Builder Pattern

복잡한 인스턴스를 단계적으로 조립하기

- 복잡한 구조를 한 번에 완성시키기 어려움. 그렇기 때문에 각 부분을 만들고 단계적으로 조립.

Architecture, Modeling, Design Pattern, Creational

Director

```
/* Builder 역할의 인터페이스(API)를 사용해서 인스턴스 조립/생성.*/
public class Director {
    private Builder builder;
    public Director(Builder builder) {
        this.builder = builder;
    }
    public void construct() {
        builder.setHeader();
        builder.setMessage("Hello", "World!", "안녕?","세상아");
        builder.setFooter();
    }
}
```

Builder, ConcreteBuilder

```
/* 인스턴스의 각 부분을 만들기 위한 메소드 준비 */
public abstract class Builder {
    protected StringBuffer buffer = new StringBuffer();
    public abstract void setHeader();
    public abstract void setMessage(String... items);
    public abstract void setFooter();
}
```

```
public class PlainTextBuilder extends Builder {

    @Override
    public void setHeader() {
        buffer.append("====================\n");
    }

    @Override
    public void setMessage(String... items) {
```

```
        Stream.of(items).forEach(str -> {
            buffer.append(str);
            buffer.append("\n");
        });
    }

    public void print() {
        System.out.println(buffer);
        System.out.println("\n");
    }

    @Override
    public void setFooter() {
        buffer.append("====================");
    }
}
```

```
public class JavaCodeBuilder extends Builder {

    private String snippet1 = "public class JavaCodeBuilder {\r\n" +
            "    public static void main(String[] args) {\r\n" +
            "        System.out.println(";
    private String snippet2 = ");\r\n" +
            "    }\r\n" +
            "}";
    @Override
    public void setHeader() {
        buffer.append(snippet1);
    }

    @Override
    public void setMessage(String... items) {
        buffer.append("\"");
        Stream.of(items).forEach(str -> {
            buffer.append(str);
            buffer.append(" ");
        });
        buffer.append("\"");
    }
    @Override
    public void setFooter() {
        buffer.append(snippet2);
    }

    public void save() {
        String fileName = "JavaCodeBuilder.java";
        System.out.println(buffer);
        System.out.println();
```

```
        System.out.println(fileName + " 로 저장되었습니다.");
        try (BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(fileName))){
            bos.write(String.valueOf(buffer).getBytes());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Client

```
public class Main {
    public static void main(String[] args) {
        PlainTextBuilder plainTextBuilder = new PlainTextBuilder();
        construct(plainTextBuilder);
        plainTextBuilder.print();
        JavaCodeBuilder javaCodeBuilder = new JavaCodeBuilder();
        construct(javaCodeBuilder);
        javaCodeBuilder.save();
    }
    private static void construct(Builder concreteBuilder) {
        Director plainTextDirector = new Director(concreteBuilder);
        plainTextDirector.construct();
    }
}


/*
====================
Hello
World!
안녕?
세상아
====================


public class JavaCodeBuilder {
    public static void main(String[] args) {
        System.out.println("Hello World! 안녕? 세상아 ");
    }
}

JavaCodeBuilder.java 로 저장되었습니다.
*/
```

From:
https://wiki.ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=design_pattern:builder_pattern

Last update: **2021/02/07 03:25**