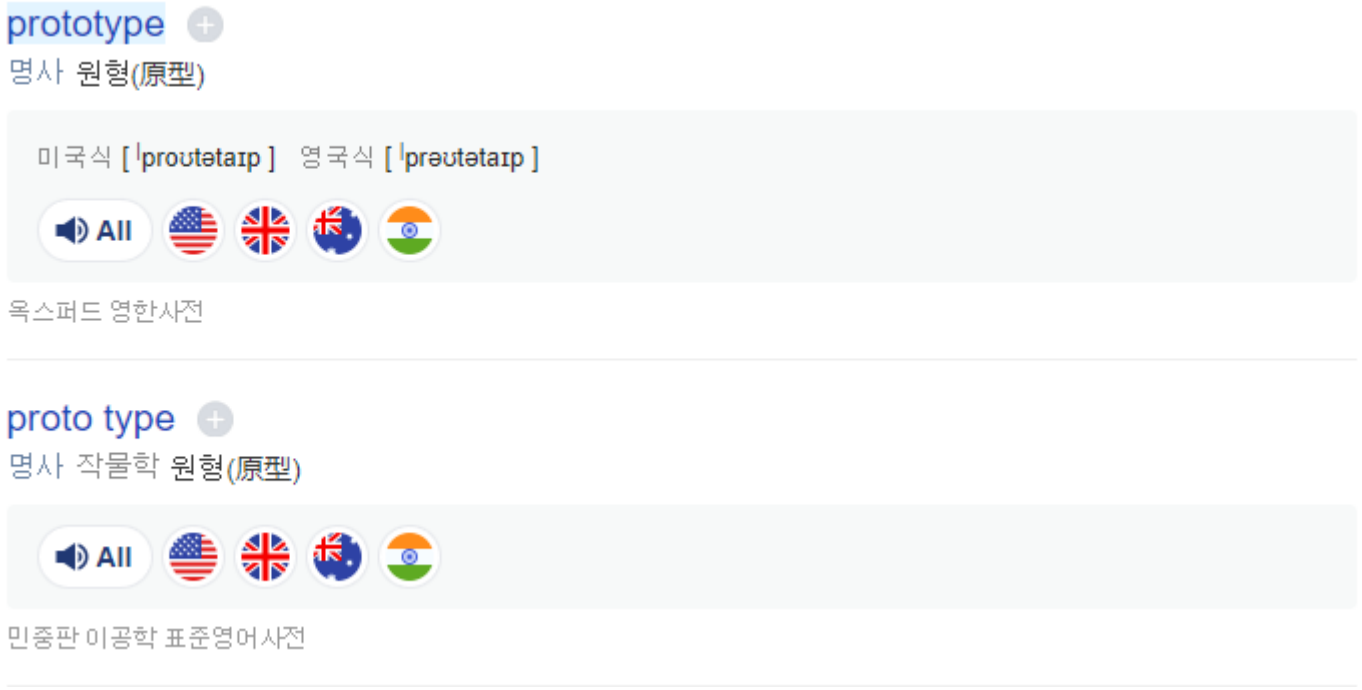
Prototype Pattern

김규현

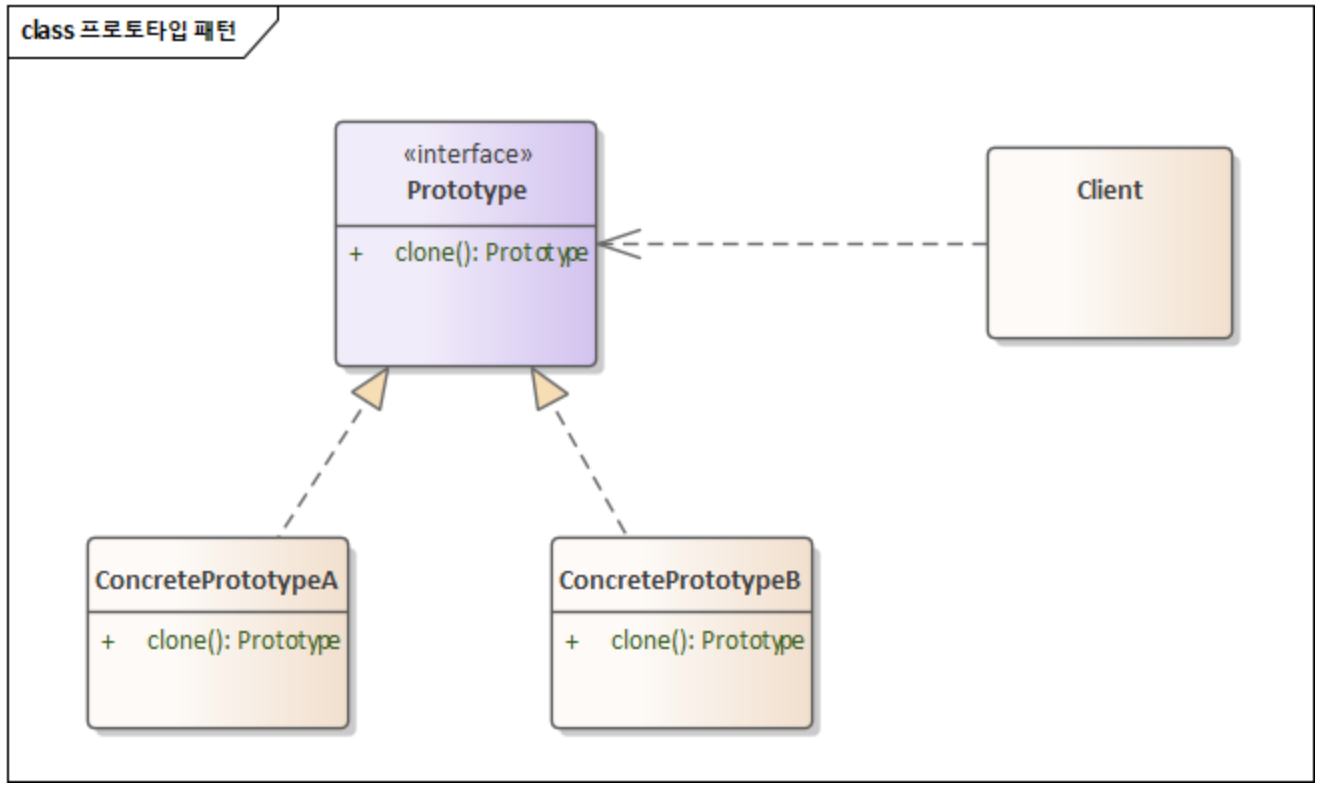
목차

Prototype 패턴이란?



하나의 인스턴스를 토대로 다른 인스턴스를 만들어내는 방법

Prototype 패턴이란? – 구현 방법



Clone() 이라는 추상 메서드를 가진 인터페이스를 구체 클래스가 구현한다.

Prototype 패턴이란? - 장단점

장점

복잡한 객체 생성 과정을 숨길 수 있다

기존 객체를 복제하는 과정이 새로운 인스턴스를 만드는 것 보다 효율적일 수 있다.

(ex. DB 질의를 해야 하는 경우 or Network 통신을 거쳐야 하는 경우)

추상 타입을 리턴 할 수 있다.

단점

복제하는 객체를 만드는 과정 자체가 복잡할 수 있다. (특히 순환참조가 있는 경우)

Java에서 사용법 – Object.clone()

```
Returns:  a clone of this instance.

Throws:   CloneNotSupportedException  – if the object's class does not support the
          Cloneable  interface. Subclasses that override the  clone  method can also
          throw this exception to indicate that an instance cannot be cloned.

See Also: Cloneable

protected native Object clone() throws CloneNotSupportedException;
```

```
public class ConcretePrototype {
    public ConcretePrototype copy(){
        return this.clone();
    }
}
```

