Factory Design Patters

Presented By

Dr. Sunirmal Khatua

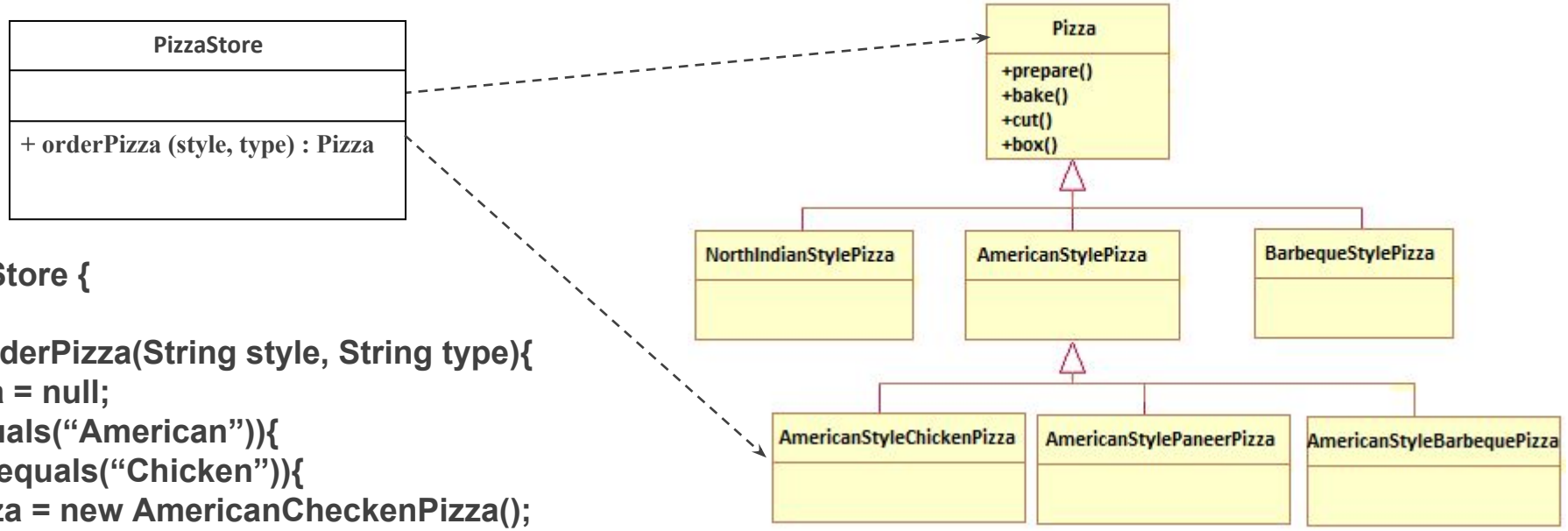
Visvesvaraya Young Faculty Fellow, Govt. of India

Asst. Professor, University of Calcutta

Classification of Design Pattern

Creational	Structural	Behavioural
Singleton	Adapter	Interpreter
Object Pool	Bridge	Template Method
Reflections	Composite	Chain of Responsibility
Factory	Decorator	Command
Factory Method	Flyweight	Iterator
Abstract Factory	Facade	Mediator
Builder	Proxy	Memento
Prototype		Observer
		State
		Strategy
		Visitor

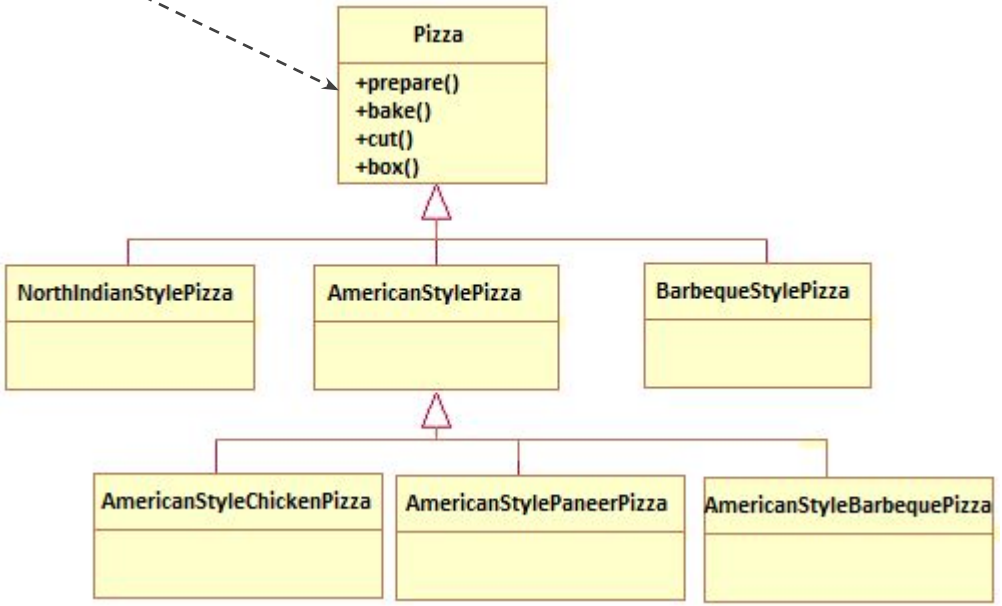
Have a look at PizzaStore Problem



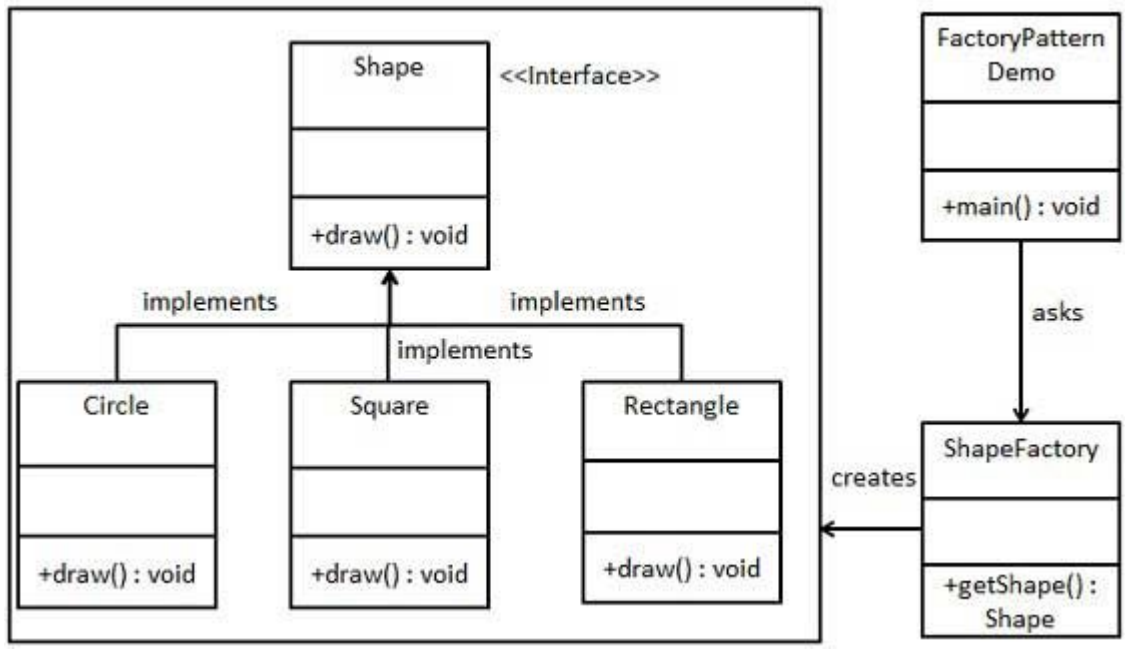

```
public class PizzaStore {

    public Pizza orderPizza(String style, String type){
        Pizza pizza = null;
        if(style.equals("American")){
            if(type.equals("Chicken")){
                pizza = new AmericanCheckenPizza();
            }
            ……
        }else if(style.equals("NorthIndian")){
            ……
        }
        ……
        ……
    }
    pizza.prepare(); pizza.bake(); pizza.cut(); pizza.box();
    return pizza;
}
```

