オブジェクト指向プログラミング

OO編　第3回

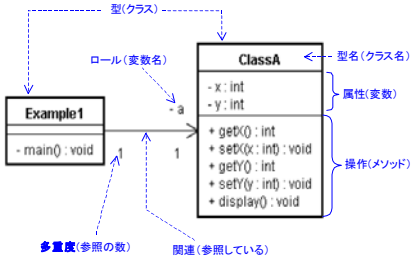
継承を利用した設計

- 目的
  - クラス図・オブジェクト図を書けるようになる
  - 継承の仕組みと利点欠点を理解する
- キーワード
  - クラス図・オブジェクト図，継承

クラス図・オブジェクト図

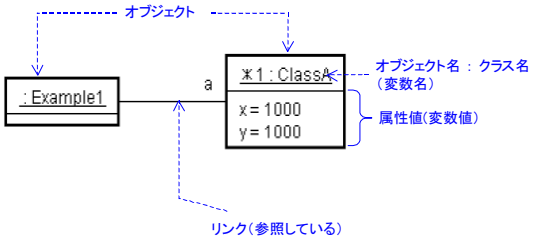
- UML(Unified Modeling Language)で定義されている記法
  - クラス図
    - クラス（型）の関連構造を表現する為の記法
  - オブジェクト図
    - クラスをインスタンス化したものの関連構造を表現する為の記法

クラス図



多重度　：最少数..最大数, “*”は無限, 省略: “*” = “0..*”, “1” = “1..1”)

オブジェクト図

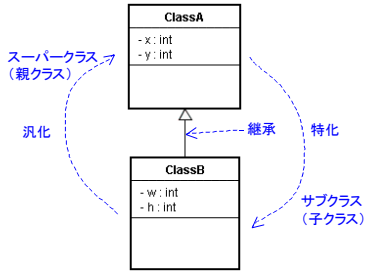


- クラス図が書ければオブジェクト図が書ける
- ※オブジェクト図に多重度はない

3. 継承

- クラスが増えると困る
  - プログラムが大きくなってくると似たようなクラスがでてくる
  - 同じコードを何度も書くなんてイヤ
  - 複雑で分かりにくい
- クラスのふるまい（変数やメソッド）を受け継いだクラスを作るのが継承
  - 同じコードを1つにまとめることが出来る
  - 全体のクラスの構造が分かりやすくなる

継承 – クラス図



- ※関連は矢印。継承は白抜き矢印

継承 – 実装方法

```
public class サブクラス名 extends スーパークラス名 {
 …
}
```

継承 – 仕組み

- 継承される
  - 変数・メソッド
  - コンストラクタ
- オーバーライド
  - サブクラスで、スーパークラスと同じシンボルの・メソッドを定義すると、それを上書きする。
  - 変数・コンストラクタはオーバーライドできない

継承 – 概念

- 継承はソースコードを再利用するためだけのものではなく、
  クラスの抽象部分を抽出するためのもの。
  汎化、特化が継承の本質。
- 継承はポリモーフィズム（多態性）を実現し、
  これを使うと、抽象化した設計が可能。
  （次回授業で）