UML

20509김연준

목차.

1.UML

UML은 Unified Modeling Language의 약자로 1997년 OMG(Object Management Group)에서 표준으로 채택한 통합모델링언어 입니다

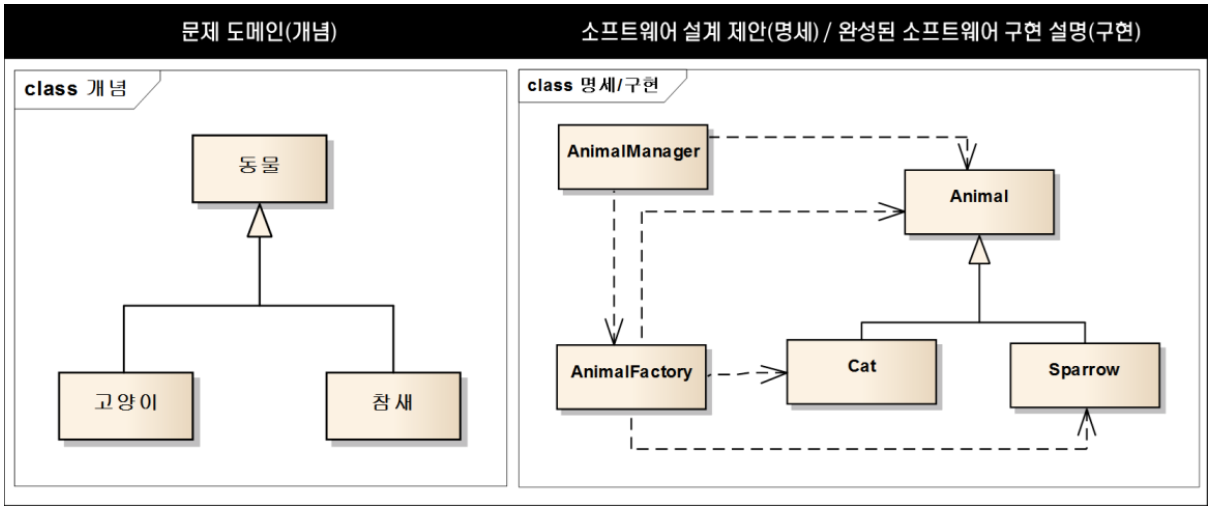
객체 지향언어의 개발을 용이하게 바꾸었으며 여러 주요 객체지향 개발 방법론과 잘 어울리도록 설계되었습니다

UML를 사용하는 유형은 3가지로 나뉘는데

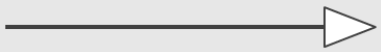
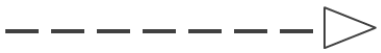
1. 다른 사람들과의 의사소통 또는 설계논의
2. 전체 시스템의 구조 및 클래스의 의존성 파악
3. 유지보수를 위한 설계의 back-end 문서

UML은 구조 다이어그램 7개, 행위 다이어그램 7개로 총 14종류의 다이어그램이 있습니다. 구조 다이어그램은 시스템의 개념, 관계 등의 측면에서 요소들을 나타내고 각 요소들의 정적인 면을 보기 위한 것이고 행위 다이어그램은 각 요소들 혹은 요소들간의 변화나 흐름, 주고받는 데이터 등의 동작을 보기 위한 것으로, 클래스 다이어그램은 구조 다이어그램에 해당합니다. 클래스 다이어그램은 클래스 내부의 정적인 내용이나 클래스 사이의 관계를 표기하는 다이어그램으로 시스템의 일부 또는 전체의 구조를 나타낼 수 있습니다. 클래스 다이어그램은 의존 관계를 명확히 보게 해주며, 순환 의존이 발생하는 지점을 찾아내서 어떻게 이 순환 고리를 깨는 것이 가장 좋은지 결정할 수 있게 해줍니다.