Have a look at PizzaStore Problem


```
public class PizzaStore {

        private PizzaFactory factory = new PizzaFactory();

        public Pizza orderPizza(String style, String type){
                return factory.createPizza(style, type);
        }
}

public class PizzaFactory {
      public Pizza createPizza(String style, String type){
            Pizza pizza = null;
            if(style.equals("American")){
                  if(type.equals("Chicken")){
                        pizza = new AmericanCheckenPizza();
                  }
                  ……
            }
            pizza.prepare(); pizza.bake(); pizza.cut(); pizza.box();
            return pizza;
      }
}
```



Have a look at Shapes Problem




```java
public interface Shape
{
      void draw();
}
```

```java
public class Rectangle implements Shape {
   @Override public void draw() {
        ……
   }
}
```

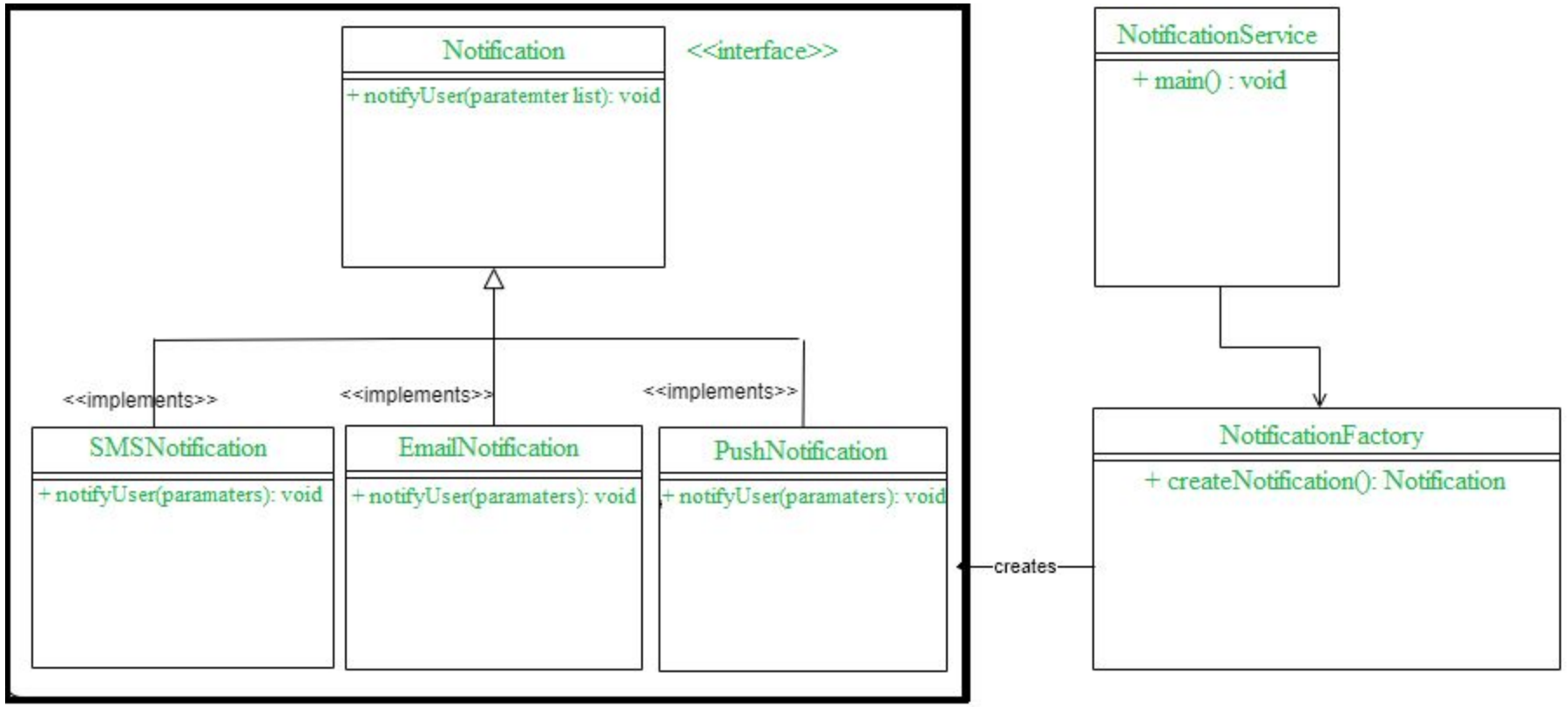
```java
public class Circle implements Shape {
   @Override public void draw() {
        ……
   }
}
```

```java
public class Square implements Shape {
   @Override public void draw() {
        ……
   }
}
```

```java
public class ShapeFactory {
    public Shape getShape(String shapeType){
       Shape s = null;
       if(shapeType.equalsIgnoreCase("CIRCLE")){
          s = new Circle();
       } else
if(shapeType.equalsIgnoreCase("RECTANGLE")){
          s =  new Rectangle();
       } else if(shapeType.equalsIgnoreCase("SQUARE")){
          s = new Square();
       }
       return s;
    }
}
```

