Design Patterns, Factory method, Abstract factory and Singleton

Dr, Ayman Ezzat

ayman@fcih.net

New is concrete class

```
Duck duck;

if (picnic) {
    duck = new MallardDuck();
} else if (hunting) {
    duck = new DecoyDuck();
} else if (inBathTub) {
    duck = new RubberDuck();
}
```

We have a bunch of different duck classes, and we don't know until runtime which one we need to instantiate.

```
Pizza orderPizza(String type) {
    Pizza pizza;

    if (type.equals("cheese")) {
        pizza = new CheesePizza();
    } else if (type.equals("greek") {
        pizza = new GreekPizza();
    } else if (type.equals("pepperoni") {
        pizza = new PepperoniPizza();
    }

    pizza.prepare();
    pizza.bake();
    pizza.cut();
    pizza.box();
    return pizza;
}
```

Based on the type of pizza, we instantiate the correct concrete class and assign it to the pizza instance variable. Note that each pizza here has to implement the Pizza interface.

Once we have a Pizza, we prepare it (you know, roll the dough, put on the sauce and add the toppings & cheese), then we bake it, cut it and box it!

Each Pizza subtype (CheesePizza, VeggiePizza, etc.) knows how to prepare itself.

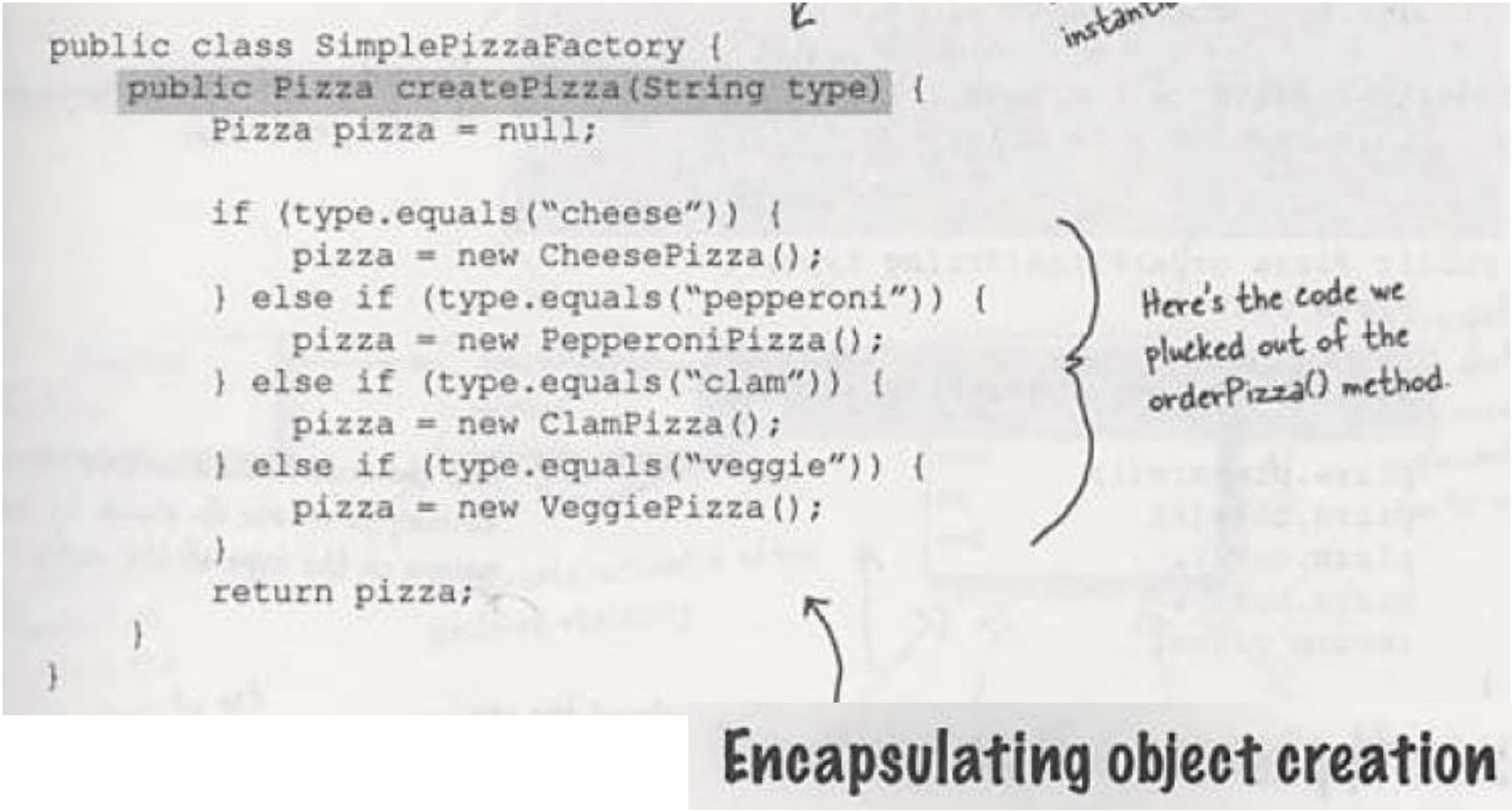
Change and Change Evolution!!

```
if (type.equals("cheese")) {
    pizza = new CheesePizza();
} else if (type.equals("greek") {
    pizza = new GreekPizza();
} else if (type.equals("pepperoni") {
    pizza = new PepperoniPizza();
} else if (type.equals("clam") {
    pizza = new ClamPizza();
} else if (type.equals("veggie") {
    pizza = new eggiePizza();
}
```

