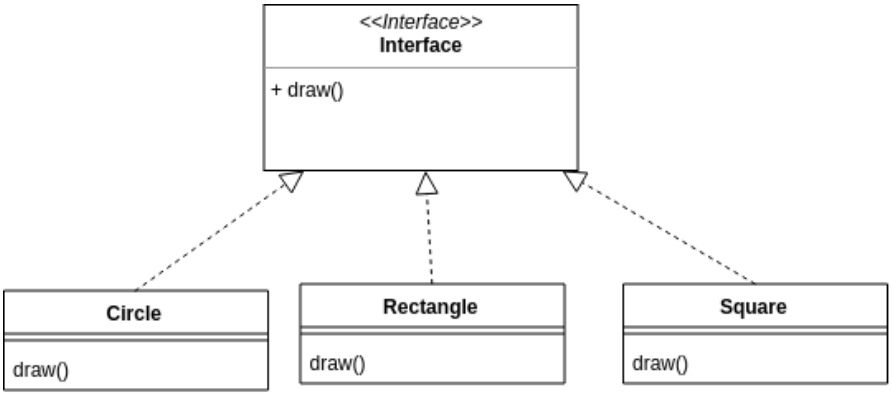
Patrón de diseño Factory en Java

| *Gustavo*

El patrón de diseño Factory es usado principalmente cuando tenemos una clase o interfaz con muchas subclases o implementaciones y según algún input necesitamos devolver una de estas subclases concretas.

Cuando crear una Factory class

Pensemos en una interfaz de la cual realizamos luego varias implementaciones. Usemos la típica interfaz de ejemplo Shape de la cual implementamos varias clases Circle, Rectangle, Square.




```
package patterns.factory;

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

```
package patterns.factory;

public class Circle implements Shape {

    @Override
    public void draw() {
        System.out.println("I am a Circle");
    }

}
```

```
package patterns.factory;

public class Rectangle implements Shape {

    @Override
    public void draw() {
        System.out.println("I am a Rectangle");
    }

}
```

```
package patterns.factory;

public class Square implements Shape {

    @Override
    public void draw() {
        System.out.println("I am a Square");
    }

}
```

En este caso tenemos varias implementaciones y según el tipo de Shape que deseamos queremos crear la implementación concreta de la interfaz.

La forma habitual (sin un factory) sería de este modo.

```
package patterns.factory;

public class FactoryPatternExample {

    public static void main(String[] args) {

        FactoryPatternExample example = new FactoryPatternExample();
        example.printShape("CIRCLE");

    }

    // Whithout factory
    public void printShape(String shapeType) {

        Shape shape;
        switch (shapeType) {
            case "CIRCLE":
                shape = new Circle();
                break;
            case "RECTANGLE":
                shape = new Rectangle();
                break;
            case "SQUARE":
                shape = new Square();
                break;
            default:
                throw new IllegalStateException("Unexpected value: " + shapeType);
        }
        shape.draw();

    }

}
```

Como crear una Factory class

Vemos en el ejemplo previo que podríamos simplificar la creación de la clase llevándonos todo lo que esta en el ***switch*** a otra clase Factory que se encargue de esto, dejando así nuestro código más simple y claro.

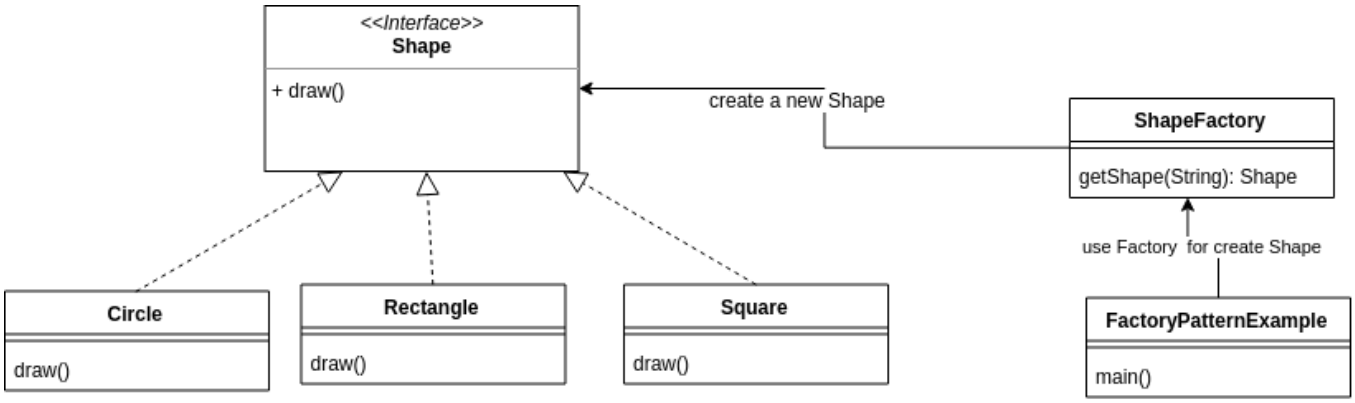
Vamos a crear una clase **ShapeFactory** para tal fin.

```
package patterns.factory;

public class ShapeFactory {

    public static 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();
        } else {
            throw new IllegalStateException("Unexpected value: " + shapeType);
        }

    }

}
```

Veamos cómo queda ahora nuestro ejemplo previo haciendo uso de nuestra Factory.

Observa que queda mucho más limpio y claro delegando la creación de las instancias a la Factory.

La Factory será ahora la encargada de crear la instancia en particular y nuestra clase main de ejemplo hará uso de esta factory olvidandose de la complejidad para crearla.



```
package patterns.factory;

public class FactoryPatternExample {

    public static void main(String[] args) {

        FactoryPatternExample example = new FactoryPatternExample();
        example.printShape("CIRCLE");

    }

    // Using Factory
    public void printShape(String shapeType) {

        Shape shape2 = ShapeFactory.getShape(shapeType);
        shape2.draw();

    }

}
```

Obviamente este es un ejemplo muy sencillo, en general las Factory class pueden ser más complejas encargándose de crear las instancias bajo condiciones de negocio o diversos inputs que hacen que su uso sea conveniente.

Puedes descargar este código en github (https://github.com/gustavopeiretti/java-examples) o en gitlab (https://gitlab.com/gustavopeiretti/java-examples)

> Hola! Si mis post te son útiles y te ayudan a aprender algo de ellos, considera brindar tu soporte invitándome un rico café. :) Muchas gracias!
>
> Buy me a coffee (http://buymeacoffee.com/NGtoizJlC)

Tags: #Java (https://gustavopeiretti.com/tags/java/) #pattern (https://gustavopeiretti.com/tags/pattern/)

← **ARTÍCULO ANTERIOR (HTTPS://GUSTAVOPEIRETTI.COM/PATRON-DE-DISENO-JAVA-SINGLETON/)**

ARTÍCULO SIGUIENTE → (HTTPS://GUSTAVOPEIRETTI.COM/PATRON-DE-DISENO-DECORATOR-EN-JAVA/)