Template Method Pattern - 김리나

사전적 의미의 template?

> 마치 모양자. 동그라미, 세모 등의 다양한 모양이 이미 모양자에 박아져있고, 그 도형을 어떤 색 펜으로 그리느냐에 따라 결과가 달라진다.

템플릿 메소드 패턴

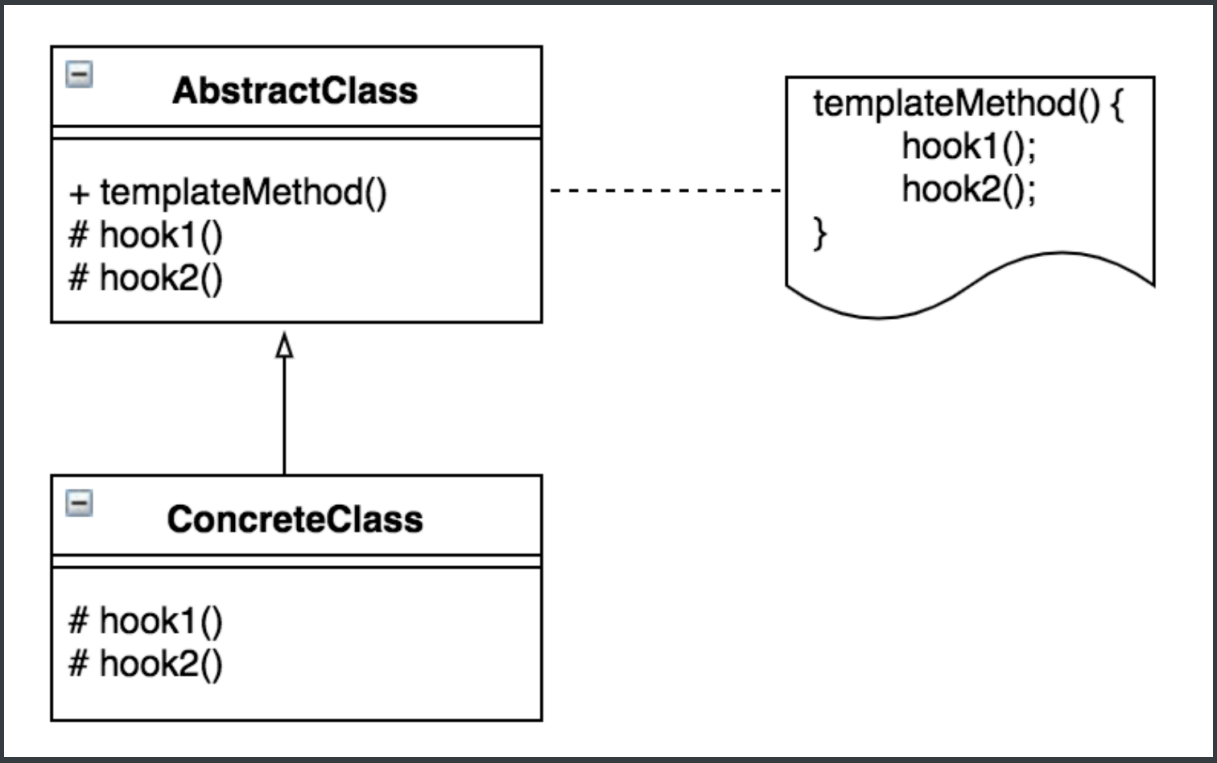
> - 알고리즘의 구조를 메소드에 정의하고 하위클래스에서 알고리즘 구조의 변경없이 알고리즘을 재정의하는 패턴
> - *Defines the skeleton of an algorithm in a method, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithms structure.*
> - 알고리즘의 구조를 메소드에 정의하고, 하위 클래스에서 알고리즘 구조의 변경없이 알고리즘을 재정의 하는 패턴이다. 알고리즘이 단계별로 나누어 지거나, 같은 역할을 하는 메소드이지만 여러곳에서 다른형태로 사용이 필요한 경우 유용한 패턴이다.
> - *상속을 통해 슈퍼클래스의 기능을 확장할 때 사용하는 가장 대표적인 방법. 변하지 않는 기능은 슈퍼클래스에 만들어두고 자주 변경되며 확장할 기능은 서브클래스에서 만들도록 한다*
>
> [사용하는 경우]
>
> - 구현하려는 알고리즘이 일정한 프로세스가 있을 때 **(단계가 있을때)** - 단계를 부모클래스에서 골격을 잡아줌으로서 유지보수에 용이

> - 구현하려는 알고리즘이 변경 가능성이 클 때 - 서브클래스에서 재정의하여 빈번하게 수정할때 용이

사용방법

> - 알고리즘을 여러 단계로 나눈다.
> - 나눠진 알고리즘의 단계를 메소드로 선언한다.
> - 알고리즘을 수행할 템플릿 메소드를 만든다.
> - 하위 클래스에서 나눠진 메소드들을 구현한다.

