팩토리 패턴

팩토리 패턴이란?

어떤 상황에서 조건에 따라 객체를 다르게 생성해야 할 때가 있습니다. 분기 : if, else if …
예를 들면, 사용자의 입력값에 따라 하는 일이 달라질 경우, 분기를 통해 특정 객체를 생성해야 합니다.
객체마다 하는 일이 다르기 때문에 조건문에 따라 객체를 다르게 생성하는 것입니다.

팩토리 메서드 패턴은 이렇게 분기에 따른 객체의 생성(new 연산자로 객체를 생성하는 부분)을 직접 하지 않고,
팩토리라는 클래스에 위임하여 팩토리 클래스가 객체를 생성하도록 하는 방식을 말합니다.
팩토리는 말 그대로 객체를 찍어내는 공장을 의미합니다.

팩토리 패턴의 장단점

장점

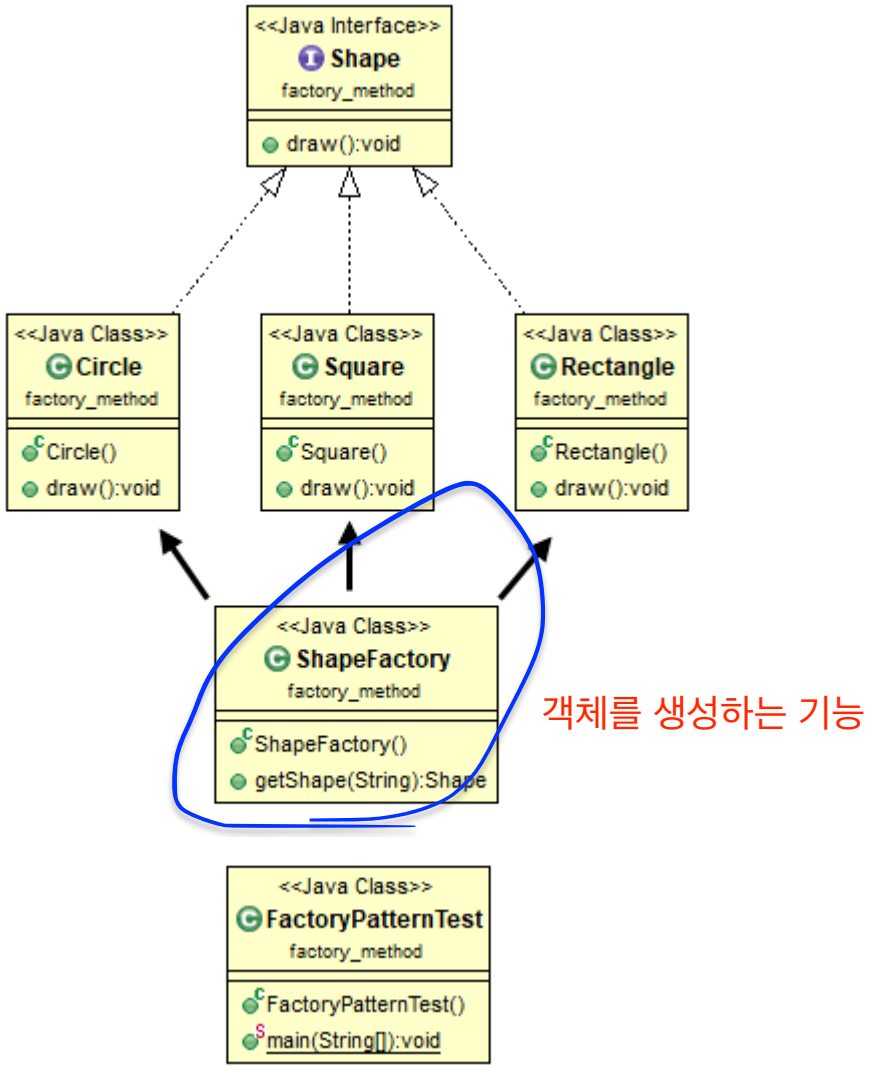
구현하는 로직끼리 엉키는 일이 적어진다.

객체 생성을 팩토리 클래스에 위임하므로, 객체 간의 결합도가 낮아지고 유지보수(확장, 수정)에 용이하다.

단점

객체가 늘어 날 때마다 하위 클래스 재정의로 인해 불필요한 클래스가 많이 생성될 수 있음.

