设计模式之策略模式

设计模式之策略模式

本课程概要

课程概要

- 设计模式入门
- 策略模式原理
- 策略模式示例演示
- 策略模式的注意点

设计模式之策略模式

设计模式入门

设计模式入门

1、设计模式是人们在面对同类型软件工程设计问题所总结出的一些有用经验。模式不是代码，而是某类问题的通用设计解决方案
2、4人组Erich Gamma、Richard Helm、Ralph Johnson、John Vlissides总结写了《设计模式》
3、设计模式的优点和用途
4、学习设计模式最好的方式：在你的设计和以往的工程里寻找何处可以使用它们
5、设计模式的本质目的是使软件工程在维护性、扩展性、变化性、复杂度方面成O(N)
6、OO是原则，设计模式是具体方法、工具

设计模式入门

在java里IO流的类设计，为什么把BufferedReader设计成：

```
new BufferedReader(new FileReader("F:\test.java"));
```

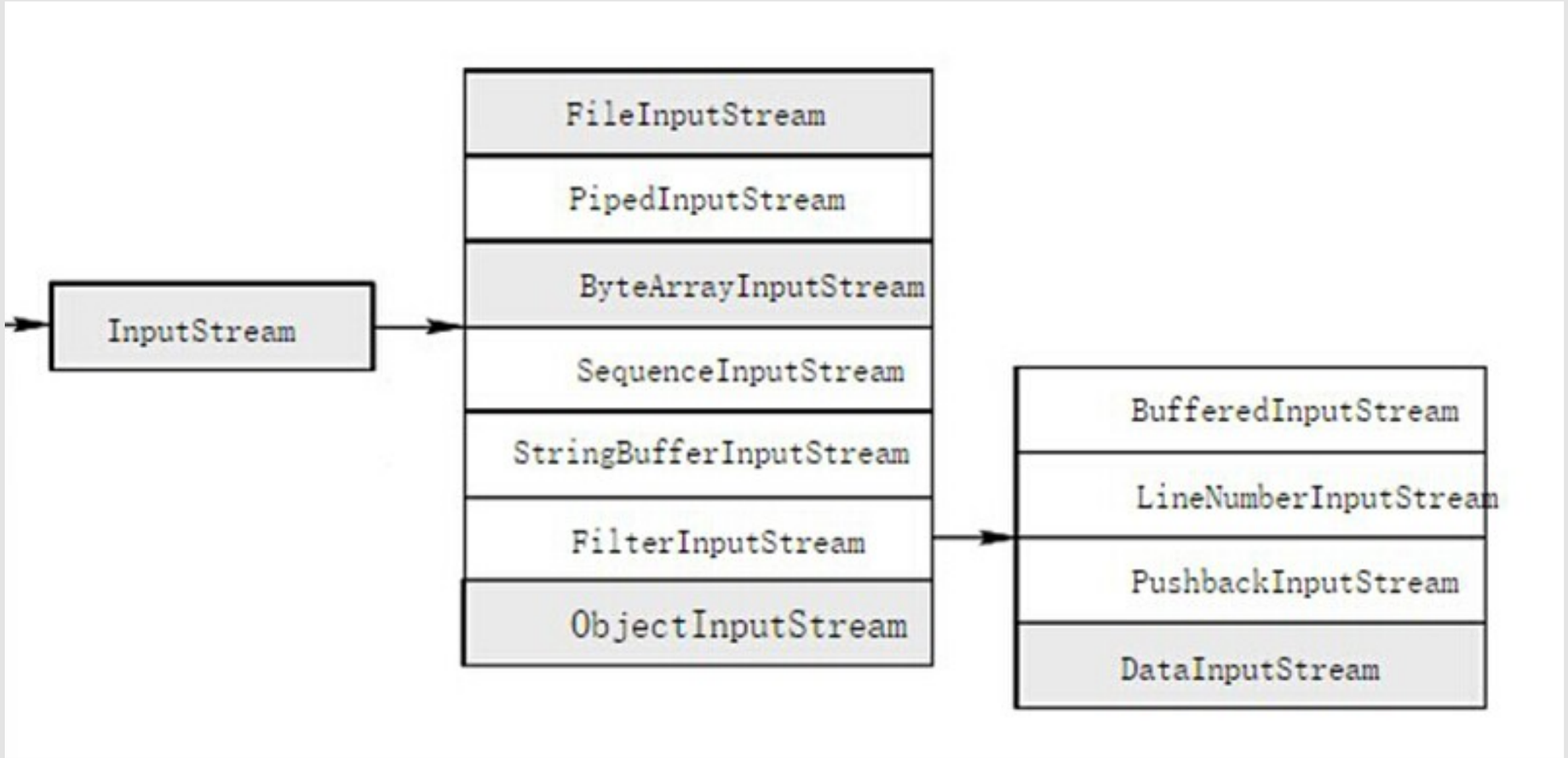
而不是设计成：

```
BufferedReader extends FileReader;
```

然后 `new BufferedReader("F:\test.java");`

...