예를 들어 클래스를 나타내며 이렇게 나타낼수 있습니다


문제 도메인(개념)
소프트웨어 설계 제안(명세) / 완성된 소프트웨어 구현 설명(구현)
class 개념
동물
고양이
참새
class 명세/구현
AnimalManager
Animal
AnimalFactory
Cat
Sparrow

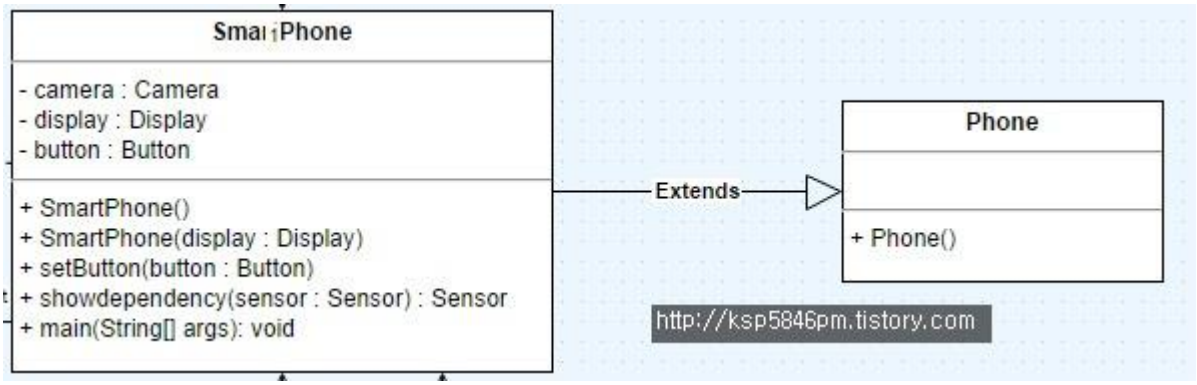
2.클래스 다이어그램

관계	UML 표기
Generalization (일반화)	
Realization (실체화)	
Dependency (의존)	
Association (연관)	
Directed Association (직접연관)	
Aggregation (집합, 집합연관)	
Composition (합성, 복합연관)	

위의 그림은 클래스간의 관계들의 종류와 표기법을 나타낸 것입니다.

클래스 다이어그램 일반화

```
public class SmartPhone extends Phone {
private Camera camera;
private Display display;
private Button button;
public SmartPhone() {
 System.out.println("카메라는 Composition");
        this.camera= new Camera(); //객체를 내부에서 생성
        }
        public static void main (String[] args){
          SmartPhone smartphone = new SmartPhone();
        }

 }
```

