



Interfaceの利用

オブジェクト指向プログラミング特論

2018年度

只木進一：工学系研究科

クラスと階層構造

■ Class

- 対象の類型を表す

■ 日常における類型化

- 階層構造

- 上位：一般化

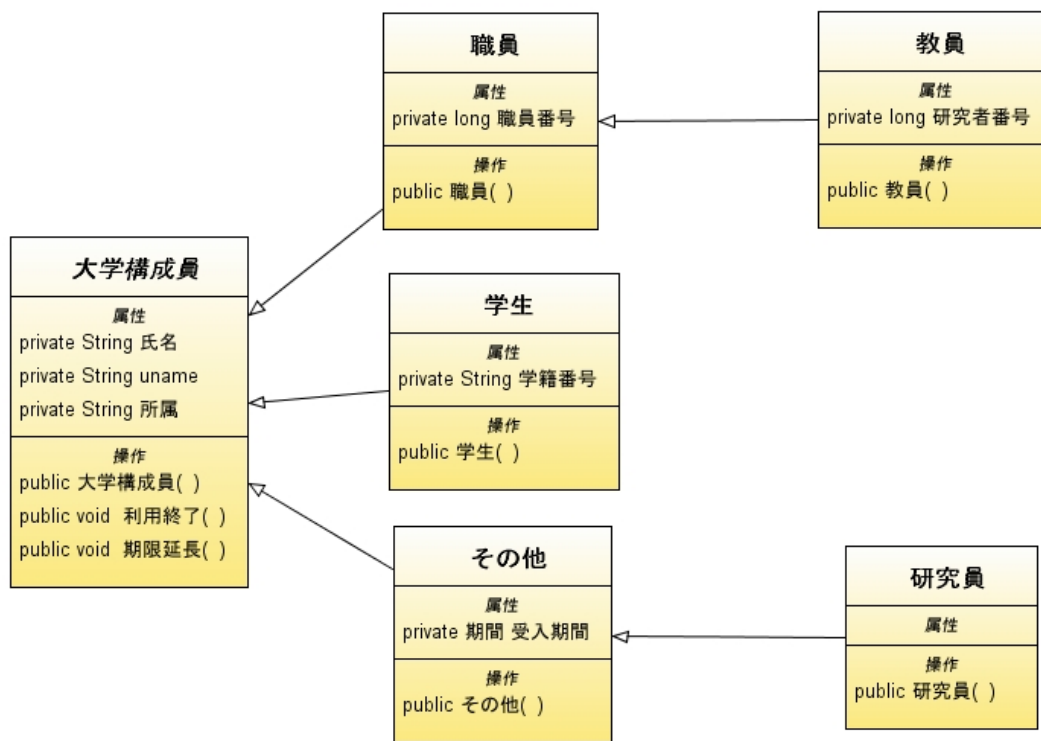
- 下位：具体化

■ 適切な階層化が概念整理に有効

Class, Super Class, Subclass

- クラスには階層構造がある
 - 例：生き物の階層構造
 - 例：組織の階層構造
- 上位のクラス：super class
 - 抽象化、汎化 (Generalization)
- 下位のクラス：subclass
 - 具体化 (Specialization)

クラス階層の例



継承 (Inheritance)

- Subclassの定義
- 全てのフィールドとメソッドが継承される
- 具体化 (Specialization)
 - フィールド・メソッドの追加
 - 実装の追加・変更

汎化 (Generalization)

- 共通なものとしての取扱い
- 共通的フィールド・メソッドの抽出
 - 実施なしの場合も
 - 名前だけの定義

Method Overload

- メソッドの二つの要素
 - **contact** または **signature**
 - メソッド名
 - メソッドの引数並び
- 実装
 - メソッドを多重に定義できる

Polymorphism

- 継承したクラスでメソッドを上書き
- **super**を使って、親のクラスを示す
- クラスを利用する側から見ると、親クラスのインスタンスとしても利用できる。

Javaでの継承の制約

- 多重継承の困難さ
 - 複数の親クラスのどの性質を引き継ぐか
- Javaでは
 - 一つのクラスしか**extends**で継承できない
 - 特殊なクラス**interface**
 - 複数の**interface**を継承できる
- **Abstract class**と**Interface**の使い分け

interface

- 特殊なクラス
 - フィールドの制限
 - `static final`のみ持つことができる
 - これらキーワードは省略化
- メソッドの制限
 - `abstract`メソッド：実装なし
 - `default`メソッド：実装あり

- 継承クラスで、**interface**を**implements**する
 - メソッドを必ず実装する
 - **default**メソッドをそのまま使う場合には**super**で明示する。
- 操作する側の視点での抽象化
 - 特定のメソッドを有しているため、同じ方法で操作できる

Comparable インターフェース

- API ドキュメントを見よう

- <https://docs.oracle.com/javase/jp/8/docs/api/>

- `java.lang.Comparable`

- 定義されているメソッド
 - 既知の実装クラス

例：StudentクラスにComparable インターフェイスを追加

■ クラス定義

```
public class Student implements  
Comparable<Student>
```

■ メソッド実装

```
public int compareTo(Student o)
```

MergeSortをComparableであるクラスに対応させる

- クラステンプレートの利用
 - 特定のクラスを指定しない
 - どのクラスの拡張であるかを指定し、使えるメソッドを特定

```
public class MergeSort<T extends Comparable>
```

MergeSortクラスをクラステンプレートに対応

- 要素の大小関係を調べている部分の特定
 - `compareTo()`メソッドに置き換え
- 正しく動作することを確認

宿題

- Comparableの派生クラスを対象としたBubbleソートを作成
- MergeSortクラスと共通の部分
 - コンストラクタでデータを与える
 - sort()メソッドで実行
 - less()メソッドは共通
- swap()メソッドを作る