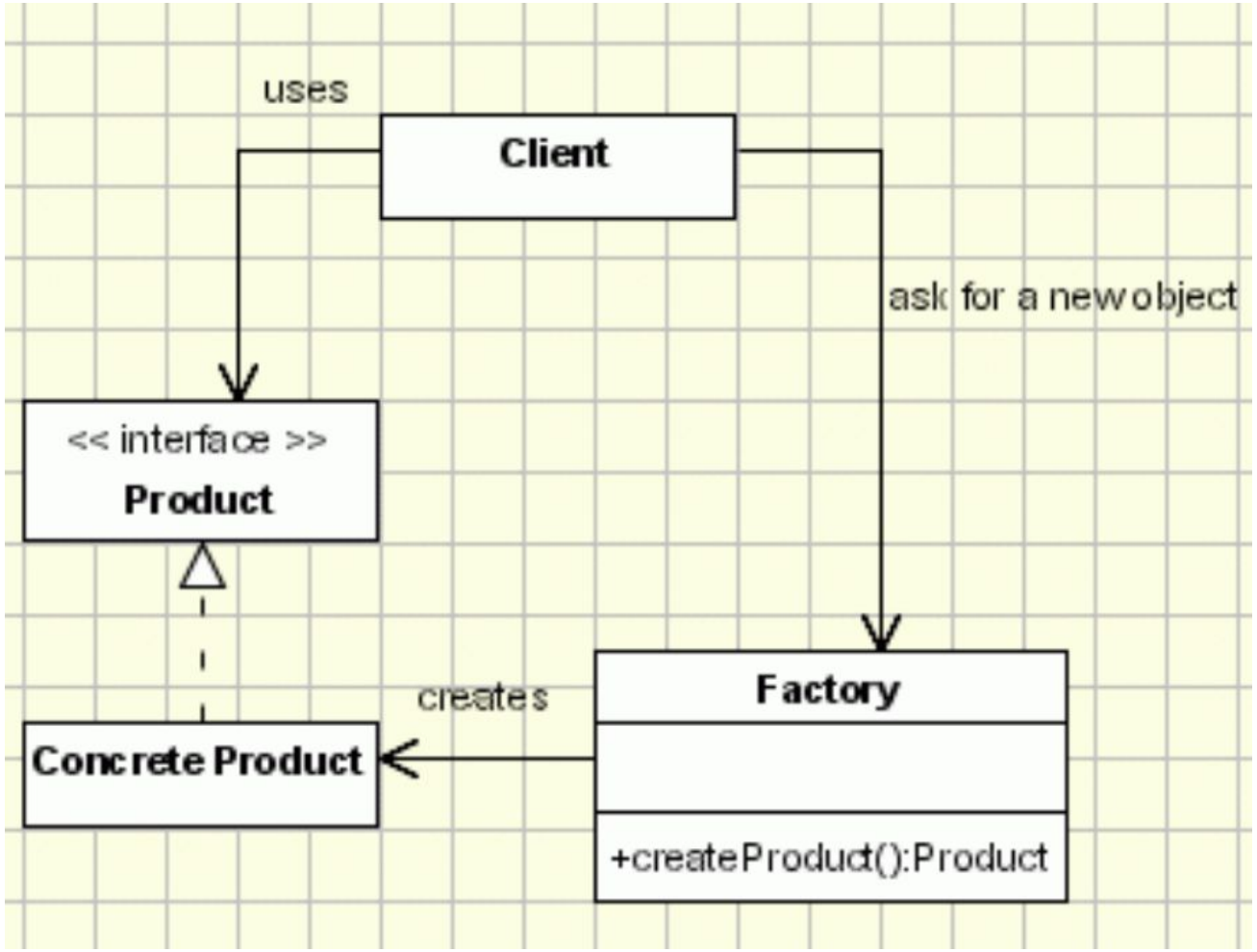
```java
public class FactoryPatternDemo {
  public static void main(String[] args) {
    ShapeFactory shapeFactory =
                    new ShapeFactory();
    Shape shape =
            shapeFactory.getShape(args[1]);
    shape.draw();
  }
}
```

Have a look at Notification Problem

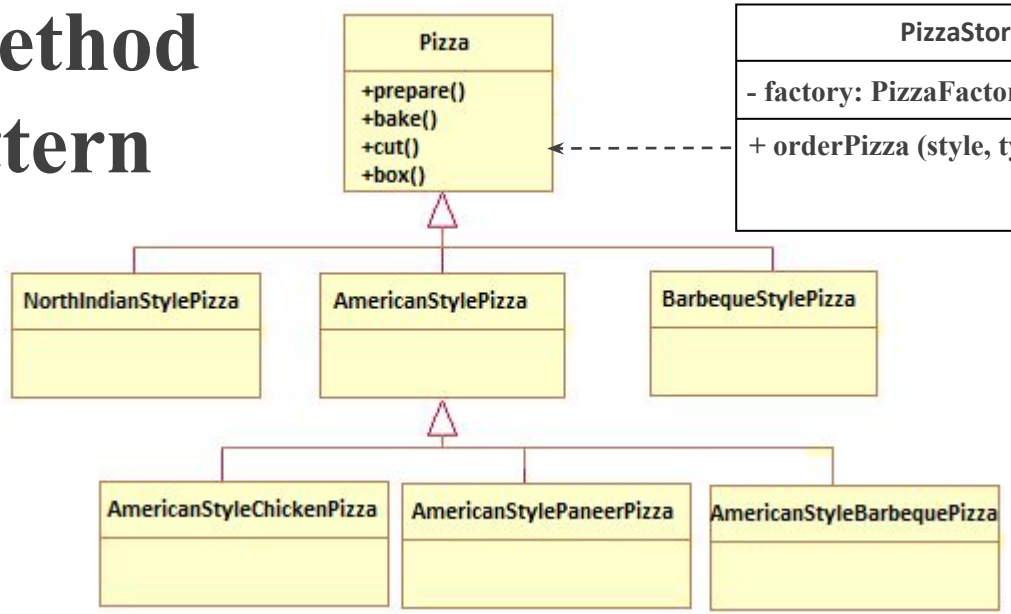


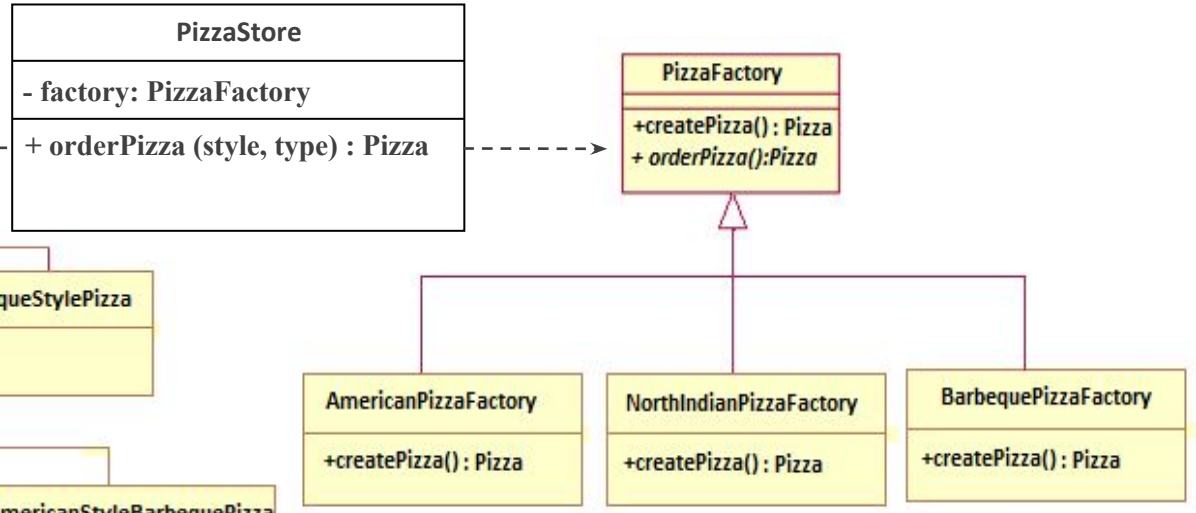
Factory Design Pattern

Allows the creation of objects without specifying their concrete type.



