# 3-2.コンストラクタ

# ■コンストラクタ

## はじめに

この章では、コンストラクタについて学んでいきましょう！
Javaにおいてもっとも基本的な構造になります。
コーディングしていく上で **コンストラクタ** が重要なポイントになるので、しっかり押さえていきましょう。

## Step1 : コンストラクタとは

**コンストラクタ** は、インスタンスを生成するタイミングで呼び出される、 特別なメソッドです。
呼び出されたコンストラクタ内では、さまざまな処理を行うことができます。

また、コンストラクタは、 **引数** を持つことが可能です。
引数 （ ひきすう ）とは、メソッドに渡すデータのことで、通常はメソッド名の隣のカッコ `()` の中に入っています。
この引数を使用して何らかの処理を行うことも可能です。

通常はコンストラクタ内で、
フィールド変数などにデフォルト値でない任意の値を格納する処理（初期化処理）を記述することが多いです。
また、クラスを使用する際に **習慣化したい処理がある場合** などにも利用します。

> **引数なしの場合**

- ConstructorSample.java

```
public class ConstructorSample {
    private String str = "Fusionia Engineer";

    // コンストラクタ
    public ConstructorSample() {
        System.out.println(str);
    }
}
```

- Main.java

```
public class Main {

    public static void main(String[] args) {
        // インスタンスの生成
        ConstructorSample cs = new ConstructorSample();
    }

}
```

**【出力結果】**

```
Fusionia Engineer
```

> **引数ありの場合**

- ConstructorSample.java

?

```
public class ConstructorSample {
    private String str = "Fusionia Engineer";

    // コンストラクタ
    public ConstructorSample(String str) {
        System.out.println(str);
    }
}
```

- Main.java

```
public class Main {

    public static void main(String[] args) {
        // インスタンスの生成
        ConstructorSample cs = new ConstructorSample("Fusionia System Engineer ");
    }

}
```

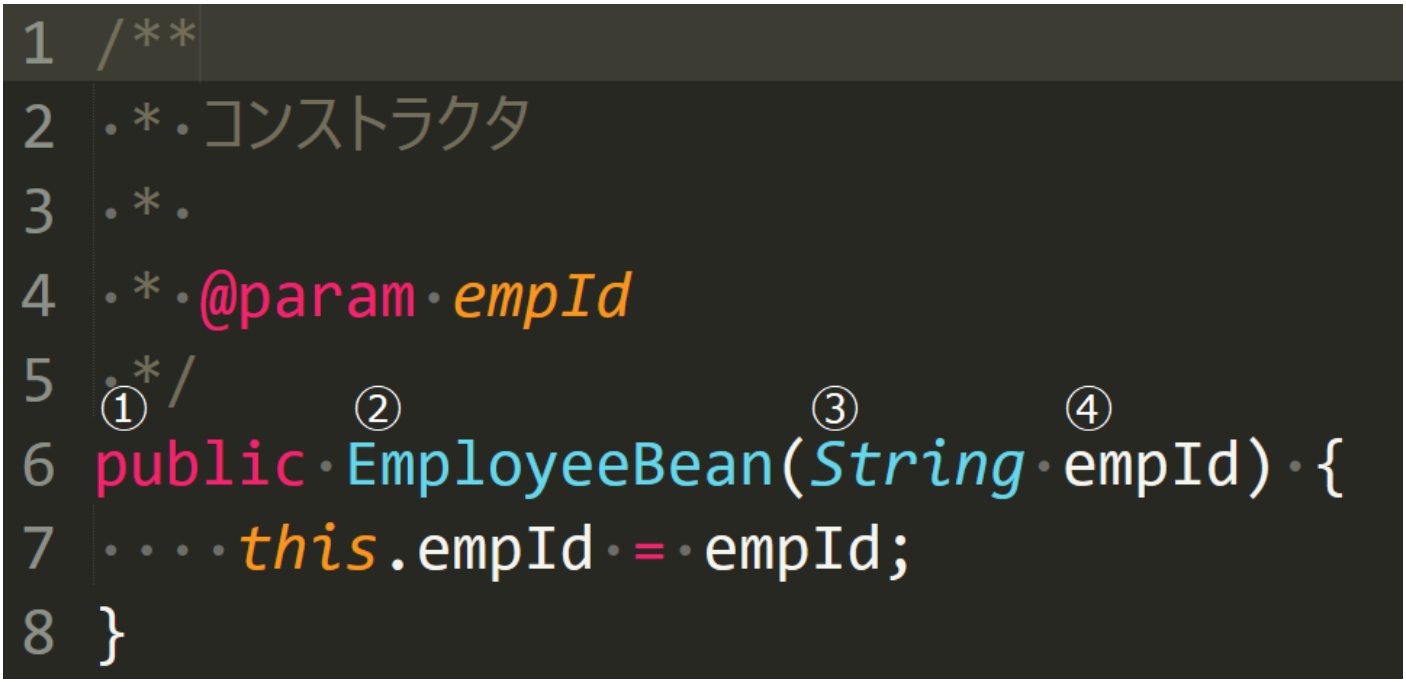
**【出力結果】**

```
Fusionia System Engineer
```

