```
NYPizzaFactory nyFactory = new NYPizzaFactory();
PizzaStore nyStore = new PizzaStore(nyFactory);
nyStore.orderPizza("Veggie");
```

Here we create a factory for making NY-style pizzas.

Then we create a PizzaStore and pass it a reference to the NY factory.

...and when we make pizzas, we get NY-style pizzas.

```
ChicagoPizzaFactory chicagoFactory = new ChicagoPizzaFactory();
PizzaStore chicagoStore = new PizzaStore(chicagoFactory);
chicagoStore.orderPizza("Veggie");
```

Likewise for the Chicago pizza stores: we create a factory for Chicago pizzas and create a store that is composed with a Chicago factory. When we make pizzas, we get the Chicago-style ones.

But you'd like a little more quality control...

So you test-marketed the SimpleFactory idea, and what you found was that the franchises were using your factory to create pizzas, but starting to employ their own home-grown procedures for the rest of the process: they'd bake things a little differently, they'd forget to cut the pizza, and they'd use third-party boxes.

Rethinking the problem a bit, you see that what you'd really like to do is create a framework that ties the store and the pizza creation together, yet still allows things to remain flexible.

In our early code, before the SimplePizzaFactory, we had the pizza-making code tied to the PizzaStore, but it wasn't flexible. So, how can we have our pizza and eat it too?

Not what you want in a good franchise. You do NOT want to know what he puts on his pizzas.