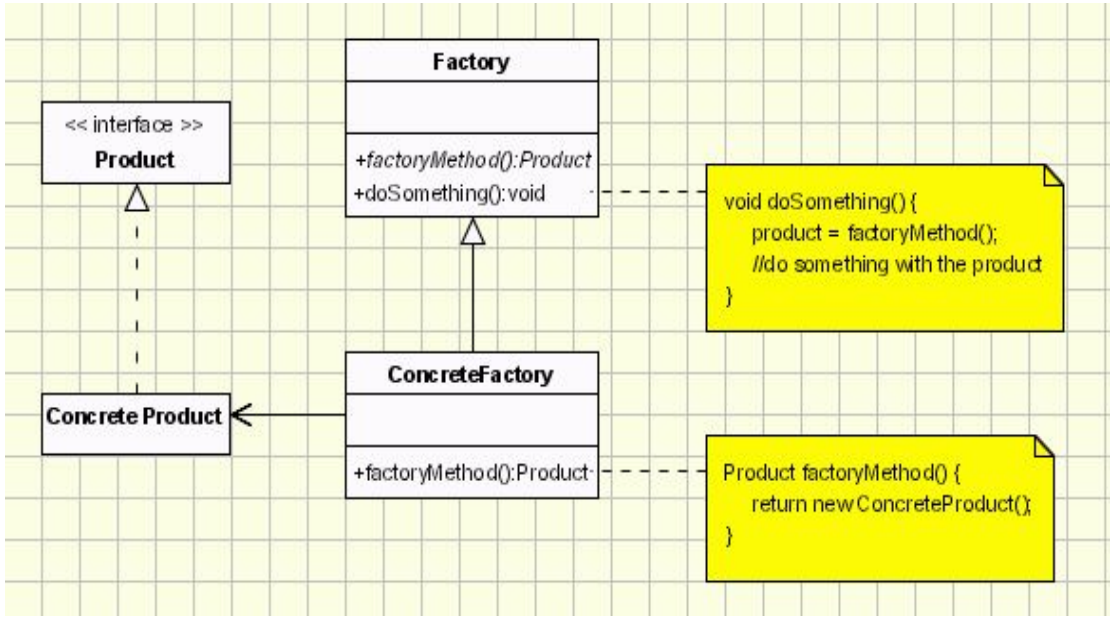
Factory Method Design Pattern



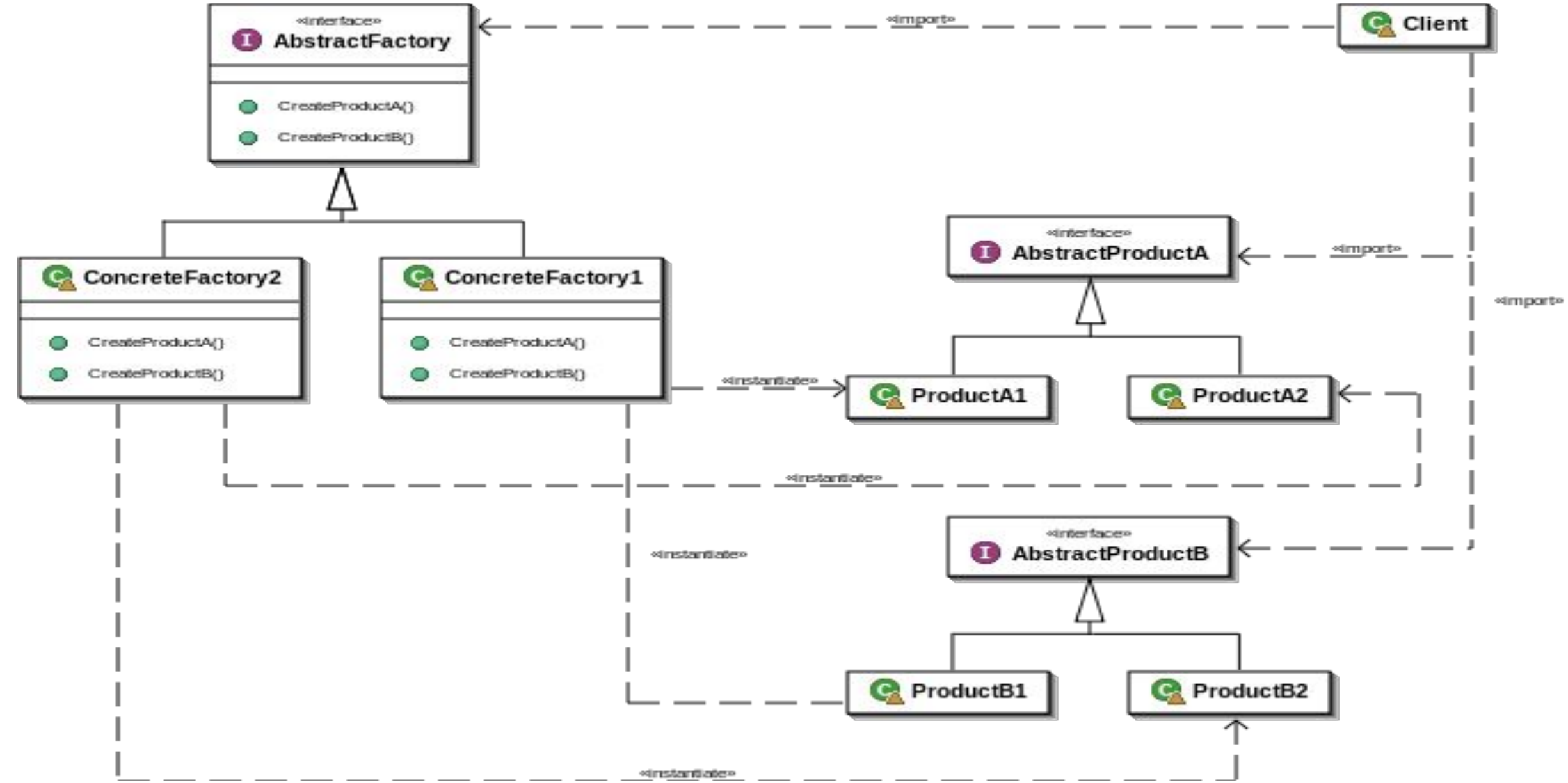



```
public abstract class PizzaFactory{

    protected abstract Pizza createPizza(String style);

    public static Pizza orderPizza(String style, String type){
        Pizza pizza = getFactory(style).createPizza(type);
        return pizza;
    }

    private static PizzaFactory getFactory(String style){
        PizzaFactory factory = null;
        if(style.equals("American") factory = new AmericalPizzaFactory();
        else if(style.equals("NorthIndian") factory = new NorthIndianPizzaFactory();
        else if(style.equals("Barbeque") factory = new BarbequePizzaFactory();
        return factory;
    }
}
```