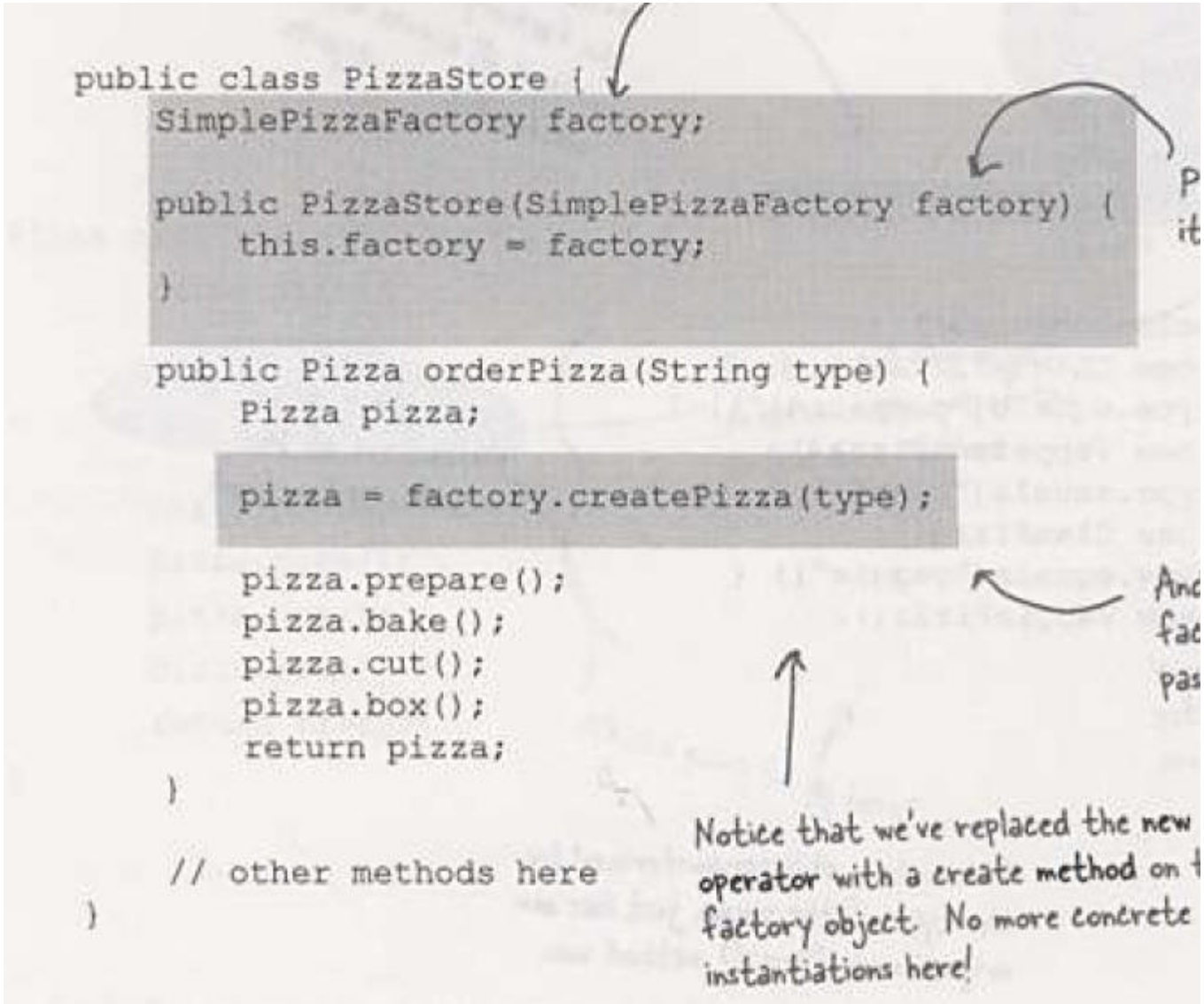
This is what varies. As the pizza selection changes over time, you'll have to modify this code over and over.

