

オブジェクト指向プログラミング

〇〇編 第1回

クラス/インスタンスと データ抽象の概念

- 目的
 - クラス, インスタンスの仕組みを理解する
 - 描画ライブラリを用いたGUIの仕組みを理解する
- キーワード
 - クラス, インスタンス, オブジェクト, インスタンス化, コンストラクタ, カプセル化

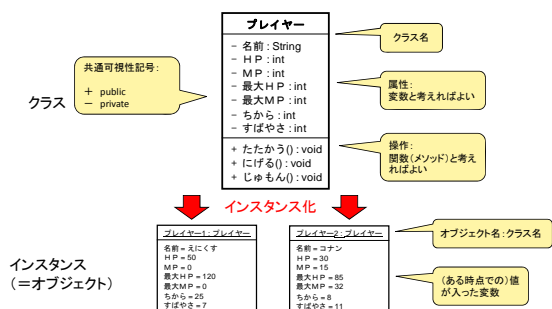
オブジェクト指向とは

- (現実)世界をオブジェクトとその関連するシステムとしてモデル化しようとする考え方
- ソフトウェアが現実世界の写像だとすれば, 現実世界のモデル化によりソフトウェアが作成できる
- オブジェクト指向モデルの採用によって, 現実とソフトウェアモデルをうまく融合できる可能性がある
 - 我々はよく, 現実世界をオブジェクト指向的な見方で捉えている
 - ソフトウェア工学の研究ではオブジェクト指向(データ抽象)がモジュール性と再利用性を高めるために有効

クラスモデル

- オブジェクト指向の概念に基づく世界のモデル
 - クラス
 - 概念的には: "もの or こと" の分類をする
 - ソフトウェア的には: 抽象的なデータを表現する
 - インスタンス
 - クラスから出来たオブジェクトのことを言う

クラス/インスタンス(UML表記)



オブジェクト指向の利点(カプセル化)

- クラスのないプログラム
 - データ構造が変わると, 利用するアルゴリズム全てに変更必要
- 例: 時刻のデータ

```
struct TimeA{
    int minute;
    int second;
}

int getMinute(TimeA* time){
    return time->minute;
}

int getSecond(TimeA* time){
    return time->second;
}
```

```
struct TimeB{
    int second;
}

int getMinute(TimeB* time){
    return time->second/60;
}

int getSecond(TimeB* time){
    return time->second%60;
}
```