

# Javaプログラマ育成コース

## 第六回 Java新機能と逆コンパイル

---

ソフトシンク株式会社

代表取締役

周 順彩

zhousc@soft-think.com

# 目次


---

- 総称型
- アノテーション
- 内部クラス、ローカルクラス
- ラムダ式、メソッド参照
- 逆コンパイル
- 練習課題
- 次回の予習タスク

# 総称型とは？

---

## ◆ 総称型とは

総称型とは「」記号で囲まれたデータ型名をクラスやメソッドにつけることで、Integer型やString型などの様々な型に対応する汎用的なクラスやメソッドを作る機能のことです。

## ◆ メリット

様々なデータ型でも同じように処理することができる。  
実行時のエラーが発生する危険もなくなる。

## 分類

---

- ◆ クラス

クラス名に続けて「<T>」を記述しています。

- ◆ メソッド

メソッドの戻り値の前に「<T>」を記述しています。

- ◆ インタフェース

インタフェースに続けて「<T>」を記述しています。

# 総称型のワイルドカード

---

ワイルドカードは、データ型を宣言せずに一時的に変数(「?」)を保持する場合に使用します。

- ◆ 非境界ワイルドカード

  - 例: `List<T>`

- ◆ 境界ワイルドカード

  - 上限境界ワイルドカード

    - 例: `List<T extends Number>`

  - 下限境界ワイルドカード

    - 例: `List<T super Number>`

# アノテーション

---

## ◆ アノテーションとは？

Javaのコード上では”@(アットマーク)”からはじまり、コードでは表現しきれない情報を、補足としてつけ加えられます。ソースと同時に管理する例：

@Override

@Deprecated

@SuppressWarnings

## ◆ できること

- ① Javadocを生成する。
- ② コンパイルの警告メッセージをなくす。
- ③ プログラムを実行する環境に合わせて、動作を変える

# 使い方

---

- ◆ 記述場所
  - ① クラス
  - ② インタフェース
  - ③ メソッド
  - ④ メンバ変数
  - ⑤ メソッド変数

# 作り方

---

## ◆ メタアノテーション

@Target

@Retention

@Documented

@Inherited



# 作り方

---

## ◆ メタアノテーション

@Target: 独自に定義したアノテーションが、何を対象に利用できるものを宣言するためのアノテーション。

指定できる値は下記の通りです。

ElementType.*TYPE*

ElementType.*FIELD*

ElementType.METHOD

ElementType.PARAMETER

ElementType.CONSTRUCTOR

ElementType.LOCAL\_VARIABLE

....

# 作り方

---

## ◆ メタアノテーション

@ Retention : 独自に定義したアノテーションの有効範囲を設定するためのアノテーション。

指定できる値は下記の通りです。

SOURCE

CLASS

RUNTIME

# 内部クラス

---

- ◆ 内部クラスとは？  
クラス内部に定義されたクラス。
  
- ◆ 分類
  - ① 非static内部クラス
  - ② static クラス
  - ③ ローカルクラス
  - ④ 匿名クラス

## 内部クラス・非static内部クラス

---

- ◆ インスタンスを生成する方法。
  - 1、外部クラスのインスタンスを生成する。
  - 2、外部クラスのインスタンスからドット「.」を使って内部クラスのオブジェクトを宣言する。

例、OuterClass out = new OuterClass();  
OuterClass.InnerClass inner = out.new InnerClass();

- ◆ 内部クラスから外部クラスにある変数に無制限的なアクセスできる。

## 内部クラス・static内部クラス

---

- ◆ インスタンスを生成する方法。  
外部クラスから直接ドット「.」を使って内部クラスのオブジェクトを宣言する。

例、OuterClass.InnerClass inner = new OuterClass .InnerClass();

- ◆ 内部クラスから外部クラスにあるstaticな変数にアクセスできる。

## 内部クラス・ローカルクラス

---

- ◆ ローカルクラスとは？

ローカルクラスはメソッド内部に生成するクラスです

- ◆ ローカルクラスの使い方

ローカルクラスのオブジェクトの生成はその宣言をしたメソッド内でおこななければならない。

- ◆ ローカルクラスから外部クラスにある変数にアクセスできる。

- ◆ ローカルクラスからメソッド内の変数にはfinal変数しかアクセスできません。

## 内部クラス・匿名クラス

---

### ◆ 無名クラスとは？

javaにおける特殊なクラスの一つです。

### ◆ 特徴

1. クラス名無し
2. 「宣言」と「利用」を一度に行う。

### ◆ 使い方

```
new 親クラスまたはインタフェース名() {  
    // クラス内容  
}
```

# ラムダ式

---

## ◆ ラムダ式とは？

C#などのプログラミング言語ではすでに導入されていたラムダ式です。

Java8からjavaにも実装されるようになりました。ラムダ式の特徴は、メソッドを変数のように扱える点にあります。

## ◆ 書式

(メソッドの引数例) -> {処理内容}

## ◆ メリット

- 関数型インタフェースを実装するためのソースを手短に書ける。
- 「Stream API」の引数としての関数型インタフェースを記述するのに適している。



# メソッド参照

---

- ◆ メソッド参照とは？  
メソッドの引数としてメソッドを参照できる仕組みのことです。
  
- ◆ 書式  
クラス名::メソッド名  
オブジェクト::メソッド名  
クラス名::new
  
- ◆ メリット
  - 関数型インタフェースを実装するためのソースを手短に書ける。

# 逆コンパイル

---

- ◆ 逆コンパイルとは？

Javaのclassファイルから元のJavaソースを生成する。

- ◆ javap

# 練習課題

---

## ◆ 四則演算の拡張

- 総称型、アノテーション、内部クラス、ラムダ式、メソッド参照を利用する。

## 次回の予習タスク

---

### ◆ アルゴリズムJava