Create a factory

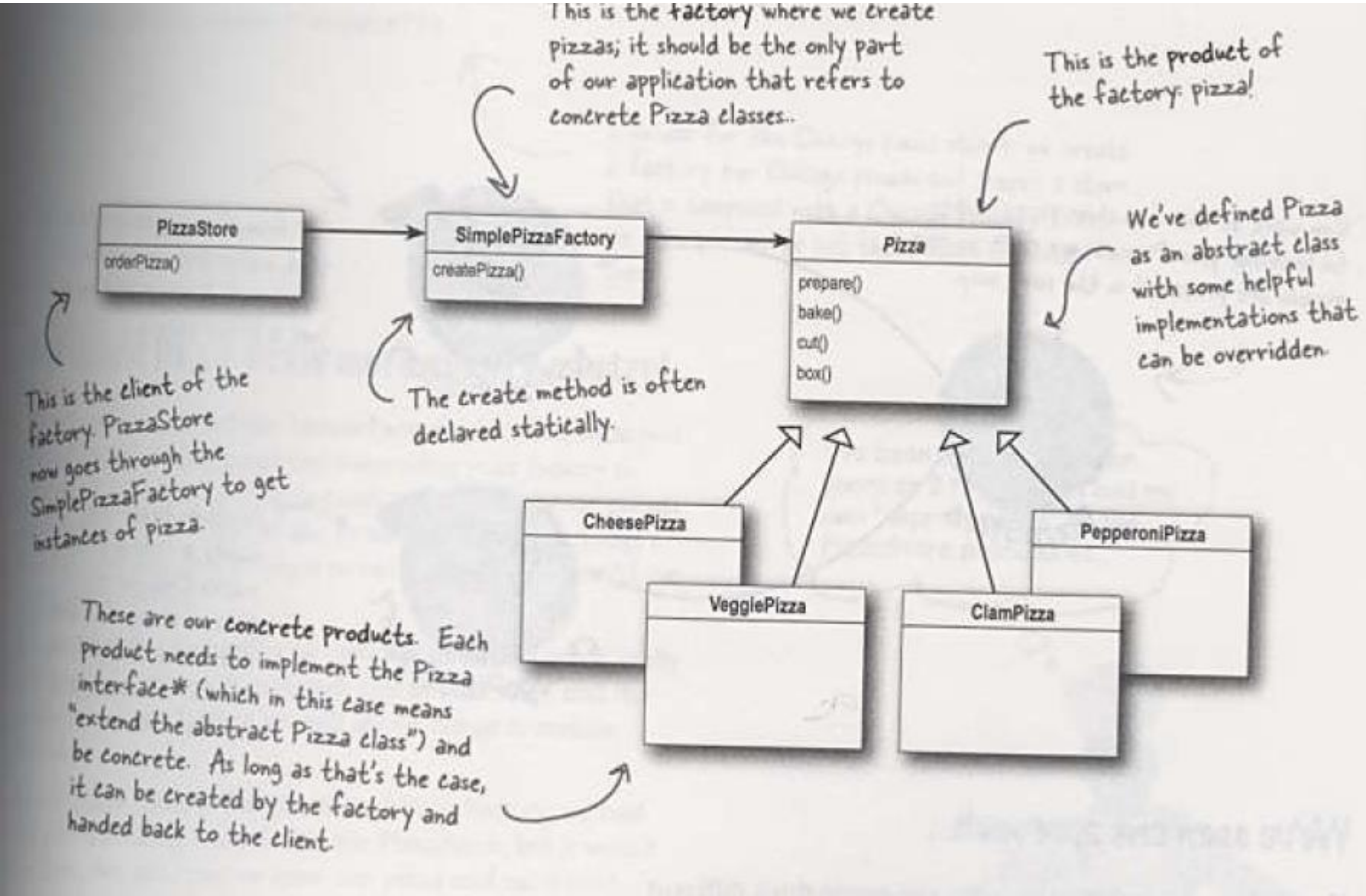
Simple Factory



Create factory object



Modified Factory class



Selling Pizza abroad