팩토리 패턴의 구조

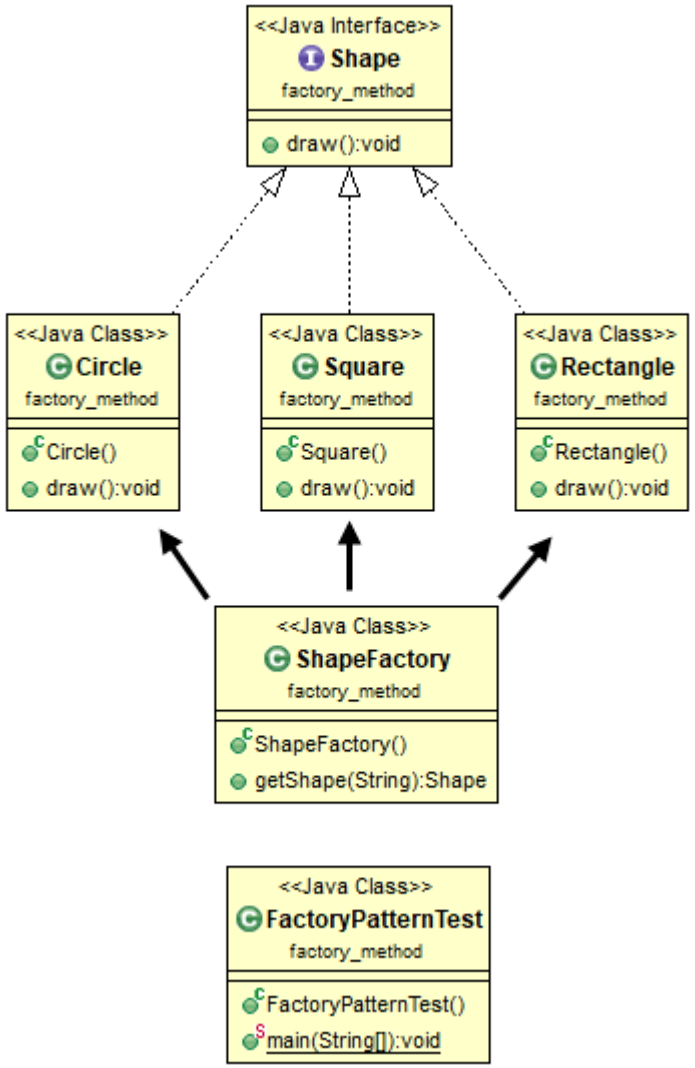


객체를 생성하는 기능

Shape.java

```java
public interface Shape {
    void draw();
}
```

팩토리 패턴의 구조




```java
public class Circle implements Shape{
    @Override
    public void draw() {
        System.out.println("Circle - draw() Method.");
    }
}
```

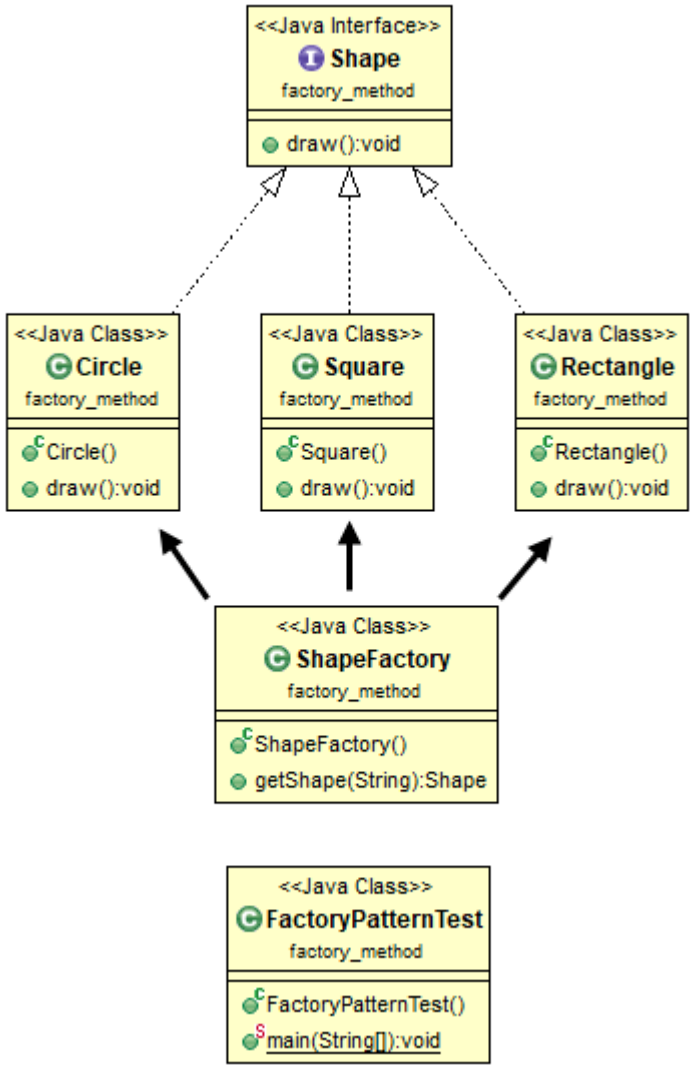


```java
public class Rectangle implements Shape{
    @Override
    public void draw() {
        System.out.println("Rectangle - draw() Method.");
    }
}
```



```java
public class Square implements Shape{
    @Override
    public void draw() {
        System.out.println("Square - draw() Method.");
    }
}
```