设计模式入门


InputStream
FileInputStream
PipedInputStream
ByteArrayInputStream
SequenceInputStream
StringBufferInputStream
FilterInputStream
ObjectInputStream
BufferedInputStream
LineNumberInputStream
PushbackInputStream
DataInputStream

设计模式之策略模式

策略模式原理

策略模式原理

1、模拟鸭子项目
2、项目的新需求
3、用OO原则解决新需求的不足
4、用策略模式来新需求解决
5、重新设计模拟鸭子项目
6、总结策略模式定义

模拟鸭子项目

1、从项目"模拟鸭子游戏"开始
2、从OO的角度设计这个项目,鸭子超类，扩展超类：

```
public abstract class Duck {
	public void Quack() {
		System.out.println("~~gaga~~");
	}
	public abstract void display();
	public void swim() {
		System.out.println("~~im swim~~");
	}

}
```

模拟鸭子项目

1、GreenHeadDuck继承Duck ：

```
public class GreenHeadDuck extends Duck {
	@Override
	public void display() {
		System.out.println("**GreenHead**");
	}
}
```

2、同理可有RedHeadDuck等

项目的新需求

1、应对新的需求，看看这个设计的可扩展性

1）添加会飞的鸭子

2、OO思维里的继承方式解决方案是：

```
public abstract class Duck {
        ...;
        public void Fly() {
        System.out.println("~~im fly~~");
    }
};
```

问题来了,这个Fly让所有子类都会飞了，这是不科学的。

继承的问题：对类的局部改动，尤其超类的局部改动，会影响其他部分。影响会有溢出效应

用OO原则解决新需求的不足

1、继续尝试用OO原理来解决，覆盖：

```
public class GreenHeadDuck extends Duck {
...;
public void Fly() {
        System.out.println("~~no fly~~");
    }
}
```

2、又有新需求，石头鸭子，填坑：

```
public class StoneDuck extends Duck {
.... };
```

超类挖的一个坑，每个子类都要来填，增加工作量，复杂度O(N^2)。不是好的设计方式

用策略模式来新需求解决

需要新的设计方式，应对项目的扩展性，降低复杂度：

1）分析项目变化与不变部分，提取变化部分，抽象成接口+实现；

2）鸭子哪些功能是会根据新需求变化的？叫声、飞行...

用策略模式来新需求解决

1、接口：

1）public interface FlyBehavior
{
void fly();}

2）public interface QuackBehavior
{
void quack();};

3）好处：新增行为简单，行为类更好的复用，组合更方便。既有继承带来的复用好处，没有挖坑

重新设计模拟鸭子项目

1、重新设计的鸭子项目：

```
public abstract class Duck {
        FlyBehavior mFlyBehavior;
        QuackBehavior mQuackBehavior;
        public Duck() {  }
        public void Fly() {
                mFlyBehavior.fly();
        }
        public void Quack() {
                mQuackBehavior.quack();
        }
        public abstract void display();
        }
```

总结策略模式定义

1、绿头鸭、石头鸭：

```
public class GreenHeadDuck extends Duck {
    public GreenHeadDuck() {
        mFlyBehavior = new GoodFlyBehavior();
        mQuackBehavior = new GaGaQuackBehavior();
    }
    @Override
    public void display() {...}
}
```

2、策略模式：分别封装行为接口，实现算法族，超类里放行为接口对象，在子类里具体设定行为对象。原则就是：分离变化部分，封装接口，基于接口编程各种功能。此模式让行为算法的变化独立于算法的使用者。

设计模式之策略模式

策略模式示例演示

设计模式之策略模式

策略模式注意点

策略模式注意点

1、分析项目中变化部分与不变部分
2、多用组合少用继承；用行为类组合，而不是行为的继承。更有弹性
3、设计模式有没有相应的库直接使用？有些库或框架本身就用某种设计模式设计的
4、如果找不到适用的模式怎么办

极客学院
jikexueyuan.com