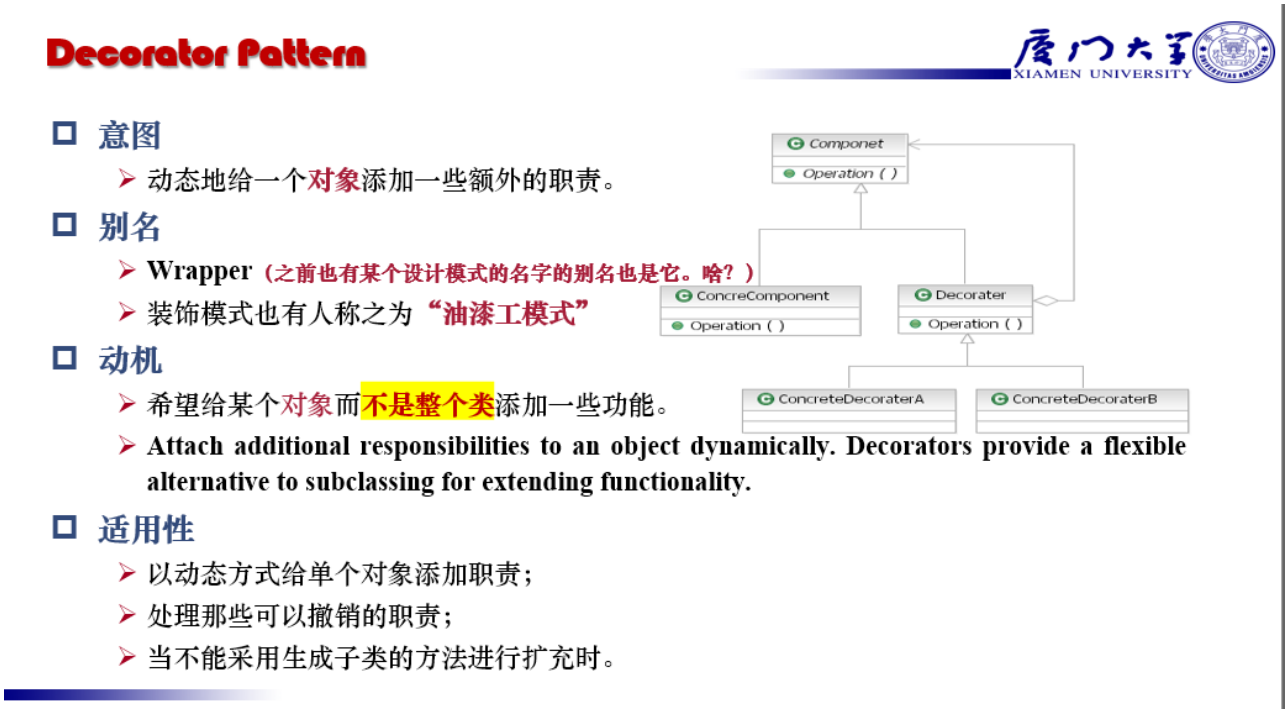
软件体系结构 作业15

22920212204392 黄勖

1　什么是透明装饰模式，什么是半透明装饰模式？请举例说明。

1. 实现思路与代码说明

装饰器模式可以动态地给一个对象添加一些额外的职责，即在不影响其他对象地情况下，以动态方式给单个对象添加职责。其中Component定义一个对象接口，可以给这些对象动态地添加职责。ConcreteComponent定义一个对象，可以给这个对象添加一些职责



（1）透明装饰模式（类似代理模式）

在透明装饰模式中要求客户端完全针对抽象编程，装饰模式的透明性要求客户端程序不应该将对象声明为具体构件类型或具体装饰类型，而应该**全部声明为抽象构件类型**。对客户端而言，具体构件类和具体装饰类对象没有任何区别。（缺点是无法单独调用装饰类的独有功能）