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

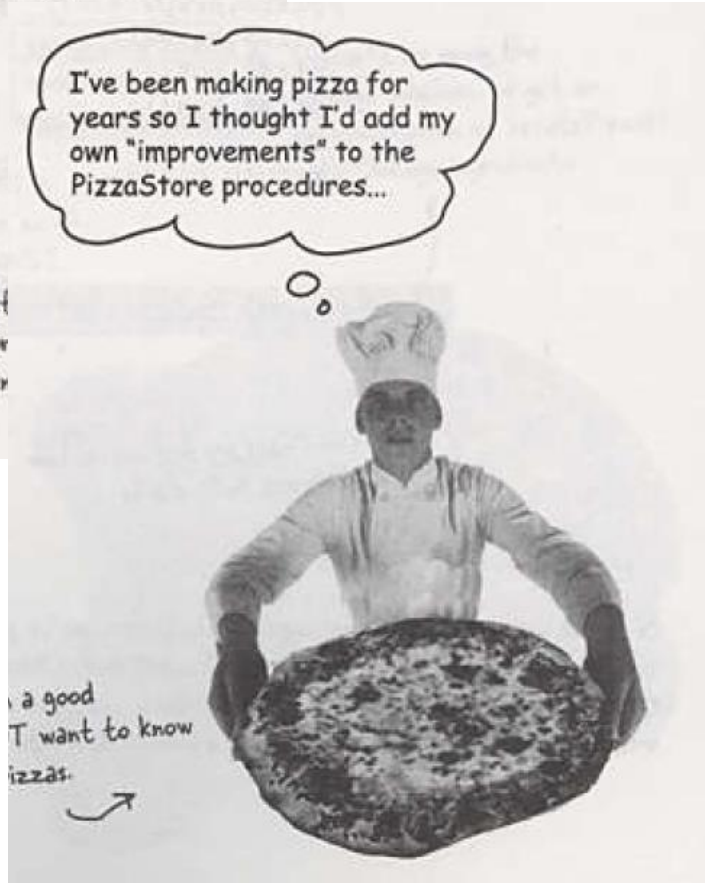
Here we create a factory for making NY style pizzas.

