モジュールへの分解

大規模システムの設計には，モジュール分解が不可欠

ではどうやって分解する？

隠蔽性
独立性
} 高める

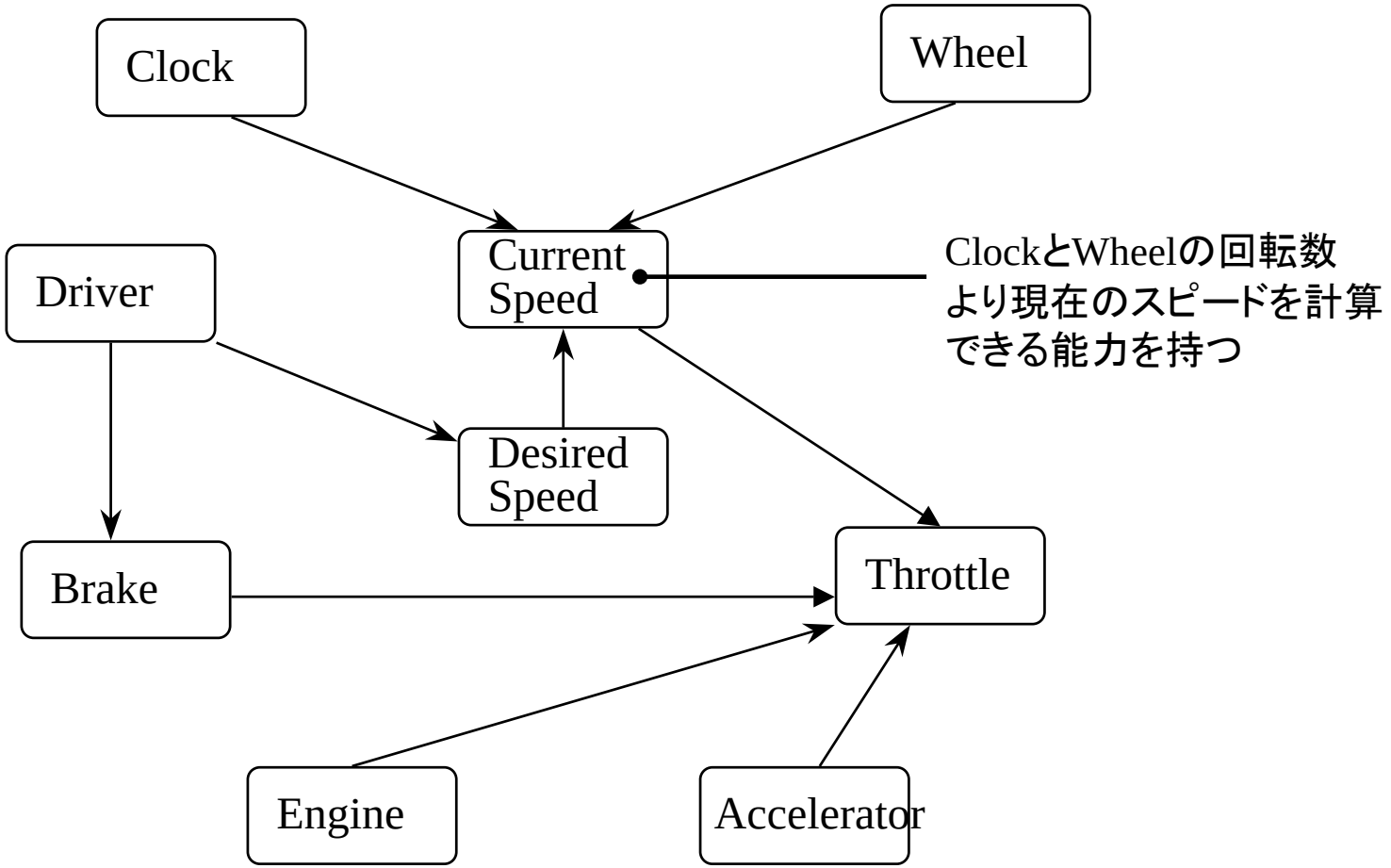
- オブジェクト指向（もの指向）
  - 実世界に存在するオブジェクトごとに分解
- 機能分解：従来のやり方

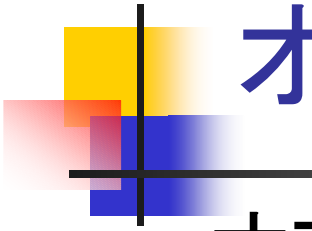
機能分解の例

設定したスピードに保つようにスロットルを制御する

オブジェクト指向分解の例

オブジェクト（もの）が独自の計算能力を持っていると考える





オブジェクト（１）

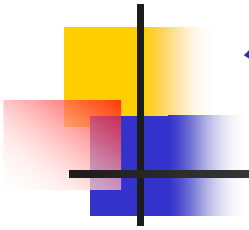
オブジェクトとは？

- 固有の属性（内部状態）を持つ
- 他のオブジェクトと関係を持つ
- uniqueな名前を持つ
- ある共通の性質を持つものをまとめて1つのグループとすることができる：クラス(Class)