（2）半透明装饰模式

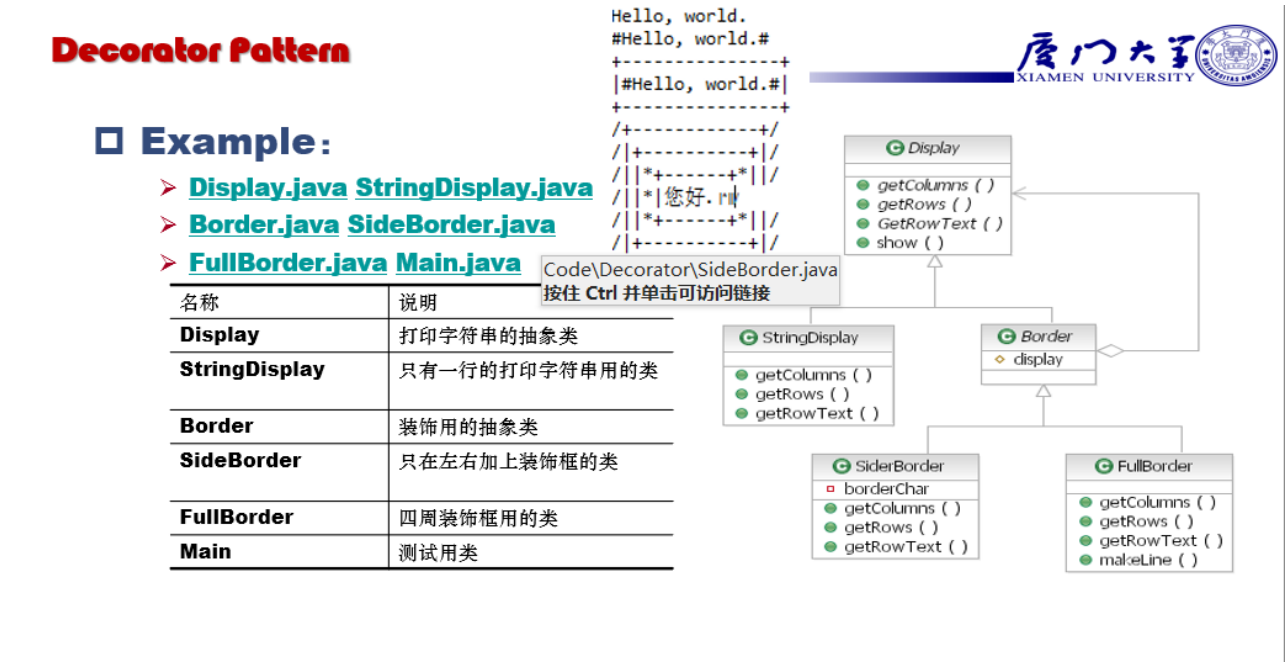
用具体装饰类型来定义装饰后的对象，而具体构件类型仍然可以使用抽象构件类型来定义，可以单独调用装饰的独有方法。（半透明装饰模式的具体装饰类没有override从父类继承的方法，调用父类方法和调用新增方法是分开进行的，导致装饰效果没有累加起来，但不会让程序运行出错）

（3）两者区别

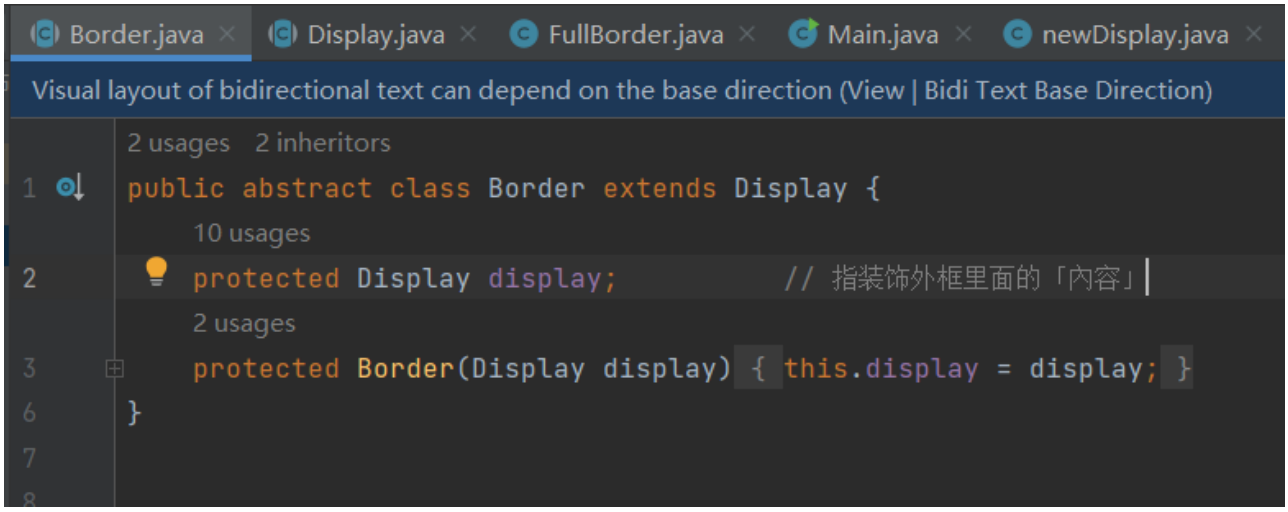
透明的装饰模式，要求具体构件角色、装饰角色的接口与抽象构件角色的接口完全一致。半透明装饰模式可以给系统带来更多的灵活性，但是其最大的缺点在于不能实现对同一个对象的多次装饰，客户端需要有区别地对待装饰之前的对象和装饰之后的对象。

