

Java入門：クラス の利用

モデリングとシミュレーション

1

2017年度

目的

- 基本的クラスの利用
 - 講義で使用するライブラリ
- 抽象クラスの利用
 - インターフェースの利用

Javaの豊富なライブラリ

- ▶ プログラミングで共通的に必要なライブラリが、言語とともに配布されている。
- ▶ 良く使うライブラリを紹介し、利用方法を学ぶ

原始型と対応するクラス

- ▶ クラスでない型
 - ▶ int、double、boolean、char、など
- ▶ 対応するクラス
 - ▶ Integer、Double、Boolean、Character、など
- ▶ 文字列クラスString
 - ▶ 比較、部分文字列などのメソッド

抽象クラス

- ▶ Abstract Class
 - ▶ メソッドの一部が実装されていない
 - ▶ データの保有や処理の具体を記述しない
- ▶ Interface
 - ▶ 少数のメソッドのみを持つ
 - ▶ 他のクラスから呼ばれる方法を定義

抽象クラス リストを例に

▶ java.util.List

- ▶ リストを操作するメソッドが定義された
インターフェース
- ▶ リストを操作する方法のみを定義

▶ java.util.AbstractList

- ▶ ランダムアクセス的リストの抽象クラス
- ▶ 一部のメソッドが実装されていない

リストの実装クラス

- ▶ `java.util.ArrayList`
 - ▶ 検索が高速
 - ▶ `AbstractList`の拡張クラス
- ▶ `java.util.LinkedList`
 - ▶ データの挿入・追加が高速
 - ▶ `AbstractSequentialList`の拡張クラス

リストの利用

■ 生成

■ `List<T> list =
Collections.synchronizedList(new
ArrayList<>());`

■ スレッド間での同期を指定

■ 要素追加 : `list.add(T t);`

■ 要素取り出し : `T t = list.get(int j);`

■ 要素再設定 : `list.set(int j, T t);`

型パラメタ

- クラスが扱い対象を特定せずにパラメタ化できる

```
public class Store<T> {  
    List<T> list;  
  
    public Store(List<T> list){this.list=list;}  
  
    public boolean addData(T t){return list.add(t);}  
}
```

型パラメタ

- ▶ List<T>のTは、型パラメタ
 - ▶ そのリストに格納されるクラスを指定
 - ▶ 左辺で型を指定した場合は、右辺では<>として省略できる。

```
//クラスインスタンスのリスト
List<Student> students=
    Collections.synchronizedList(new ArrayList<>());
//配列に登録、及び成績登録
for (int i = 0; i < names.length; i++) {
    Student s =new Student(names[i], i);
    s.setRecord(records[i]);
    students.add(s);
}
```

インターフェースの例

Comparable

- クラスインスタンスが比較できることを定義
- 必ず `compareTo` メソッドを有する
 - 自分が対象よりも大きい：正
 - 自分が対象よりも小さい：負
 - 自分が対象と同じ大きさ：ゼロ
 - 何を比較するかを記述する。

Comparableインターフェース
は比較対象を指定する

```
//クラスDataは相互に比較できる
public class Data implements Comparable<Data>{
    private int x;

    public Data(int x){this.x = x;}

    //比較はx の値の大小で行う
    public int compareTo(Data o){
        return this.x - o.x;
    }
}
```

型パラメータを持つメソッド generic method

- 汎用的なメソッド（サブルーチン）
 - 対象となる具体的な型を特定せず、その特性（インターフェース）が分かれば処理できる
 - 例えば、整列は比較できるモノであれば整列できるはず

- 戻り値の前で型パラメタを定義
- ここでは、TはTと比較できるクラスであること
 - TのインスタンスはcompareToを持っていて→これを使って比較できる

```
public static <T extends Comparable<T>> void sort(List<T> list) {  
    //ソートの実装  
}
```

NetBeans 利用tips

- ソースファイルの自動整形
 - ソースファイル編集画面でマウス右ボタン→「フォーマット」
- 変数名等の変更
 - ソースファイル編集画面でマウス右ボタン→「リファクタリング」→「名前の変更」
 - プロジェクト全体に変更が行われる

- ▶ コードの挿入
 - ▶ ソースファイル編集画面でマウス右ボタン→「コードの挿入」
 - ▶ コンストラクタ、設定メソッド、取得メソッドなど
- ▶ メソッドや変数名の自動補完
 - ▶ インスタンスの後にピリオド
- ▶ 行番号に出るバルーン
 - ▶ マウスを合わせてメッセージ表示
 - ▶ マウス左ボタンで、対応策提案