

# 소프트웨어 분석 설계

## 14주차

20170677 오윙택

### 1. 연습문제 1번

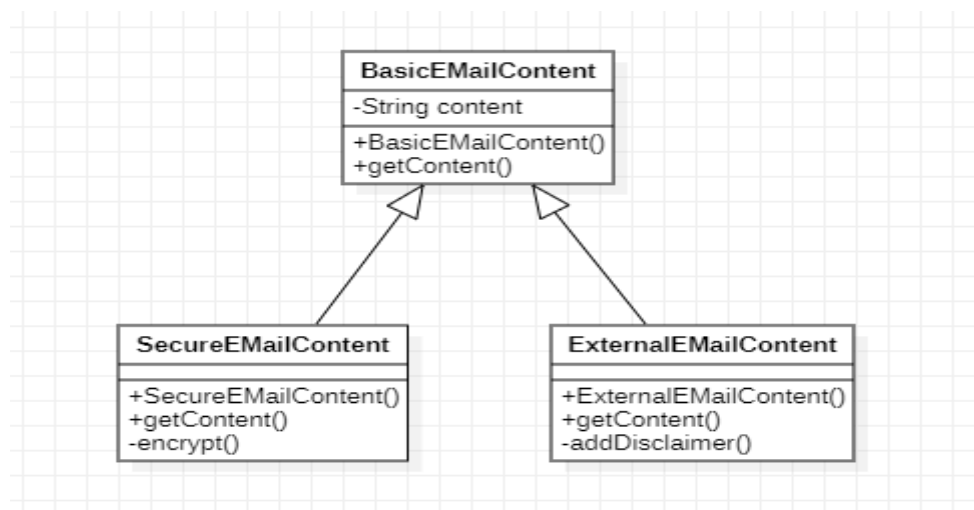
#### 1) 각 클래스의 기능 설명하기

BasicEmailContent : 기본적으로 문자열 Content를 반환하는 클래스

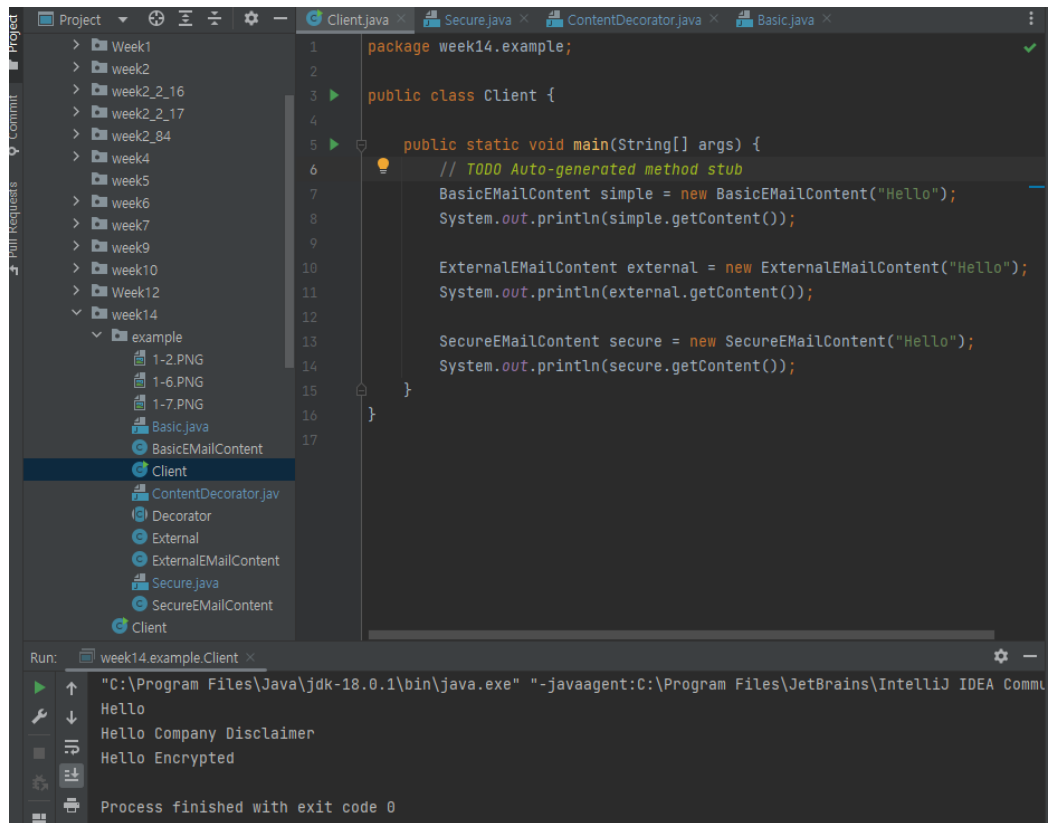
ExternalEmailContent : BasicEmailContent을 상속받아 추가적으로 Disclaimer라는 문자열을 추가하는 addDisclaimer 라는 메소드를 확장하는 자식 클래스

