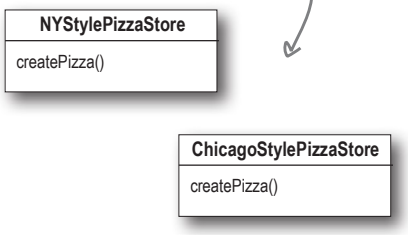
Declaring a factory method

With just a couple of transformations to the PizzaStore class, we've gone from having an object handle the instantiation of our concrete classes to a set of subclasses that are now taking on that responsibility. Let's take a closer look:

```
public abstract class PizzaStore {

    public Pizza orderPizza(String type) {
        Pizza pizza;

        pizza = createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }

    protected abstract Pizza createPizza(String type);

    // other methods here
}
```

The subclasses of PizzaStore handle object instantiation for us in the createPizza() method.



All the responsibility for instantiating Pizzas has been moved into a method that acts as a factory.

Code Up Close

A factory method handles object creation and encapsulates it in a subclass. This decouples the client code in the superclass from the object creation code in the subclass.

A factory method may be parameterized (or not) to select among several variations of a product.

```
abstract Product factoryMethod(String type)
```

A factory method is abstract so the subclasses are counted on to handle object creation.

A factory method returns a Product that is typically used within methods defined in the superclass.

A factory method isolates the client (the code in the superclass, like orderPizza()) from knowing what kind of concrete Product is actually created.