기본 설계



- - 하위클래스 concrete class에서 메소드 1, 2, 3을 구현해준다.
- 요구사항
  - 신작 게임의 접속을 구현해주세요.
    - requestConnection(String str) : String
  - 유저가 게임 접속시 다음을 고려해야합니다.
    - 보안과정 : 보안 관련 부분을 처리합니다.
      - doSecurity(String string) : String
    - 인증과정 : user name과 password가 일치하는지 확인합니다.
      - authentication(String id, String password) : boolean

        - 권한과정 : 접속자가 유료 회원인지 무료회원인지 게임 마스터인지 확인합니다.
            - authorization(String userName) : int
        - 접속과정 : 접속자에게 커넥션의 정보를 넘겨줍니다.
            - connection(String info) : String
- 더 공부해봅시다.
    - 보안 부분이 정부 정책에 의해서 강화되었다. 강화된 방식으로 코드를 변경해야한다.
    - 여가부에서 밤 10시 이후에 접속이 제한되도록 한다.
- => 기능이 다 나뉘어져있으니 유지보수가 편해짐.
- 접속이라는 template method 내에 네 가지 단계를 호출한다.

실습

Super Class

```
package TemplateMethodPattern;

public abstract class AbstGameConnectionHelper {

  // 하위클래스가 재정의를 해야하므로 private은 안되지만 외부에 노출되진 않도록 protected 사용
  protected abstract String doSecurity(String string); // 보안

  protected abstract boolean authentication(String id, String password);// 인증

  protected abstract int authorization(String userName); //권한

  protected abstract String connection(String info);// 접속

  // 접속을 도와주는 템플릿 메소드
  public String requestConnection(String encodedInfo) {
    // 보안과정 -> 암호화 된 문자열을 디코드한다.
    String decodedInfo = doSecurity(encodedInfo);

    // 반환된 것을 가지고 아이디, 패스워드를 할당한다.
    String id = "aaa";
    String password = "bbb";

    // 인증과정 -> db에 있는 아이디, 패스워드와 비교해 인증 여부 결정
    if(!authentication(id, password)) {
```

```
      throw new Error("아이디 암호 불일치");
    }
    // 권한 부여 과정 -> 게임 매니저, 유로회원, 무료회원 등으로 나누어 서로 다른 권한을 부여할
수 있다.
    String userName = "";
    int i = authorization(userName);

    switch (i) {
    case -1:
      throw new Error("셧다운");// 나이때문에 10시지나면 셧다운
    case 0:
      System.out.println("게임 매니저 권한");
      break;
    case 1 :
      System.out.println("유료회원 권한 ");
      break;
    case 2 :
      System.out.println("무료회원 권한 ");
      break;
    case 3 :
      System.out.println("권한 없음 ");
    default:// 기타상황
      break;
    }

    // 접속 과정
    return connection(decodedInfo);
  }
}
```

SubClass

```
package TemplateMethodPattern;

public abstract class DefaultGameConnectionHelper extends
AbstGameConnectionHelper {
// 틀은 추상클래스에서,구체적 구현은 하위클래스에서
  @Override
  protected String doSecurity(String string) {
    System.out.println("강화된 알고리즘을 이용한 디코드");
    return string;
  }
```

```
    @Override
    protected boolean authentication(String id, String password) {
      System.out.println("아이디, 암호 확인 과정 ");
      return true;
    }

    @Override
    protected int authorization(String userName) {
      System.out.println("권한 확인 ");
      // 서버에서 유저 이름을 가지고 유저의 나이를 알 수 있다.
      // 나이를 확인하고 지금 시간을 확인하고 미성년자에 10시가 지났다면
      // 권한이 없는 것으로 한다.
      return 0;
    }

    @Override
    protected String connection(String info) {
      System.out.println("마지막 접속 단계!");
      return info;
    }

  }
```

Main

```
  package TemplateMethodPattern;

  public class Main {

    public static void main(String[] args) {
      AbstGameConnectionHelper helper = new DefaultGameConnectionHelper() {
      };

      helper.requestConnection("아이디 암호 등 접속 정보 ");

    }
  }
```

JDK8 에서의 사용 예제

- AbstractMap<K,V> 클래스를 보면 템플릿 메소드 패턴이 적용된 것을 볼 수 있다. 토비의 스프링에 나온 설명처럼 "변하지 않는 기능은 슈퍼클래스에 만들어두고 자주 변경되며 확장할 기능은 서브클래스에서 만들도록 한다" 는 관점에서 보면, hashCode() 메소드는 AbstractMap 추상클래스에 있는 것을 사용하고, get() 등 기타 많은 메소드들은 HashMap, TreeMap 등 서브클래스에서 오버라이드하여 재정의 하고 있는 것을 볼 수 있다. 즉, AbstractMap 추상클래스를 상속받은 HashMap, TreeMap은 똑같이 get() 메소드를 가지지만 각자 자신만의 구현 방법으로 서로 다른 방식으로 구현되있다.
- Hook 메소드 : 슈퍼클래스에서 디폴트 기능을 정의해두거나 비워뒀다가 서브클래스에서 선택적으로 오버라이드 할 수 있도록 만들어둔 메소드. get()이 그러하다.