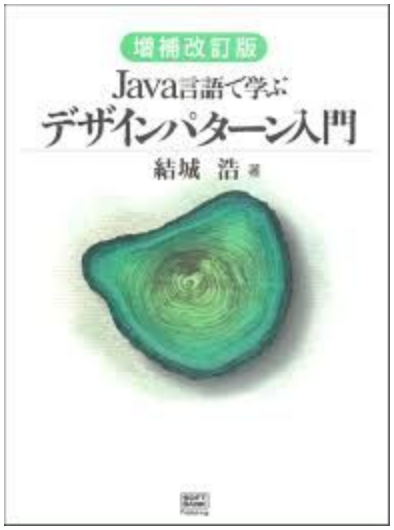
Java言語で学ぶデザインパターン入門

Template Method
具体的な処理をサブクラスにまかせる



テンプレートメソッドパターンとは

スーパークラスで処理の枠組みを定め、サブクラスでその具体的内容を定めるようなデザインパターン

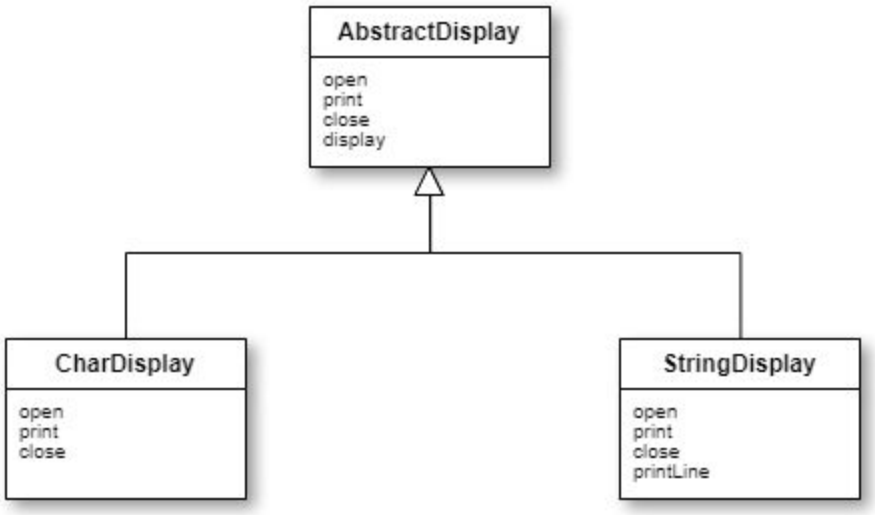
サンプルプログラム

「文字や文字列を５回繰り返して表示する」

クラス一覧

名前	解説
AbstractDisplay	メソッドdisplayのみ実装されているクラス
CharDisplay	open、print、closeを実装しているクラス
StringDisplay	open、print、closeを実装しているクラス
Main	動作テスト用のクラス

クラス図



ソースコード

```
public abstract class AbstractDisplay {
	public abstract void open();
	public abstract void print();
	public abstract void close();
```

```
	public final void display() {
		open();
		for(int i=0; i<5; i++) {
			print();
		}
		close();
	}
}
```

AbstractDisplayクラス

open, close, printの各メソッドで何をするかは、サブクラスに任せられている。

CharDisplayクラス

メソッド名	処理
open	文字列”<<”を表示する
print	コンストラクタで与えられていた１文字を表示する
close	文字列”>>”を表示する

例えばコンストラクタに’H’という文字が渡されていたとすると、

<<HHHHH>>

という文字列が表示されることになる。

```
public class CharDisplay extends AbstractDisplay{
	private String string;

	public CharDisplay(String string) {
		this.string = string;
	}
	public void open() {
		System.out.print("<<");
	}
```

```
	public void print() {
		System.out.print(string);
	}

	public void close() {
		System.out.println(">>");
	}

}
```

CharDisplayクラス

StringDisplayクラス

メソッド名	処理
open	文字列”+----+”を表示する
print	コンストラクタで与えられていた文字列を”|”で挟んで表示する
close	文字列”+----+”を表示する

```
public class StringDisplay extends AbstractDisplay {
	private String string;
	private int width;

	public StringDisplay(String string) {
		this.string = string;
		this.width = string.getBytes().length;
	}

	public void open() {
		printLine();
	}

	public void print() {
		System.out.println("|" + string + "|");
```

```
	}

	public void close() {
		printLine();
	}

	private void printLine() {
		System.out.print("+");
		for (int i = 0; i < width; i++) {
			System.out.print("-");
		}
		System.out.println("+");
	}
}
```

StringDisplayクラス

```
public class Main {
	public static void main(String[] args) {
		AbstractDisplay d1 = new CharDisplay("H");
		AbstractDisplay d2 = new StringDisplay("Hello, world.");
		AbstractDisplay d3 = new StringDisplay("こんにちは。");

		d1.display();
		d2.display();
		d3.display();
	}
}
```

Mainクラス

Template Methodを使う意味

Template Methodパターン使わずに、コピペで複数のConcreateClassを作ってしまったとする。ConcleateClass1,ConcleateClass2,ConcleateClass3,...すべて似て非なるクラスになる。このとき、ConcreateClass1にバグが見つかり修正するとすべてのConcreateClassに修正を反映

させなければならない。（今回の例でいうと、displayメソッド）Template Methodを使用していれば、テンプレートのみの修正で済む。