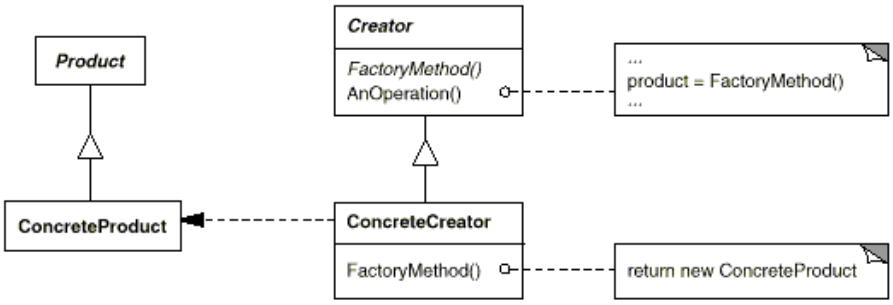
- PizzaStore Example
  - We have a pizza store program that wants to separate the process of creating a pizza from the process of preparing/ordering a pizza
  - There are different franchises for pizza that exist in different parts of the country
  - Each franchise needs its own factory to match the preferences of the locals, i.e.: a New York cheese pizza would be different from a Los Angeles cheese pizza.
  - However, we want to retain the preparation process for the original PizzaStore (since they are uniformed across all franchises).

1. Look at the initial code on the next page. How would you address the needs for "franchising"? (i.e., different cities' pizza stores that produce different pizzas.)
2. Can we make the Pizza class abstract? Why or why not? What about the PizzaStore class?
3. This is the class diagram representation of the Factory Method Design Pattern. What do you think this diagram means? Try not looking it up and come up with your own theory first. (Note: the arrow from "ConcreteCreator" to "ConcreteProduct" represents "creation")



4. Now, we want to follow this pattern to solve our problem.
   a) First order is to separate the creation code with the preparation ones. How?
   b) Can you map all the classes in our example, to the class diagram above?
   c) If we are missing some classes… what should we add?

5. What are the benefits and drawbacks of this method compares to what you originally thought of?

6. Questions to discuss:
   a) Can you think of other examples in your experience that could be solved with this pattern?
   b) What if, the question becomes slightly more complicated, and we want to add another level of differences?
      i. What if we consider pizzas to be made of toppings and doughs, and NY and LA and Pullman each has different toppings and doughs? i.e., the cheese/pineapples/pepperoni in each city is also different?

```
public class PizzaStore {

       Pizza orderPizza(String type) {
            Pizza pizza;
            //all these pizzas are subclasses of the pizza class
            if (type.equals("cheese")) {
              pizza = new CheesePizza();
            } else if (type.equals("greek")) {
              pizza = new GreekPizza();
            } else if (type.equals("pepperoni")) {
              pizza = new PepperoniPizza();
            }

            //Preparation steps
            pizza.prepare();
            pizza.bake();
            pizza.cut();
            pizza.box();
            return pizza;
      }
}
class Pizza {
   String style;
}
class CheesePizza extends Pizza {
      public CheesePizza() {
          //what would the code looks like here?...
          this.style = “cheese”;
      }
}
```