| | |
|---|---|
| 目的・範囲:<br>*Objective & Scope* | Design Pattern(builderパターン) |
| 分類名:<br>*Classification Name* | |
| 著作者:<br>*Author* | 富田裕作 |
| 実施日:<br>*Enforcement day* | 2014年6月7日 |
| バージョン:<br>*Version* | ver1 |
| 初版発行日:<br>*Original Release* | 2014年6月7日 |
| 現行版発行日:<br>*Current Release* | 2014年6月7日 |
| キーワード:<br>*Key Words* | デザインパターン,builderパターン |
| 背景情報:<br>*Background* | |

基本用語

| 用語 | 説明 |
| --- | --- |
| builder | 「表現形式」を決定する |
| Director | 「作成過程」を決定する |
| | |
| | |
| | |

※引用文献e-words、WikiPedia

◇目的

# Builder パターンとは、同じ作成過程で異なる表現形式の結果を得るためのパターンです。

◇効果

## 同じような作成過程を省略することができます。

◇背景

**「砂糖水」と「食塩水」をこれから作るとします。**

◇Builderパターンの実際のコードと考え方

```
# SugarWater: 砂糖水クラス (ConcreteBuilder:ビルダーの実装部分)

# SugarWater: 砂糖水クラス (ConcreteBuilder:ビルダーの実
装部分)
class SugarWater
  attr_accessor :water, :sugar
  def initialize(water, sugar)
    @water = water
    @sugar = sugar
  end

  # 素材(砂糖)を加える
  def add_material(sugar_amount)
    @sugar += sugar_amount
  end
end

# SaltWater 塩水クラス (ConcreteBuilder:ビルダーの実装部
分)
class SaltWater
  attr_accessor :water, :salt
  def initialize(water, salt)
    @water = water
    @salt = salt
  end

  # 素材(塩)を加える
  def add_material(salt_amount)
    @salt += salt_amount
  end
end

# SugarWaterBuilder: 加工した水を生成するためのインターフェイス(Builder)
class WaterWithMaterialBuilder
  def initialize(class_name)
    @water_with_material = class_name.new(0,0)
  end

  # 素材を入れる
  def add_material(material_amount)
    @water_with_material.add_material(material_amount)
  end

  # 水を加える
```

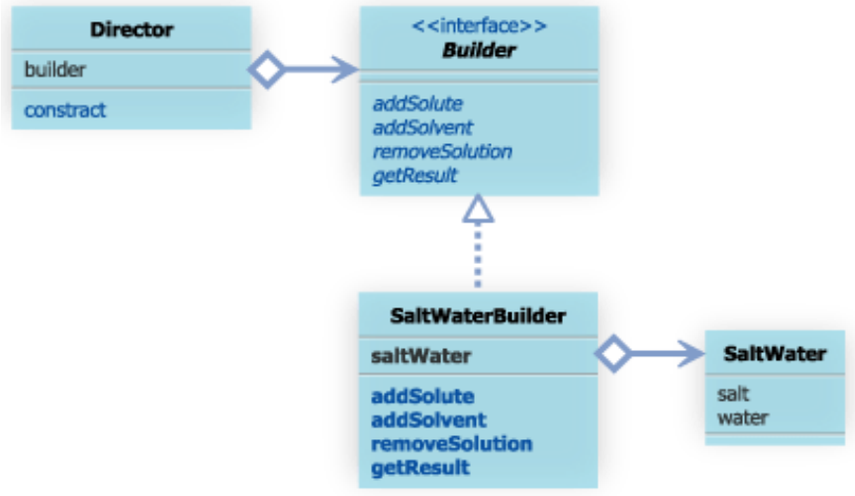
```
  # 水を加える
  def add_water(water_amount)
    @water_with_material.water += water_amount
  end

  # 加工水の状態を返す
  def result
    @water_with_material
  end
end

# Director: 加工水の作成過程を取り決める
class Director
  def initialize(builder)
    @builder = builder
  end
  def cook
    @builder.add_water(150)
    @builder.add_material(90)
    @builder.add_water(300)
    @builder.add_material(35)
  end
end

builder = WaterWithMaterialBuilder.new(SugarWater)
director = Director.new(builder)
director.cook
p builder.result

builder = WaterWithMaterialBuilder.new(SaltWater)
director = Director.new(builder)
director.cook
p builder.result
```




◇注意

## 誰がインスタンスの生成手順を知っているのか

**Directorを使うユーザは、Builderで作られたインスタンスが何かということを知っていなければならない。**

## Builderが提供する生成手順は、増やしたり減らしたりするのが困難です。

◇総括

# 同じような物を作るときは、builderパターンは便利だと思います。