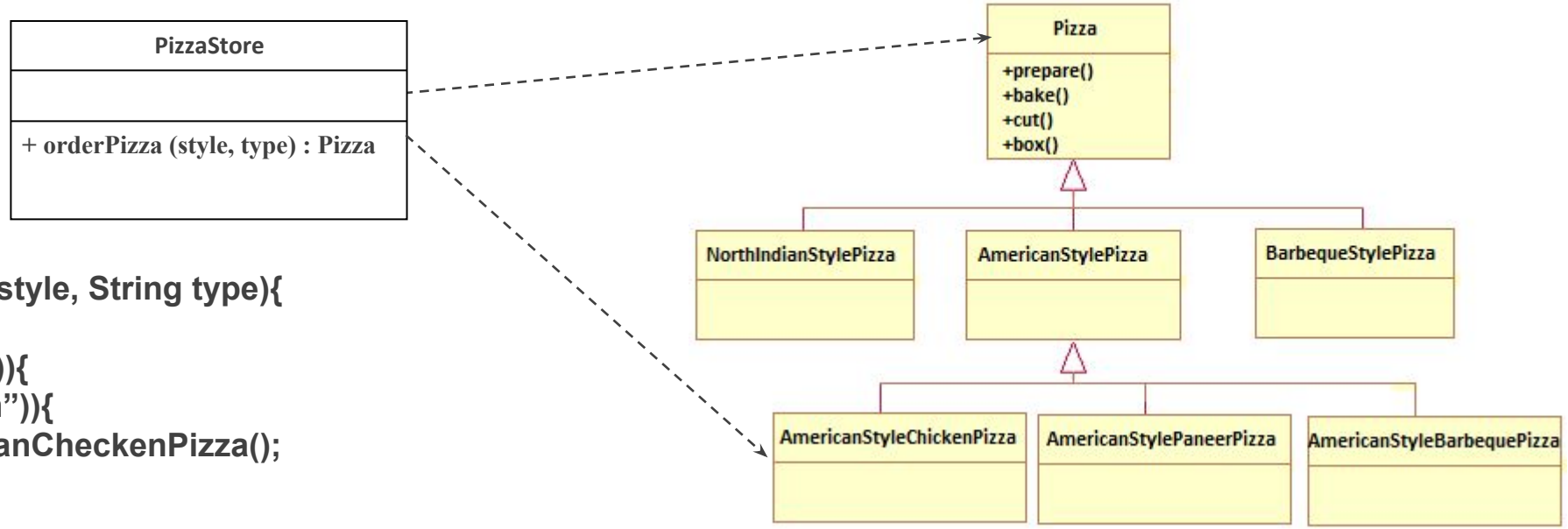
```
public class PizzaStore {
    public Pizza orderPizza(String style, String type){
        return PizzaFactory.orderPizza(style, type);
    }
}
```



Abstract Factory Design Pattern

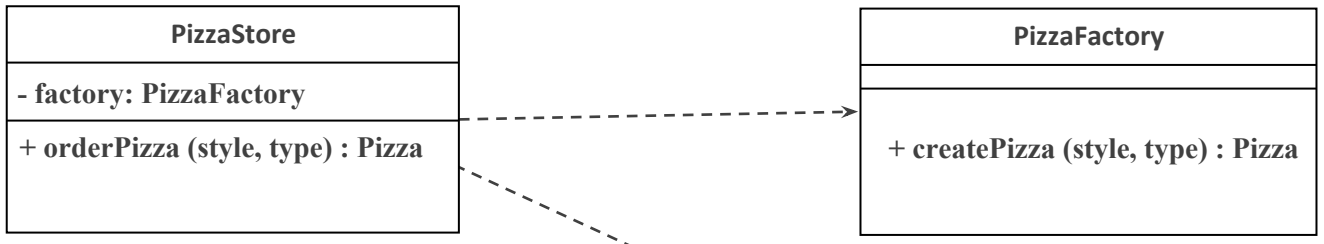


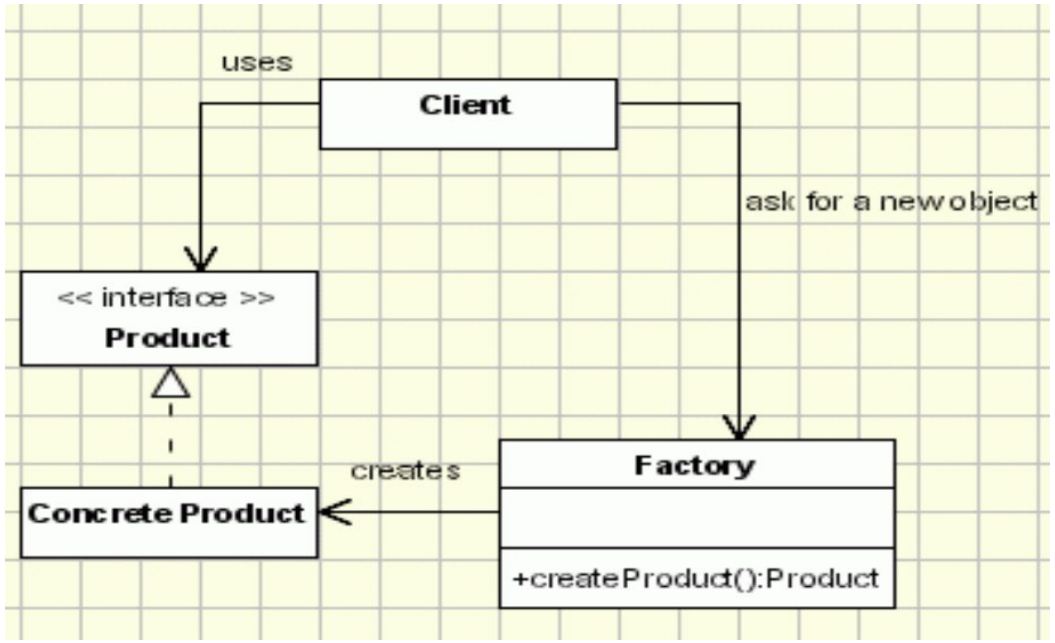
Have a look at PizzaStore Problem




```
public class PizzaStore {

    public Pizza orderPizza(String style, String type){
        Pizza pizza = null;
        if(style.equals("American")){
            if(type.equals("Chicken")){
                pizza = new AmericanCheckenPizza();
            }
            ……
        }else if(style.equals("NorthIndian")){
            ……
        }
        ……
        ……
    }
    pizza.prepare(); pizza.bake(); pizza.cut(); pizza.box();
    return pizza;
}
```