オブジェクト（２）

オブジェクトとは？

- 固有の操作を持ち，この操作のみが許される：カプセル化，情報隠蔽
  - 属性値の参照
  - 属性値の変更
  - オブジェクトの生成・消滅
- 操作名が同じでも異なるクラスのオブジェクトに適用されれば，効果は異なる(Polymorphism)
- メッセージ通信による計算(Message Passing)

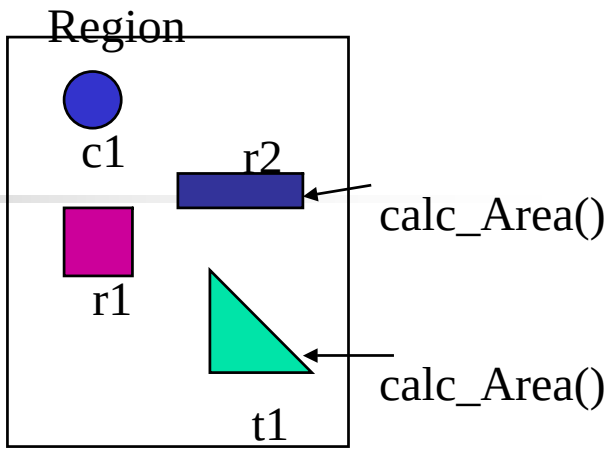


クラス階層

- スーパークラス，サブクラス
  - サブクラスの元はスーパークラスの元
  - スーパークラスが持っている性質はサブクラスに引き継がれる(Inheritance：インヘリタンス)
    - 属性や操作を共通に持つことができる
  - 差分（異なる部分）を記述すればよい

例題

- Region内に含まれる図形の面積の総和




```
public class Figure {
…
}

public class Rectangle
    extends Figure {
   …
}
….
```

```
public class Region {
    Figure[]  f = new Figure[50];

  public double sum_of_Area() {
          /* 計算 */
     return sum ;
    }
 }
```

図で書こう

```
public class Figure {
   double x, y ;
     ...
  }
```

```
public class Region {
   Figure[]  f = new Figure[50];
   ...
}
```

```
public class Rectangle extends Figure {
     double width, height ;
     public double calc_Area( ) {
       ...
     }
     ...
}
```