SecureEmailContent : BasicEmailContent을 상속받아 암호화했다는 문자열을 출력하는 encrypt 메소드를 추가적으로 가지고 있는 자식 클래스

#### 2) 클래스 다이어그램 작성하기



### 3) Client코드 실행 결과 작성하기



The screenshot shows the IntelliJ IDEA IDE. The left sidebar displays a project structure with folders 'Week1' through 'Week14' and a subfolder 'example' containing files like '1-2.PNG', '1-6.PNG', '1-7.PNG', 'Basic.java', 'BasicEmailContent', 'Client', 'ContentDecorator.java', 'Decorator', 'External', 'ExternalEmailContent', 'Secure.java', 'SecureEmailContent', and 'Client'. The main editor window shows the code for 'Client.java' in the 'week14.example' package. The code defines a 'Client' class with a 'main' method that creates and prints the content of 'BasicEmailContent', 'ExternalEmailContent', and 'SecureEmailContent' objects. The bottom 'Run' window shows the output of the program: 'Hello', 'Hello Company Disclaimer', and 'Hello Encrypted', followed by 'Process finished with exit code 0'.

```
package week14.example;

public class Client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        BasicEmailContent simple = new BasicEmailContent("Hello");
        System.out.println(simple.getContent());

        ExternalEmailContent external = new ExternalEmailContent("Hello");
        System.out.println(external.getContent());

        SecureEmailContent secure = new SecureEmailContent("Hello");
        System.out.println(secure.getContent());
    }
}
```

Run: week14.example.Client ×

"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commu  
Hello  
Hello Company Disclaimer  
Hello Encrypted  
Process finished with exit code 0