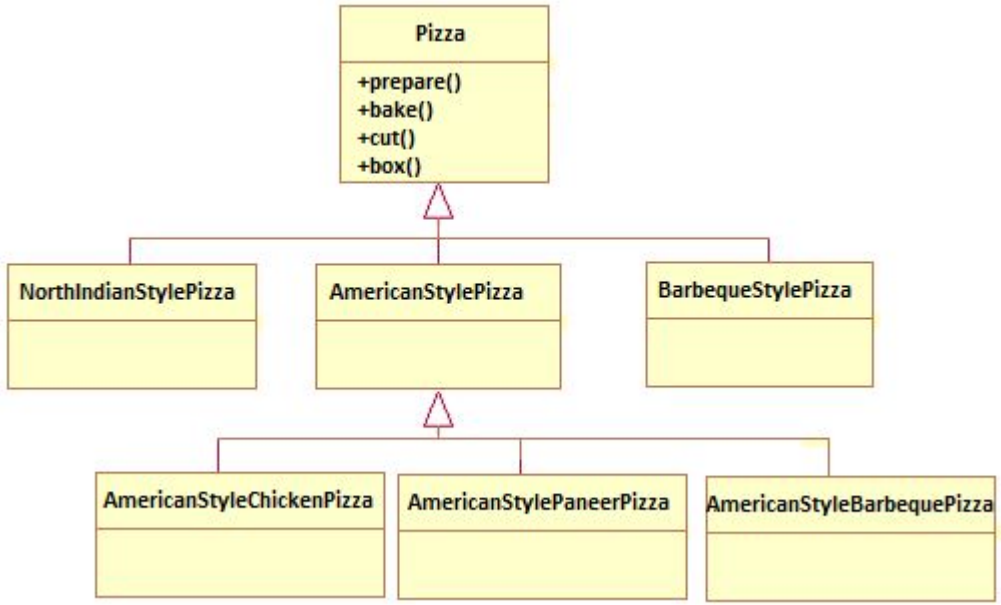
Factory Design Pattern



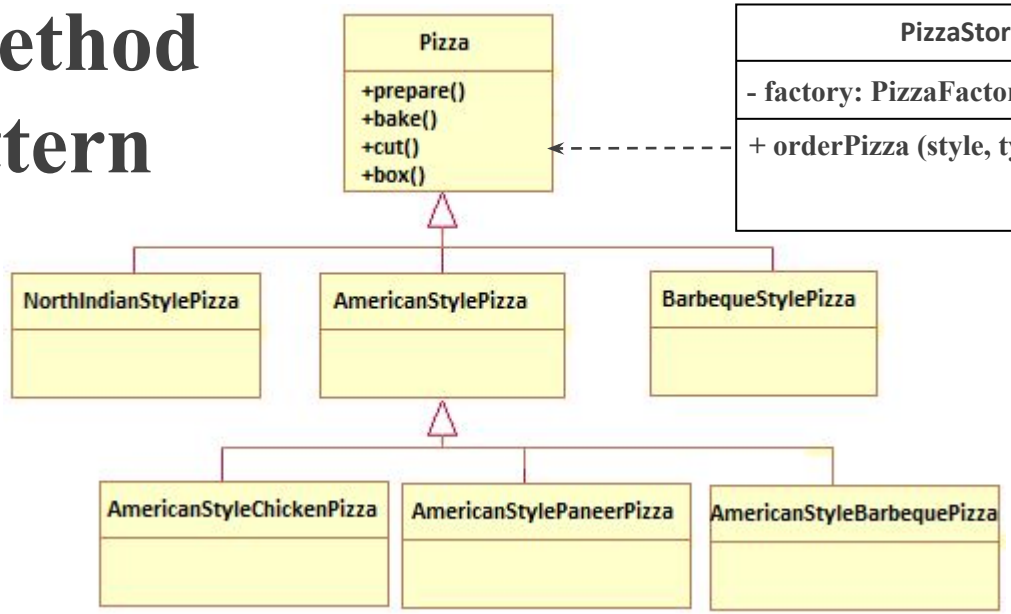


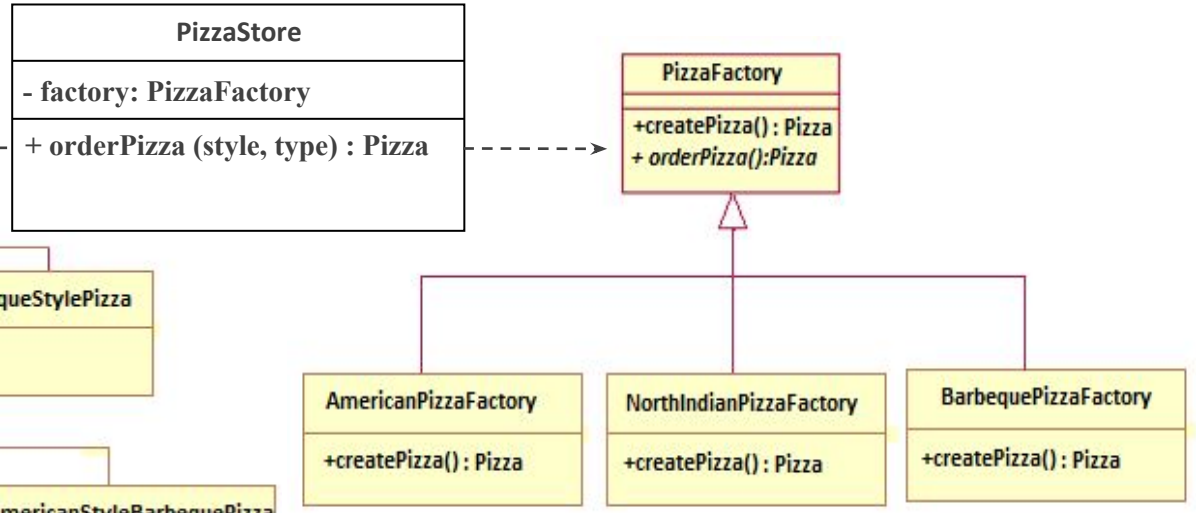

```
public class PizzaStore {

      private PizzaFactory factory = new PizzaFactory();

      public Pizza orderPizza(String style, String type){
            Pizza pizza = factory.createPizza(style, type);
            pizza.prepare(); pizza.bake(); pizza.cut(); pizza.box();
            return pizza;
      }
}
```

```
public class PizzaFactory {
     public Pizza createPizza(String style, String type){
          Pizza pizza = null;
          if(style.equals("American")){
               if(type.equals("Chicken")){
                    pizza = new AmericanCheckenPizza();
               }
               ……
          }
          return pizza;
     }
}
```



