




# 基本的なクラス

オブジェクト指向プログラミング特論

2016年度

只木進一：工学系研究科



# 目的

- Javaが標準で持つクラスで、良く用いるものを知る
- 抽象クラス、インターフェース、型パラメタなどを、例を通じて理解する。
- Lambda式の使用例を理解する。


# 原始型と対応するクラス

- クラスでない型(原始型)
  - int, short, long, float, double, char, byte, boolean
- 対応するクラスがある
  - Integer, Short, Long, Float, Double, Character, Boolean
- 代入は、自動的に型変換される



# String

- Stringは文字列を保持するクラス
- 保持している文字列を変更できない
- 比較できる(Comparableを実装)



# Stringの主要メソッド

- `String valueOf()` : 引数に対応する文字列を返す
- `int length()` : 長さを返す
- `boolean equals(Object)` : 比較
- `String[] split(String reg)` : 正規表現 `reg` で、文字列を分割



# StringBuilder

- 文字列を作るクラス
- 主要メソッド
  - `append(Object o)`
    - `o` を文字列に変換して追加する
  - `delete(int start,int end)`
    - 一部分の削除
  - `toString()`
    - 文字列へ変換



# リストなど


- リスト、集合、写像など、クラスオブジェクトの集まりを収容するクラス
- 保存できるクラスを指定する
  - コンパイル時にエラー検出可能
  - 内容取り出し時に、自動的に型変換

# リスト

- java.util.Listは、インターフェイスであり、インスタンスを生成できない
- スレッド間での整合に注意

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





# リストの利用

- リストへの追加 : `put(E e)`
- リストからの取り出し :  $i$  番目の要素
  - `E get(int i)`

# Listの操作

## ➡ 「全ての要素に対して」の基本形

```
StringBuilder sb = new StringBuilder();  
for(T p:list){  
    sb.append(p.toString()).append(nl);  
}
```

## ➡ 総和を計算する

```
int a = 0;  
for(Entry entry:list){  
    a += entry.getScore();  
}
```

# Lambda式を使ったListの操作

## ■ 「全ての要素に対して」の基本形

```
StringBuilder sb = new StringBuilder();  
list.stream().forEachOrdered(  
    p -> {  
        sb.append(p.toString()).append(nl);  
    }  
);
```

## ■ 総和を計算する

```
int a = list.stream().map(  
    (entry) -> entry.getScore()).reduce(0,  
    (accumulator, v) -> accumulator + v);
```

# 写像

## ■ キーとバリューの組

```
Map<K,V> map =  
    Collections.synchronizedMap(new HashMap<>());
```

■ 一つのキーに一つのバリュー

■ put(K,V)で追加

## ■ キーの集合の取り出し

■ Map.keySet()



# 写像の操作：Lambda式

```
int a = map.keySet().stream().map(  
    (name) -> map.get(name)  
).reduce(  
    0, (accumulator, v) -> accumulator + v);
```