装饰者模式

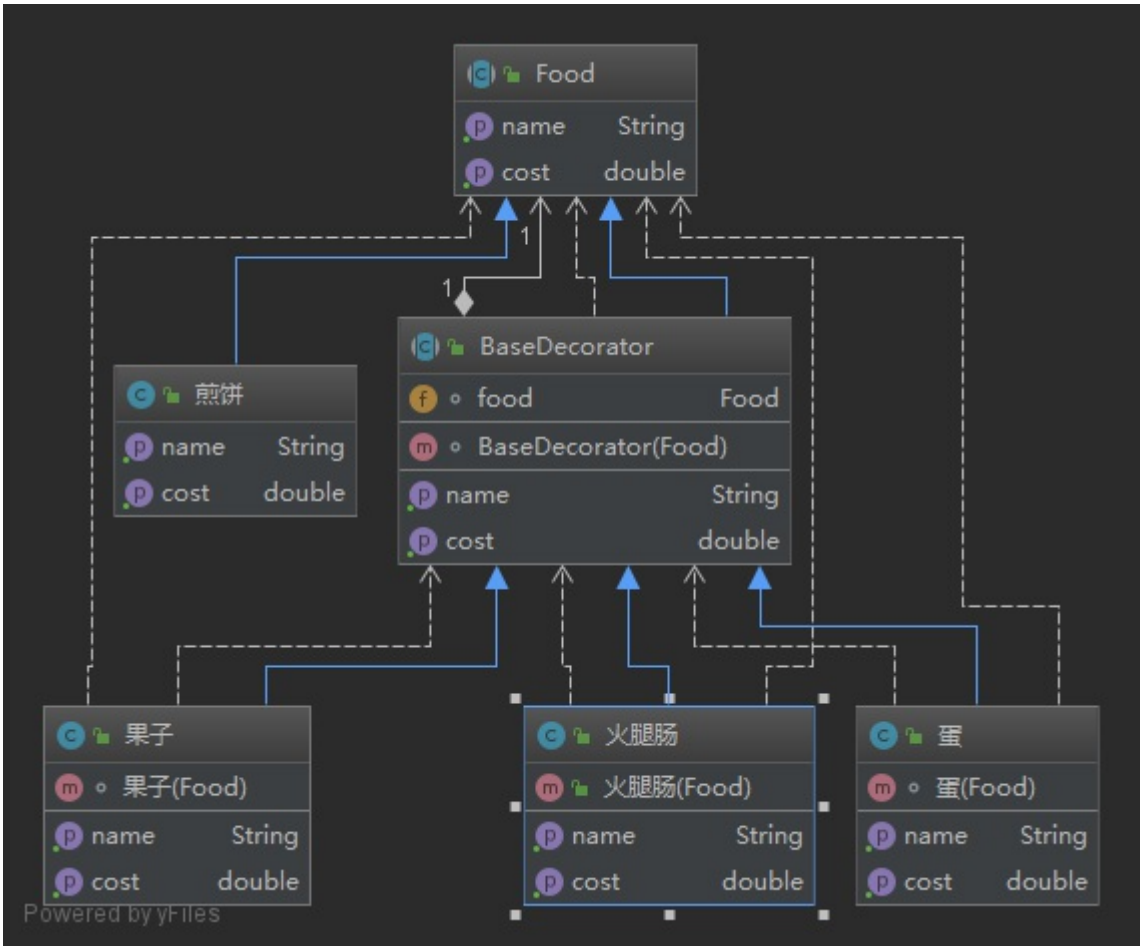
模式定义

装饰者模式动态的将责任附加在对象上, 若要扩展功能, 装饰者提供了比继承更有 弹性的替代方案

问题需求描述

模拟北门卖煎饼果子的

设计类图



Code

Food.java

```
package Decorator;

public abstract class Food {
    public abstract double getCost();

    public abstract String getName();
}
```

煎饼.java

```
package Decorator;
```

```
public class 煎饼 extends Food {
    @Override
    public double getCost() {
        return 1.0;
    }

    @Override
    public String getName() {
        return "煎饼";
    }
}
```

BaseDrcorator.java

```
package Decorator;

public abstract class BaseDecorator extends Food {
    Food food;

    BaseDecorator(Food food) {
        this.food = food;
    }

    @Override
    public abstract double getCost();

    @Override
    public abstract String getName();
}
```

果子.java

```
package Decorator;

public class 果子 extends BaseDecorator {
    果子(Food food) {
        super(food);
    }

    @Override
    public double getCost() {
        return food.getCost() + 1.0;
    }

    @Override
    public String getName() {
        return food.getName() + "+果子";
    }
}
```

火腿肠.java

```
package Decorator;

public class 火腿肠 extends BaseDecorator {

    public 火腿肠(Food food) {
        super(food);
    }

    @Override
    public double getCost() {
        return food.getCost() + .5;
    }

    @Override
    public String getName() {
        return food.getName() + "+ 火腿肠";
    }
}
```

蛋.java

```
package Decorator;

public class 蛋 extends BaseDecorator {
    蛋(Food food) {
        super(food);
    }

    @Override
    public double getCost() {
        return food.getCost() + .3;
    }

    @Override
    public String getName() {
        return food.getName() + "+蛋";
    }
}
```

运行结果

Main.java

```
package Decorator;

public class Main {
    public static void main(String[] args) {
        Food food = new 果子(new 蛋(new 煎饼()));
        System.out.println(food.getCost());
        System.out.println(food.getName());
    }
}
```

out

```
2.3
煎饼+蛋+果子
```