設計図 → ソースコード

クラス図，オブジェクト図（1）

クラス

人間

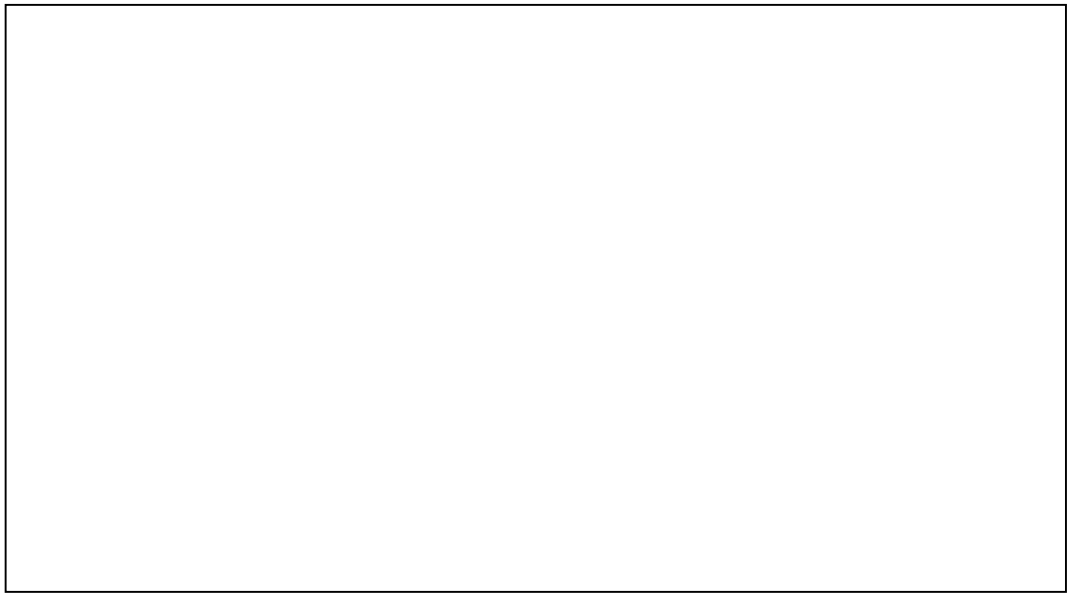
属性

人間
氏名:String 年齢:Integer

操作

人間
氏名:String 年齢:Integer
転職する 転居する

Class name

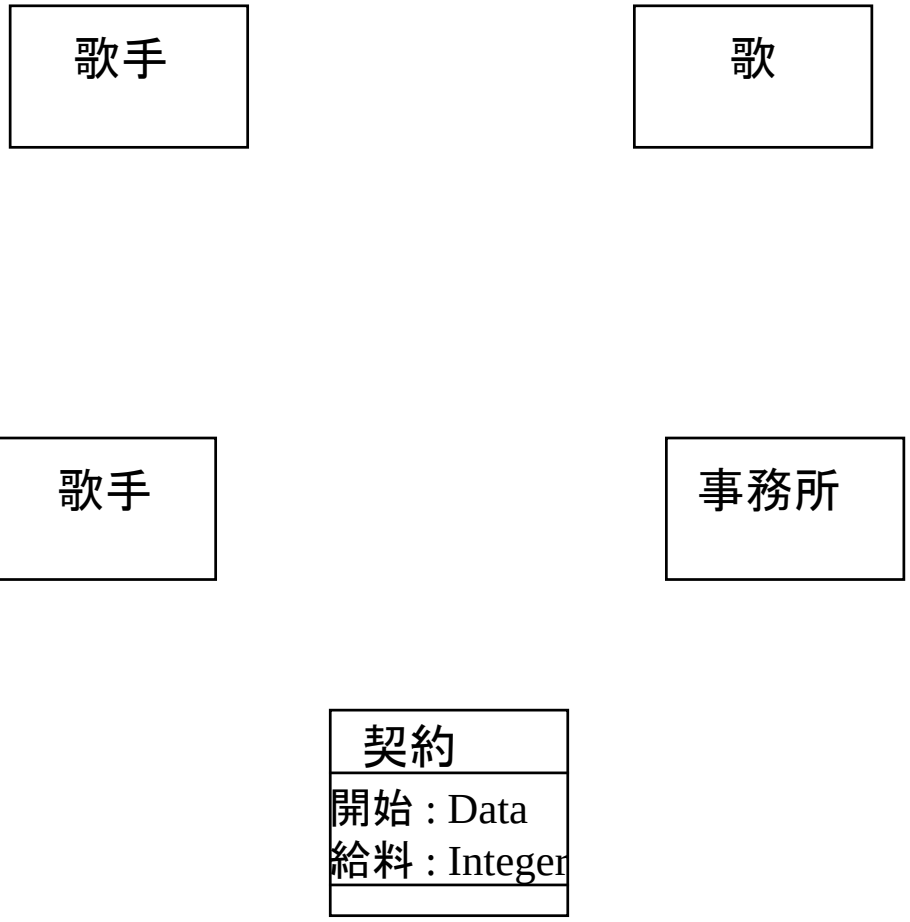


オブジェクト間関係

- 関連とリンク：基本は2項関係
  多重度(Multiplicity, Cardinality)
  ：何対何対応か
  - 関連（Association
  - リンク：関係のインスタンス (Link)
- 集約(Aggregation)：Has a 関係
  関連の一つと見ることができる