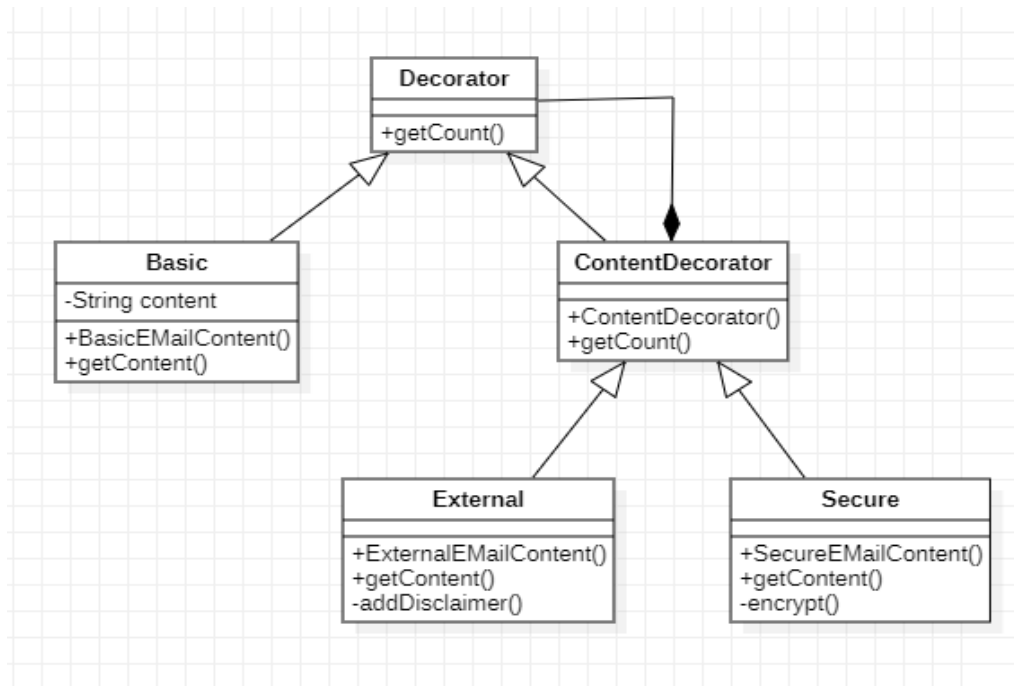
### 4) OCP위반 설명하기

XX

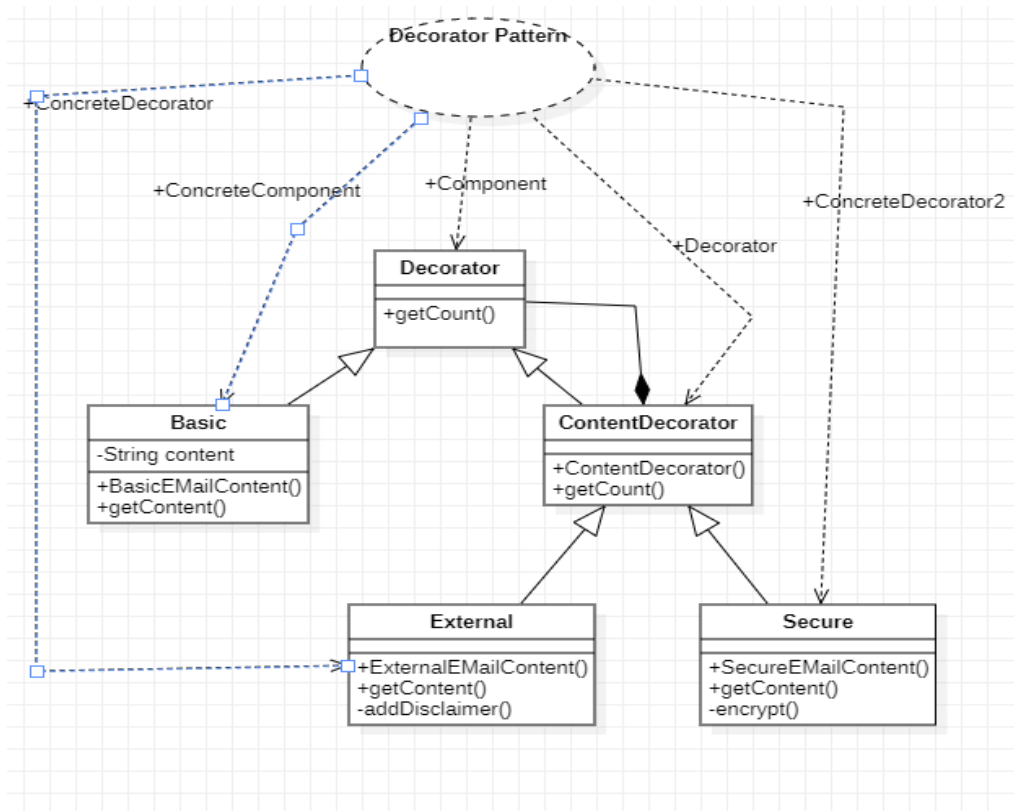
### 5) 데커레이터 패턴을 활용해 취약점 개선하기

데커레이터 패턴을 활용하면 필요한 기능을 추가적으로 조합하여 합체로봇 만들듯이 인스턴스를 찍어낼 수 있다. 이렇게 하면 새로 생긴 기능(Ex 번역하기)와 같은 기능을 OCP위배 없이 추가할 수 있다.