팩토리 패턴의 구조

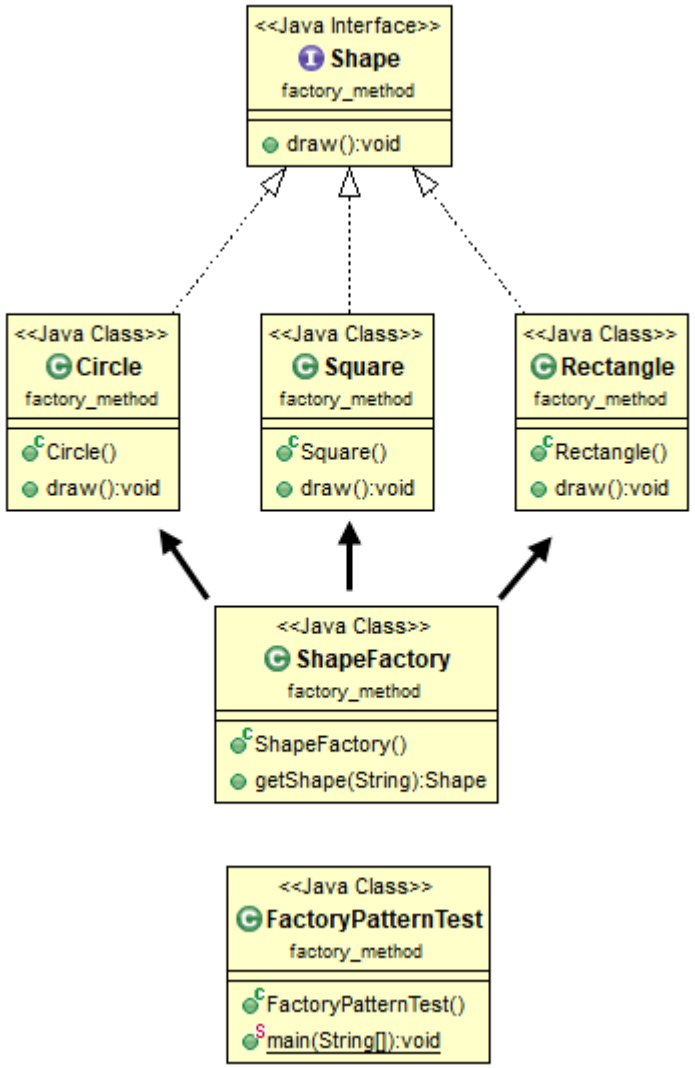


ShapeFactory.java

분기를 팩토리 클래스에서 따로 다 만들어줌

```java
public class ShapeFactory {

   //팩토리 메소드 - 객체 생성 후 반환
   public Shape getShape(String shapeType){
      if(shapeType == null){
         return null;
      }
      if(shapeType.equalsIgnoreCase("CIRCLE")){
         return new Circle();

      } else if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new Rectangle();

      } else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new Square();
      }

      return null;
   }
}
```

팩토리 패턴의 구조



FactoryPatternTest.java

```java
public class FactoryPatternTest {
    public static void main(String[] args) {

        //팩토리 클래스에서 객체를 생성 후 반환
        ShapeFactory shapeFactory = new ShapeFactory();

        Shape shape1 = shapeFactory.getShape("CIRCLE");

        //Circle 메소드 호출
        shape1.draw();

        Shape shape2 = shapeFactory.getShape("RECTANGLE");

        //Rectangle 메소드 호출
        shape2.draw();

        Shape shape3 = shapeFactory.getShape("SQUARE");

        //Square 메소드 호출
        shape3.draw();
    }
}
```

활용 방안

입력 받는 값에 따라 다른 객체를 생성해야 한다면,
팩토리 패턴을 활용해서 확장과 수정을 손쉽게 해봅시다!