## Step2 : コンストラクタの記述

記述方法を詳細に見ていきます。

以下の画像は **社員情報を扱うようなクラス** のコンストラクタのサンプルになります。
**社員の情報** ってどんな内容が必要かな？ということをイメージして見ていきましょう！




**①アクセス修飾子（省略可）**
**②コンストラクタ名**
**③丸括弧内に引数の型**
**④引数名**

の順で記述し、 **ブロック（ { } ）で囲むようにして** 実際の処理を記述します。
コンストラクタ名については、必ず **クラス名と同名にする** という決まりがあります。

メソッドは値を返却する機能があり、
返却される値のことを戻り値と言います。
**しかしコンストラクタには戻り値がありません。**

クラスに **コンストラクタ** を実装することで、
そのクラス内に存在するフィールド変数への値の設定を一括で行うことができるため、
インスタンス生成後は持つべき情報が詰まったクラスが使用可能となります。

例えば以下のような情報を持つでしょう。

- 社員番号
- 所属部署
- 名前
- 電話番号
- メールアドレス

...etc

仮にこれらの情報が **すべて必須**であるならば、
いちいちインスタンスを生成してからセットするのではなく、
以下のようにして **インスタンス生成時に一括で値をセットする** 方が賢い書き方になります。

> **コンストラクタ: ありの場合の値のセット方法**

- Main.java

```
public class Main{
    public static void main(String[] args) {
        // 呼び出し（new した後で、引数へ与えられた値を元に EmployeeBean が生成されます）
        EmployeeBean employeeBean = new EmployeeBean("00000", "技術部", "コンストラクタ", "XXX-XXXX-XXXX", "java@fusionia");

        // 使用時
        System.out.println(employeeBean.empId);
        System.out.println(employeeBean.department);
        System.out.println(employeeBean.name);
        System.out.println(employeeBean.tel);
        System.out.println(employeeBean.mailAddress);
    }
}
```

- EmployeeBean.java

```
public class EmployeeBean {
    public String empId;
    public String department;
    public String name;
    public String tel;
    public String mailAddress;

    /**
     * コンストラクタ
     */
    EmployeeBean(String empId, String department, String name, String tel, String mailAddress) {
       this.empId = empId;
       this.department = department;
       this.name = name;
       this.tel = tel;
       this.mailAddress = mailAddress;
    }
}
```

> **補足**

`this`を使用することによって、 **「そのクラスのインスタンス自身」** であることを明示的に表しています。

サンプルコード中の`this.empId`を例に挙げるなら、それは「生成したインスタンス自身のフィールド変数`empId`」であるということです。
そのため、どんなインスタンスでも`this.empId`と書くと、そのインスタンスが持つフィールド変数の場所を指すことになります。

`this`の詳細については後ほど説明しますので、ここでは形だけ覚えていただければ大丈夫です。

---

> **コンストラクタ: なしの場合の値のセット方法**

- Main.java

```
public class Main{
    public static void main(String[] args) {
        EmployeeBean employeeBean = new EmployeeBean();

        employeeBean.empId = "0000";
        employeeBean.department = "技術部";
        employeeBean.name = "コンストラクタなし";
        employeeBean.tel = "YYY-YYYY-YYYY";
        employeeBean.mailAddress = "java@fusionia";

        System.out.println(employeeBean.empId);
        System.out.println(employeeBean.department);
        System.out.println(employeeBean.name);
        System.out.println(employeeBean.tel);
        System.out.println(employeeBean.mailAddress);
    }
}
```

- EmployeeBean.java

```
public class EmployeeBean {
    public String empId;
    public String department;
    public String name;
    public String tel;
    public String mailAddress;
}
```

対象となる情報も多くはなく、コンストラクタなしで記述すること自体そこまで手間に感じない方もいるかと思いますが、
使用するシチュエーションによっては、事前にコンストラクタを用意したクラスの方が使い勝手がよいケースもあります。

一概に、コンストラクタを使用することが正解！とは言いませんが、
書き方と使い方はしっかり押さえておきましょう！

## 課題

提出課題はありませんので、一通り学習が終わったら次の章へ進んで下さい。

最終更新日時: 2022年 08月 21日(日曜日) 15:02