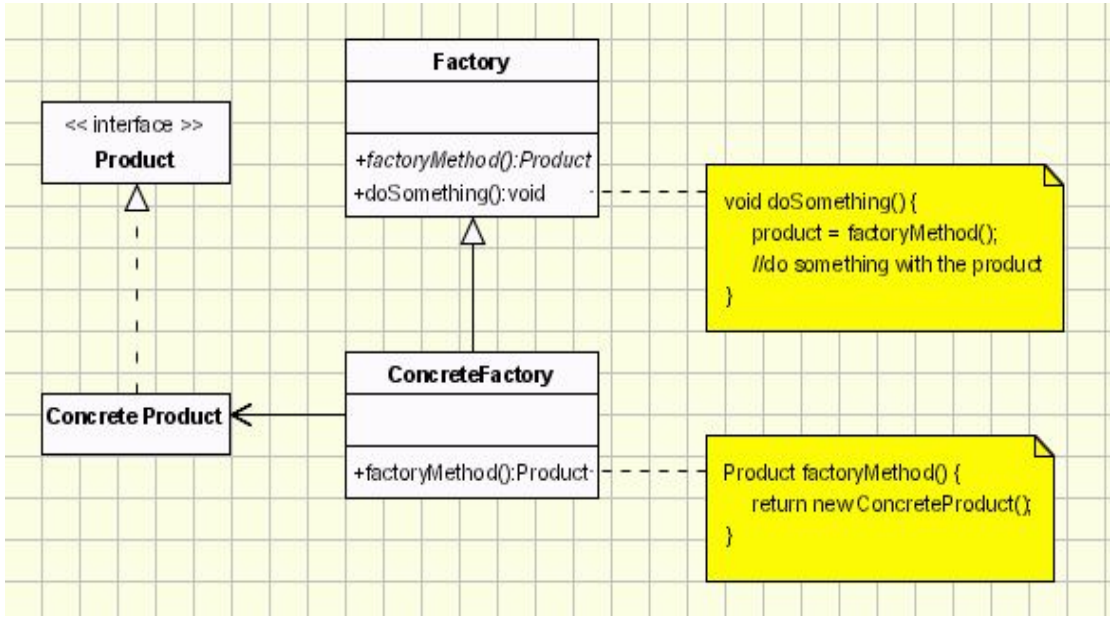
Factory Method Design Pattern



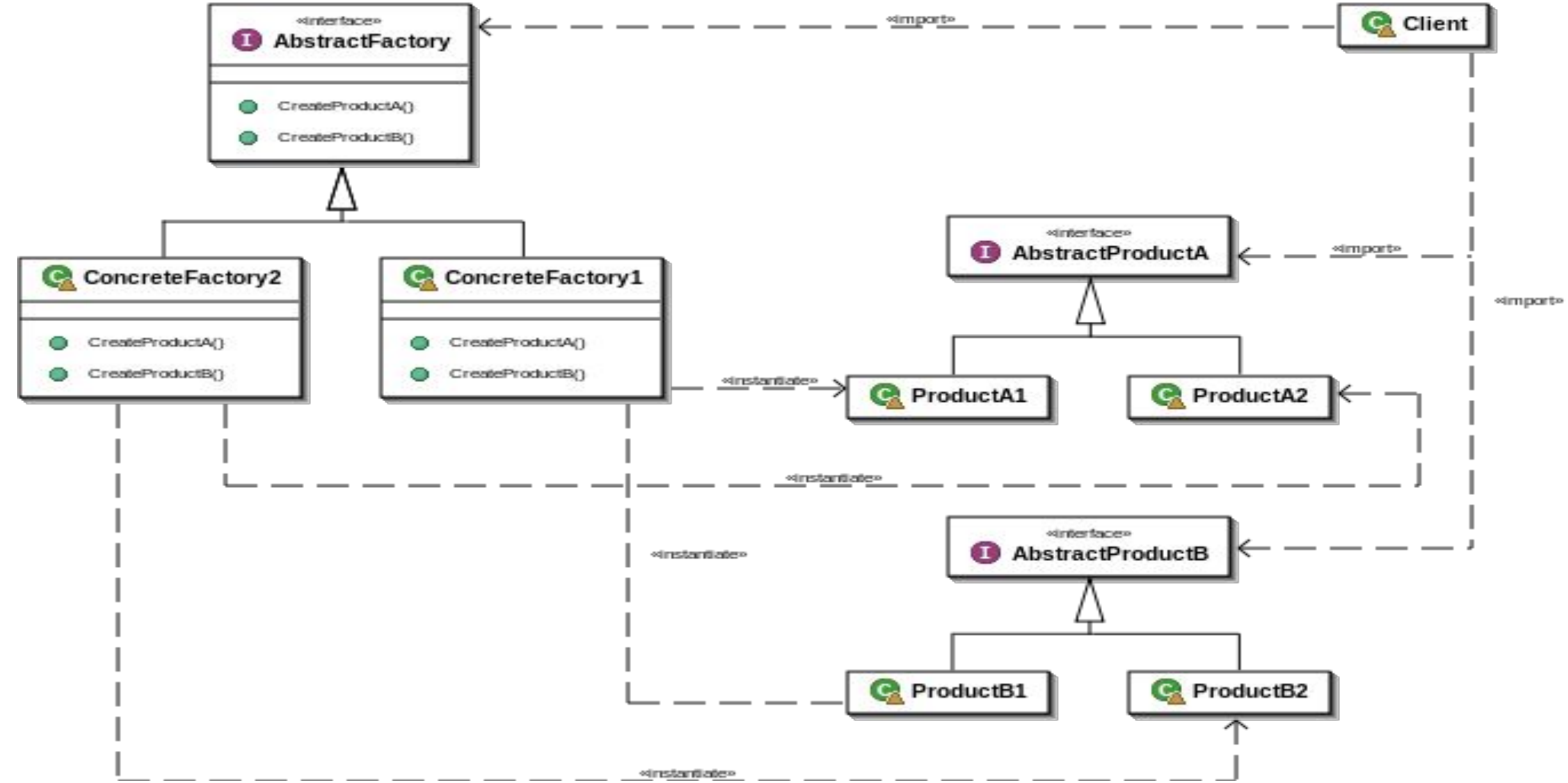



```
public abstract class PizzaFactory{

    protected abstract Pizza createPizza(String style);

    public static Pizza orderPizza(String type){
        Pizza pizza = createPizza(type);
        pizza.prepare(); pizza.bake(); pizza.cut(); pizza.box();
        return pizza;
    }

    public static PizzaFactory getFactory(String style){
        PizzaFactory factory = null;
        if(style.equals("American") factory = new AmericalPizzaFactory();
        else if(style.equals("NorthIndian") factory = new NorthIndianPizzaFactory();
        else if(style.equals("Barbeque") factory = new BarbequePizzaFactory();
        return factory;
    }
}
```