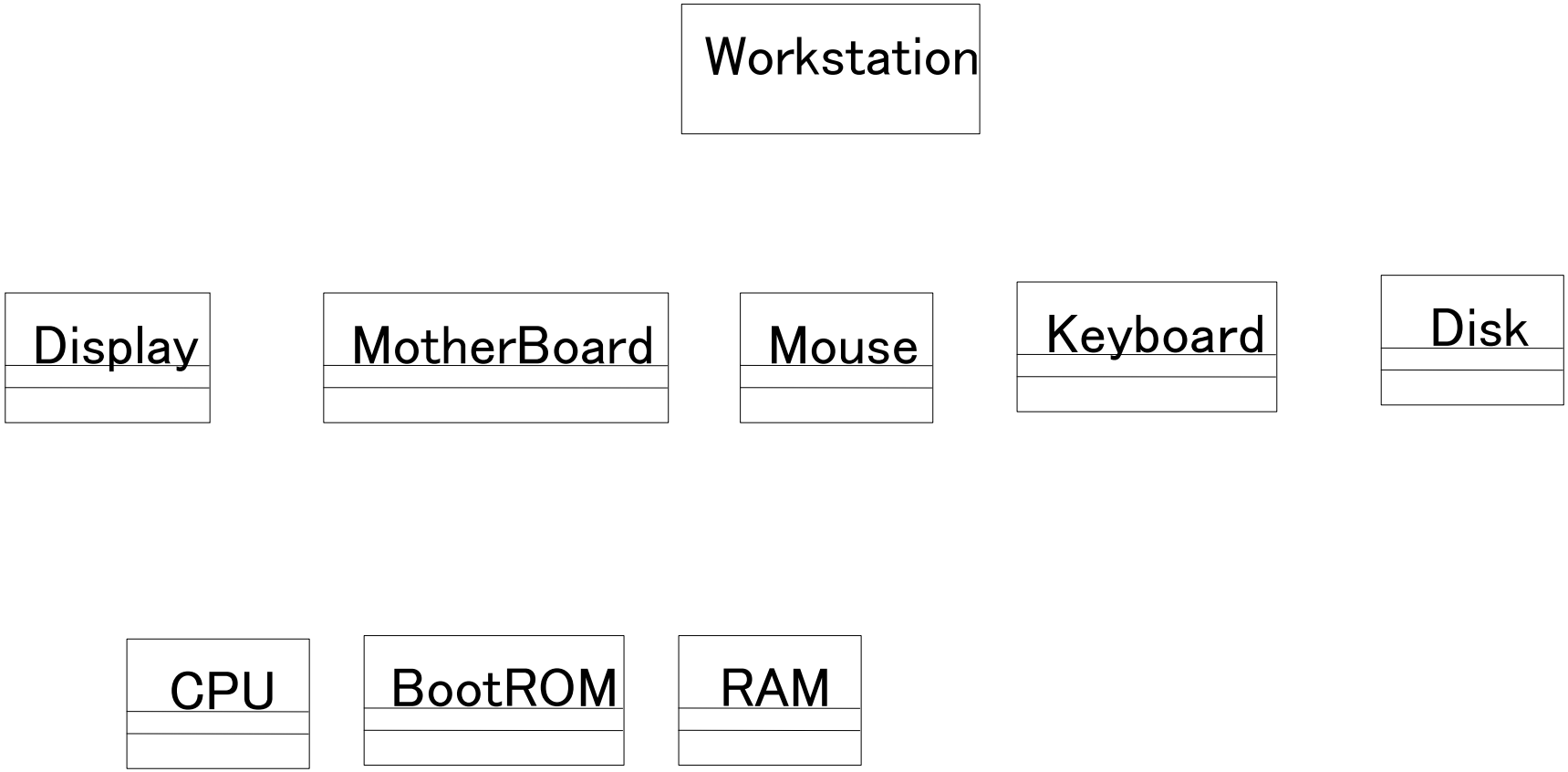
クラス図，オブジェクト図（2）

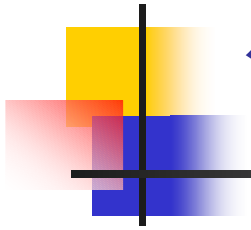
クラス図，オブジェクト図（３）



クラス図，オブジェクト図（４）

集約(Aggregation)：Has a 関係





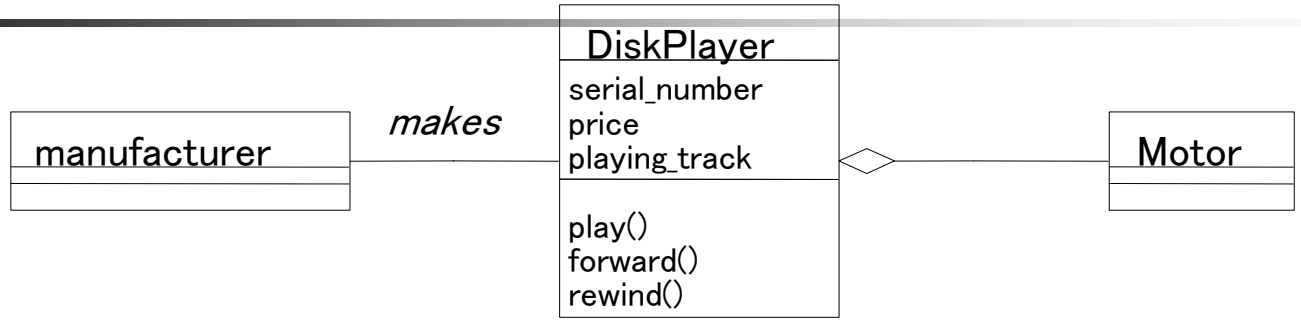
クラス間関係

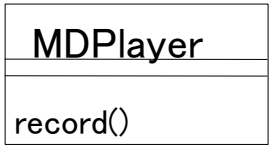
- 汎化（一般化：Generalization）：Is a 関係
  - スーパークラス，サブクラスの関係
- 継承 (Inheritance)
  - スーパークラスの属性，操作，関連を引き継ぐ
  - オーバーライド (Override)

    ：同じ定義があったとき，サブクラスの定義を優先

クラス図・オブジェクト図（５）

汎化(Generalization)，継承





多重継承
(Multiple Inheritance

クラス図(6)

ロール（Role:役割），限定付き関連（キー付き）