6) 개선된 설계를 클래스 다이어그램으로 표현하고 간단히 설명하기



7) Meta 다이어그램으로 표현하기



## 8) 코드 작성하기

Decorator.java

```
package week14.example;

public abstract class Decorator {
    public abstract String getContent();
}
```

Basic.java

```
package week14.example;

public class Basic extends Decorator{
    private String content;

    public Basic(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }
}
```

ContentDecorator.java

```
package week14.example;

public abstract class ContentDecorator extends Decorator{
    private Decorator decorator;

    public ContentDecorator(Decorator content) {
        this.decorator = content;
    }

    public String getContent() {
        return decorator.getContent();
    }
}
```

## External.java

```
package week14.example;

public class External extends ContentDecorator{

    public External(Decorator decorator) {
        super(decorator);
    }

    public String getContent() {
        String content = super.getContent();
        String external = addDisClaimer(content);
        return external;
    }

    private String addDisClaimer(String content) {
        return content + " Company Disclaimer";
    }
}
```

## Secure.java

```
package week14.example;

public class Secure extends ContentDecorator{

    public Secure(Decorator decorator) {
        super(decorator);
    }

    public String getContent() {
        String content = super.getContent();
        String encryptContent = encrypt(content);
        return encryptContent;
    }

    private String encrypt(String content) {
        return content + " Encryped";
    }
}
```

## 9) Client 클래스 재작성하고 Decorator패턴 추가하기

Transcribe.java

```
package week14.example;

public class Transcribe extends ContentDecorator{

    public Transcribe(Decorator content) {
        super(content);
    }

    public String getContent() {
        String content = super.getContent();
        String translate = Translate(content);
        return translate;
    }

    private String Translate(String content) {
        return content + "has been Translated";
    }
}
```

Client.java

The screenshot shows an IDE with the following components:

- Project Explorer:** A tree view on the left showing a project named 'example'. It contains several image files (1-2.PNG, 1-6.PNG, 1-7.PNG) and Java classes: Basic, BasicEmailContent, Client, ContentDecorator, Decorator, External, ExternalEmailContent, Secure, SecureEmailContent, Transcribe, Client (highlighted), CrossingDisplay, Display, DisplayDecorator, LaneDecorator, RoadDisplay, and TrafficDecorator. Below this, there is a 'week15' folder containing 'module-info.java', 'Week1\_fix.zip', and 'week6.zip', followed by '1-1.png' and '1-2.1.png'.
- Code Editor:** The main window shows the 'Client.java' file. The code is as follows:

```
package week14.example;

public class Client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Decorator simple = new Basic( content: "Hello");
        System.out.println(simple.getContent());

        Decorator external = new External(simple);
        System.out.println(external.getContent());

        Decorator secure = new Secure(simple);
        System.out.println(secure.getContent());

        Decorator translate = new Transcribe(simple);
        System.out.println(translate.getContent());
    }
}
```
- Run Console:** At the bottom, the output of the program is shown:

```
Run: week14.example.Client x
"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea-agent.jar" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA\config -Didea.home.path=C:\Program Files\JetBrains\IntelliJ IDEA\bin -Didea.platform.prefix=Java -Didea.vendor.id=IntelliJ -Didea.version=2023.2.4 -Didea.welcome.url=https://www.jetbrains.com/idea/ -Didea.welcome.show=true -Didea.welcome.show.message=Hello
Hello
Hello Company Disclaimer
Hello Encrypted
Hellohas been Translated
Process finished with exit code 0
```