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

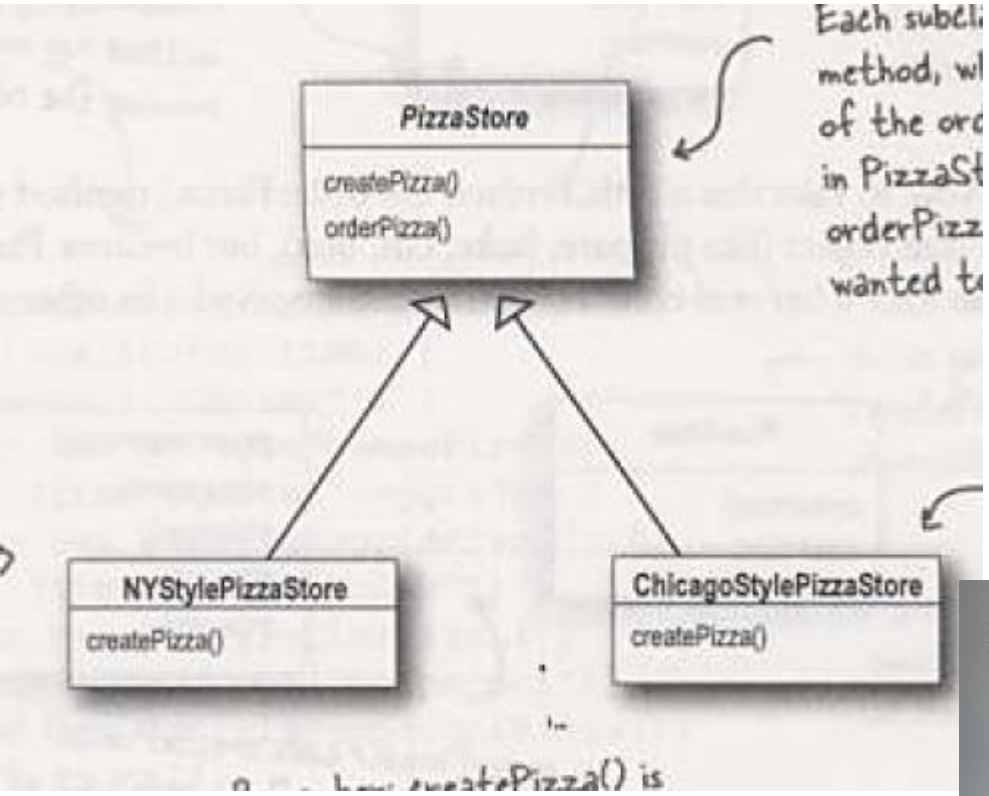
...and when we make pizzas, we get NY-styled pizzas.

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

Likewise for the Chicago pizza stores: we creat a factory for Chicago pizzas and create a stor that is composed with a Chicago factory. Wher we make pizzas, we get the Chicago flavored ones