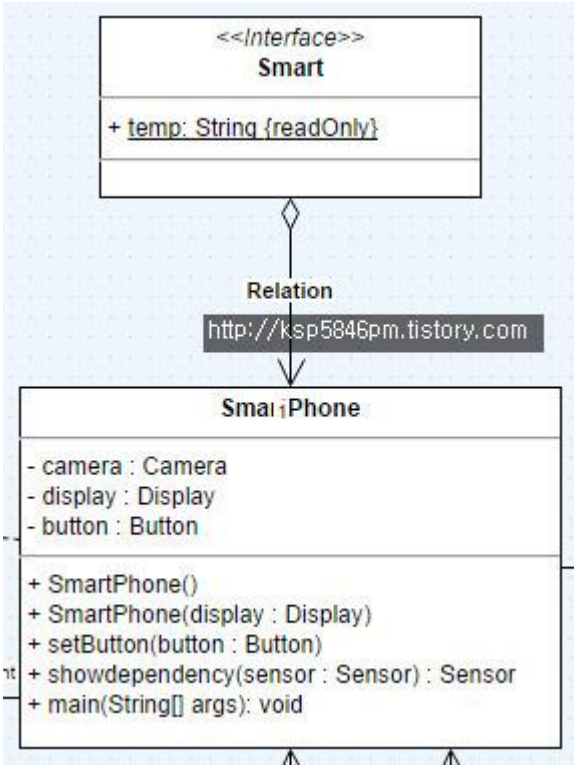


슈퍼(부모)클래스와 서브(자식)클래스간의 Inheritance(상속) 관계를 나타냅니다

클래스 다이어 그램 실체화

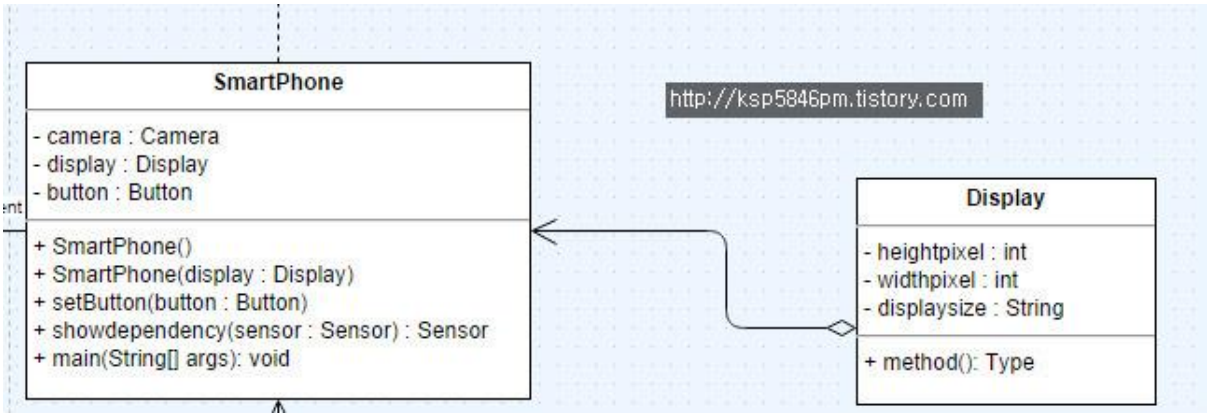
interface의 spec(명세, 정의)만 있는 메서드를 오버라이딩 하여 실제 기능으로 구현 하는 것을 말합니다.

```
public class SmartPhone implements Smart {
                private Camera camera;
                private Display display;
                private Button button;

                public SmartPhone() {
                    System.out.println("카메라는 Composition");
                    this.camera= new Camera();
                }

                public static void main (String[] args){
                    SmartPhone smartphone = new SmartPhone();

                }
}
```


<<Interface>>
Smart
+ temp: String {readOnly}
Relation
http://ksp5846pm.tistory.com
SmartPhone
- camera : Camera
- display : Display
- button : Button
+ SmartPhone()
+ SmartPhone(display : Display)
+ setButton(button : Button)
+ showdependency(sensor : Sensor) : Sensor
+ main(String[] args): void

composition(합성, 또는 강한 결합)

```
class Camera {
private int spec;
private String company;
Camera() {
this.company = "삼성";
System.out.println("카메라 생성자호출");
System.out.println("getCompany메소드 호출");
}
public void getCompnayName() {
System.out.println("회사이름은"+this.company);
}
}; public class SmartPhone {
private Camera camera;
//스마트폰 구성요소중 카메라
public SmartPhone() {
System.out.println("카메라는 Composition");
this.camera= new Camera();
//객체를 내부에서 생성
} //메인함수
 public static void main (String[] args){
System.out.println("-----Composition------");
SmartPhone smartphone = new SmartPhone(); } }
```


SmartPhone
- camera : Camera
- display : Display
- button : Button
+ SmartPhone()
+ SmartPhone(display : Display)
+ setButton(button : Button)
+ showdependency(sensor : Sensor) : Sensor
+ main(String[] args): void
http://ksp5846pm.tistory.com
Display
- heightpixel : int
- widthpixel : int
- displaysize : String
+ method(): Type

느낌점. 평소 TDD(테스트주도개발론)에 관심있어 많이 찾아보고 그랬었는데 새로운 개발방법에 대하여 알게되어서 한번 해보고 싶긴한데 더 효율적인 방법이 있지않은가 생각이됩니다