딸깍

Java에서 사용법 - 주의점

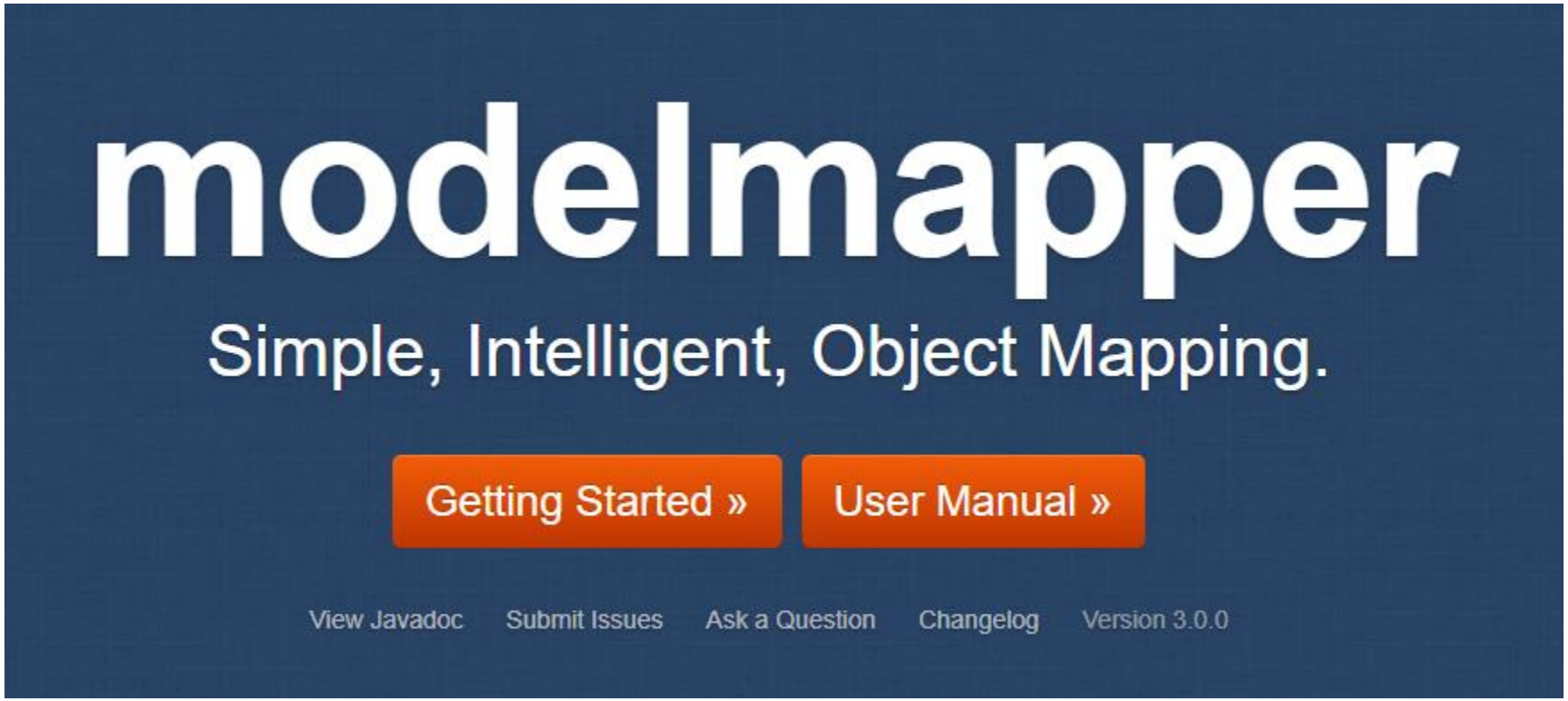
주의

Clonable 인터페이스 없이 clone() 메서드를 이용할 경우 CloneNotSupportedException이 발생

clone() 메서드는 기본적으로 얕은 복사를 하기 때문에, 깊은 복사가 필요하다면 본인이 재정의 해줘야 함.

```
public class ConcretePrototype implements Cloneable{
    public ConcretePrototype copy() throws CloneNotSupportedException {
        return (ConcretePrototype) this.clone();
    }
}
```

실제 사용 예시



ModelMapper 라이브러리 리플렉션을 활용해서 프로토타입 패턴 구현

실제 사용 예시 – 경험담

```
    public void copy() {
        if(this.selectedShape!=null) {
        this.copiedShape = this.selectedShape.clone();
        this.pastePosition = 0;
        }
    }
    public void paste() {
        if(this.copiedShape!=null) {
            // paste ?Ч? ??¦ U???????.
        this.pastePosition += 5;
        GShapeTool tempShape = this.copiedShape.clone();
        tempShape.move((Graphics2D) this.getGraphics(), pastePosition,
pastePosition);
        shapes.add(tempShape);
        this.undoStack.push(this.deepCopy(this.shapes));
        setSelected(tempShape);
        this.repaint();
        }
    }
```