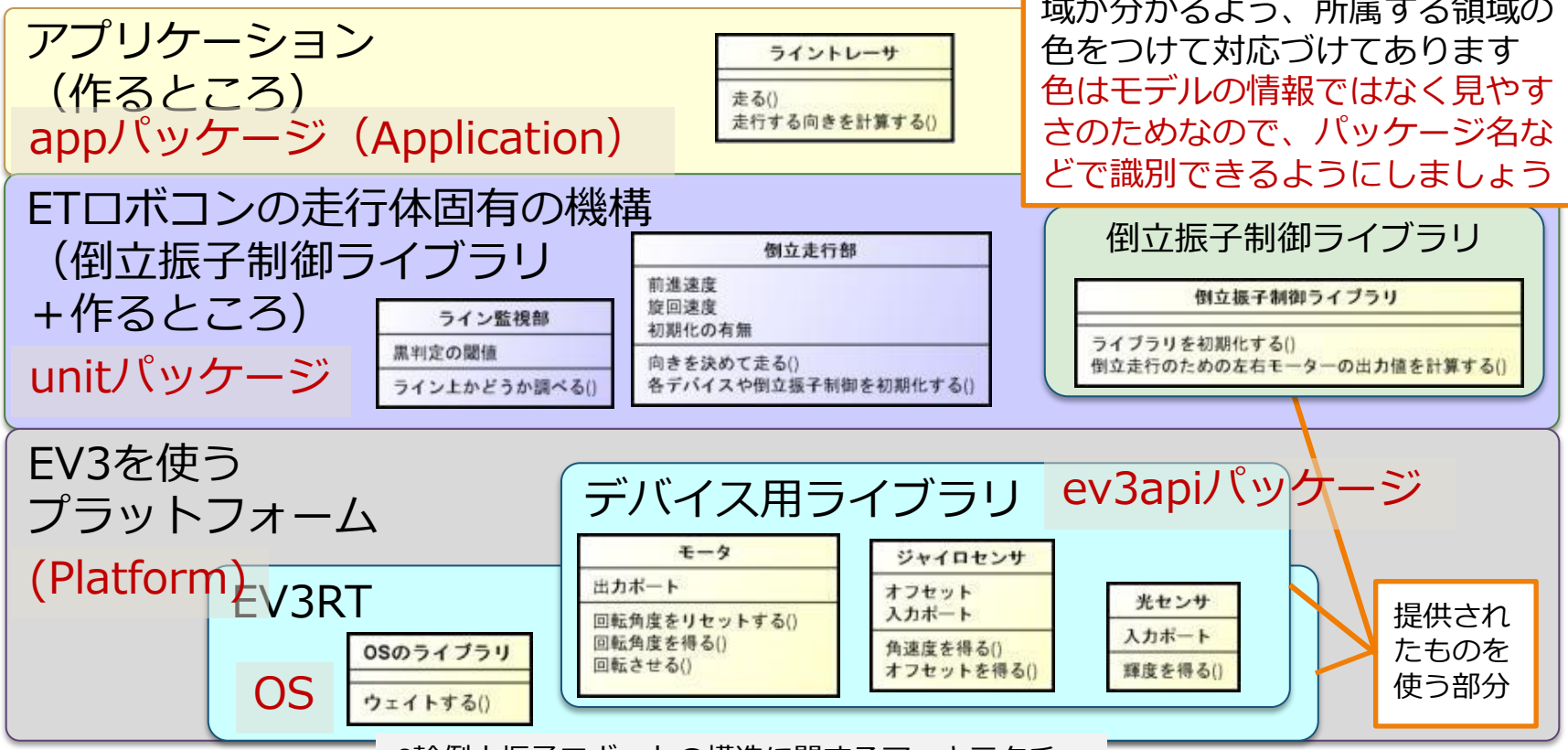
```
コンストラクタ(引数) : メンバ名(初期化値),メンバ名(初期化値), ... {
    コンストラクタ本体の処理
}
```

- 初期化子リストには記述順序があります
  - クラス定義の中でメンバ変数を書いた順番通りに書きます
    - コンパイラに警告されます（警告しないコンパイラもあるそうです）
- 参考文献：Effective C++改訂第2版
  - p72-76、12項「コンストラクタでは、代入より初期化リストを使おう」
  - p77-78、13項「初期化リストのメンバは、宣言した順に並べよう」

a-5.　演習のモデルの構造のアーキテクチャ

- 4-3「実装を作成する」の背景としている構造のアーキテクチャです
  - センサやモータは、開発環境が提供するデバイスドライバを使います
  - 倒立振子制御ライブラリは、提供されたものを使います
  - 残りは、ロボットの機構と機能を実現するアプリケーションに分けます

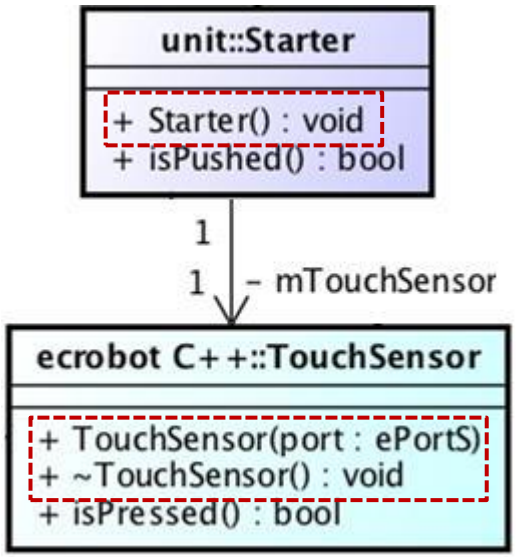
4-3.でこの構造のアーキテクチャを見出す以前の色分けは主観的ですが、この構造のアーキテクチャを見出して以降に作成したクラスやオブジェクトには、所属する領域が分かるよう、所属する領域の色をつけて対応づけてあります
色はモデルの情報ではなく見やすさのためなので、パッケージ名などで識別できるようにしましょう



2輪倒立振子ロボットの構造に関するアーキテクチャ

a-6. デストラクタに関する補足

- サンプル中のデストラクタを作成していないクラスについて
  - コンパイラによって用意される暗黙のデストラクタを使っています
    - C++の新しい仕様（C++11）では、「virtual ~Starter() = default;」のように、暗黙のデストラクタに任せることを明示できます
    - 参考：https://dev.activebasic.com/egtra/2011/12/02/451/
  - 明示的なデストラクタの有無について
    - オブジェクトが開放されるときに一緒に破棄したい他のオブジェクトがある時は、明示的にデストラクタを書いて破棄の手順を書きます
    - 参照しているオブジェクトなどを破棄しないなら、デストラクタを書かないで暗黙のものに任せられます
- コンストラクタ、デストラクタ使い方
  - コンストラクタ、デストラクタの使い方の詳細は、参考文献にあるC++の解説書を参考にしましょう



明示的なデストラクタがないクラス（Starter）あるクラス（TouchSensor）の例