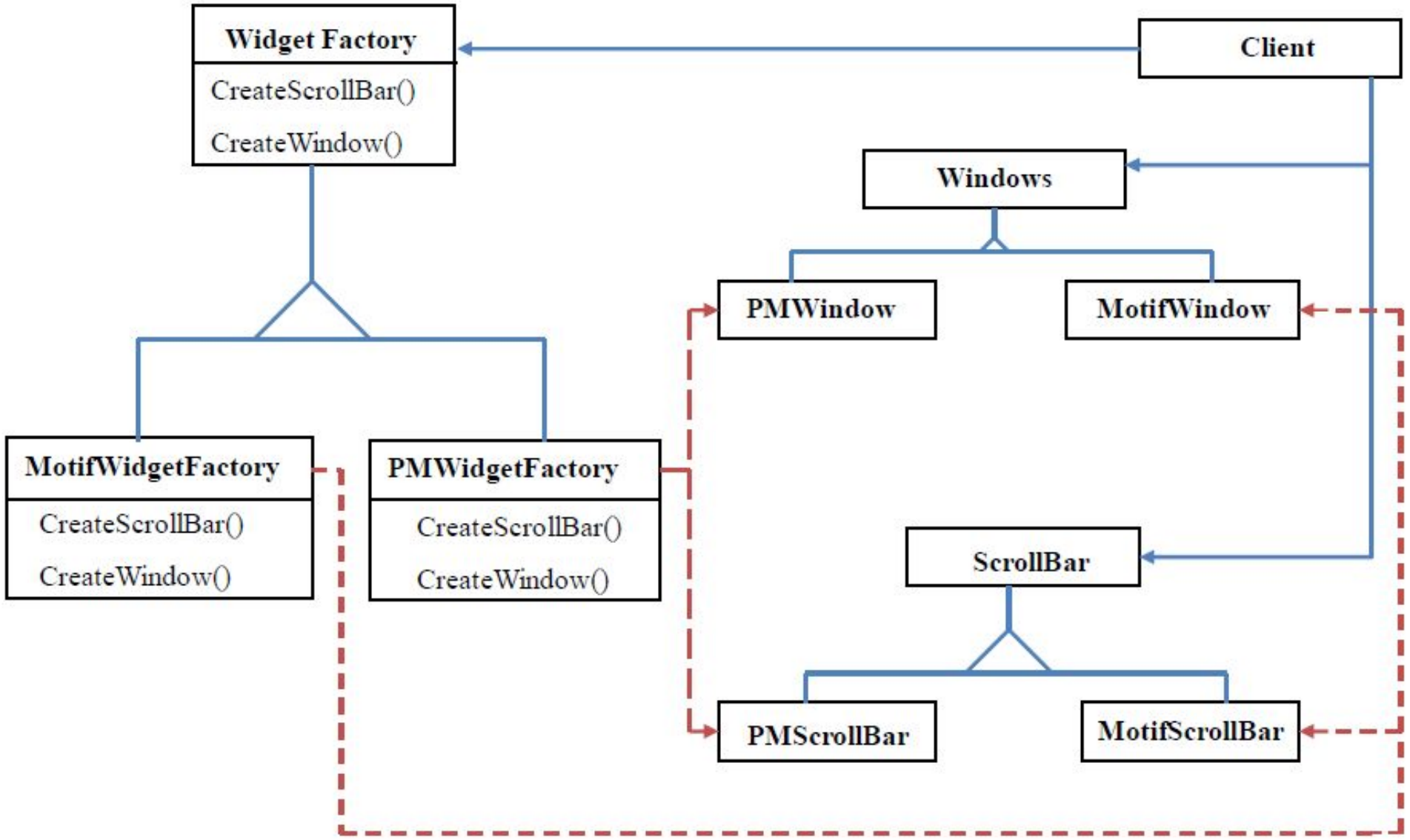
```
public class PizzaStore {
    public Pizza orderPizza(String style, String type){
        return PizzaFactory.getFactory(style).orderPizza(type);
    }
}
```



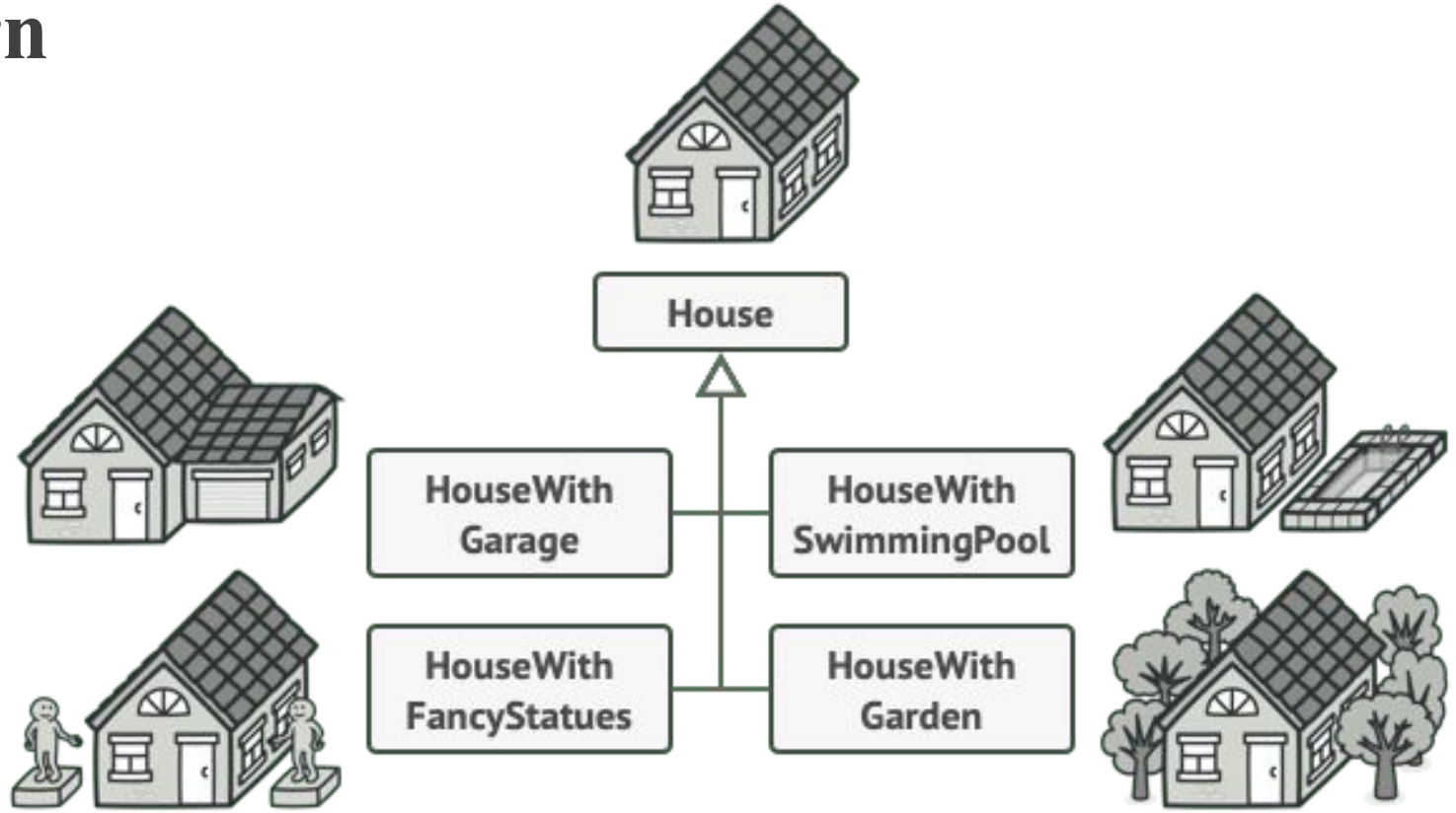
Abstract Factory Design Pattern



Abstract Factory Design Pattern



Builder Design Pattern




```
public User (String firstName, String lastName, int age, String
phone){ ... }

public User (String firstName, String lastName, String phone, String
address){ ... }

public User (String firstName, String lastName, int age){ ... }

public User (String firstName, String lastName){ ... }
```

Thank You