Abstract Factory

```
public interface Shape {
	public void draw();
}
public class Circle implements Shape {
	@Override
	public void draw() {
		System.out.println("Inside Circle::draw() method.");
	}
}
public class Rectangle implements Shape {
	@Override
	public void draw() {
		System.out.println("Inside Rectangle::draw() method.");
	}
}
public class Square implements Shape {
	@Override
	public void draw() {
		System.out.println("Inside Square::draw() method.");
	}
}
public abstract class ShapeAbstractFactory {
	public abstract Shape createShape(String shapeType);
}
public class ShapeConcreteAbstractFactory extends ShapeAbstractFactory {
	@Override
	public Shape createShape(String shapeType) {
		if(shapeType == null) {
			return null;
		}
		if(shapeType.equals("circle")) {
			return new Circle();
		} else if(shapeType.equals("square")) {
			return new Square();
		} else if(shapeType.equals("rectangle")) {
			return new Rectangle();
		}
		return null;
	}
}
public class Main {
	public static void main(String[] args) {
		new Main().go();
	}
	void go() {
		ShapeAbstractFactory shapeFactory = new ShapeConcreteAbstractFactory();

		Shape shape1 = shapeFactory.createShape("circle");
		shape1.draw();

		Shape shape2 = shapeFactory.createShape("square");
		shape2.draw();

		Shape shape3 = shapeFactory.createShape("rectangle");
		shape3.draw();
	}
}
```