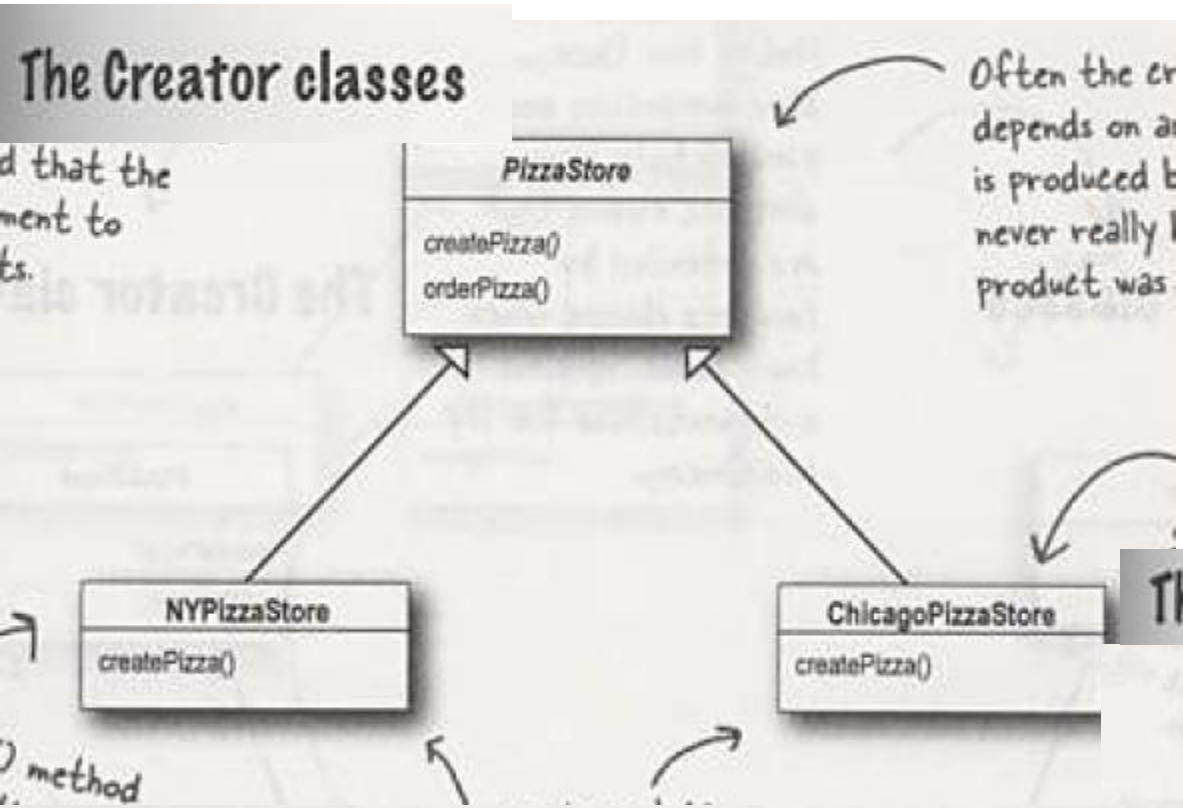


Framework for pizza store

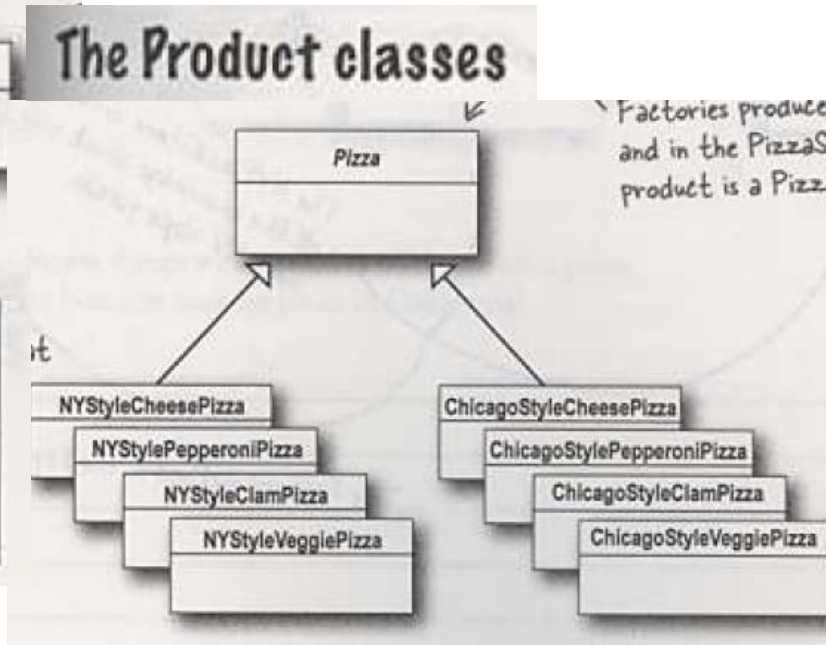



```
public class NYPizzaStore extends PizzaStore {
    Pizza createPizza(String item) {
        if (item.equals("cheese")) {
            return new NYStyleCheesePizza();
        } else if (item.equals("veggie")) {
            return new NYStyleVeggiePizza();
        } else if (item.equals("clam")) {
            return new NYStyleClamPizza();
        } else if (item.equals("pepperoni")) {
            return new NYStylePepperoniPizza();
        } else return null;
    }
}
```

Factory method pattern



The Factory Method Pattern defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.



Official definition