（4）代码示例

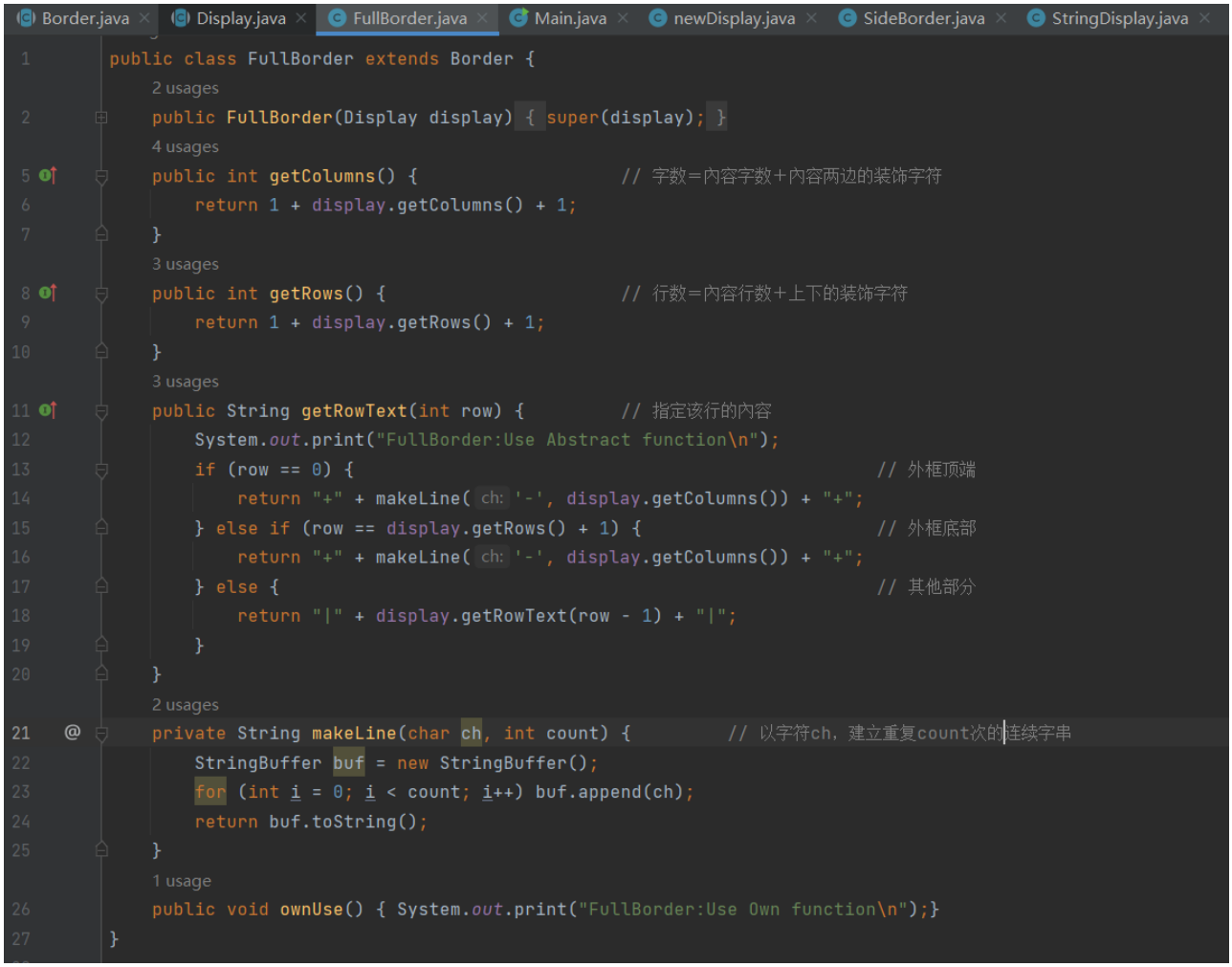
在下面代码中，Display是打印字符串的抽象类，相当于定义了对象接口的Component，在其中有getColumns、getRows、GetRowText、show四种方法。StringDisplay相当于ConcreComponent，定义了打印字符串的具体类。Border相当于抽象类装饰器。而SideBorder和FullBorder则是具体的抽象类，Main用于测试。

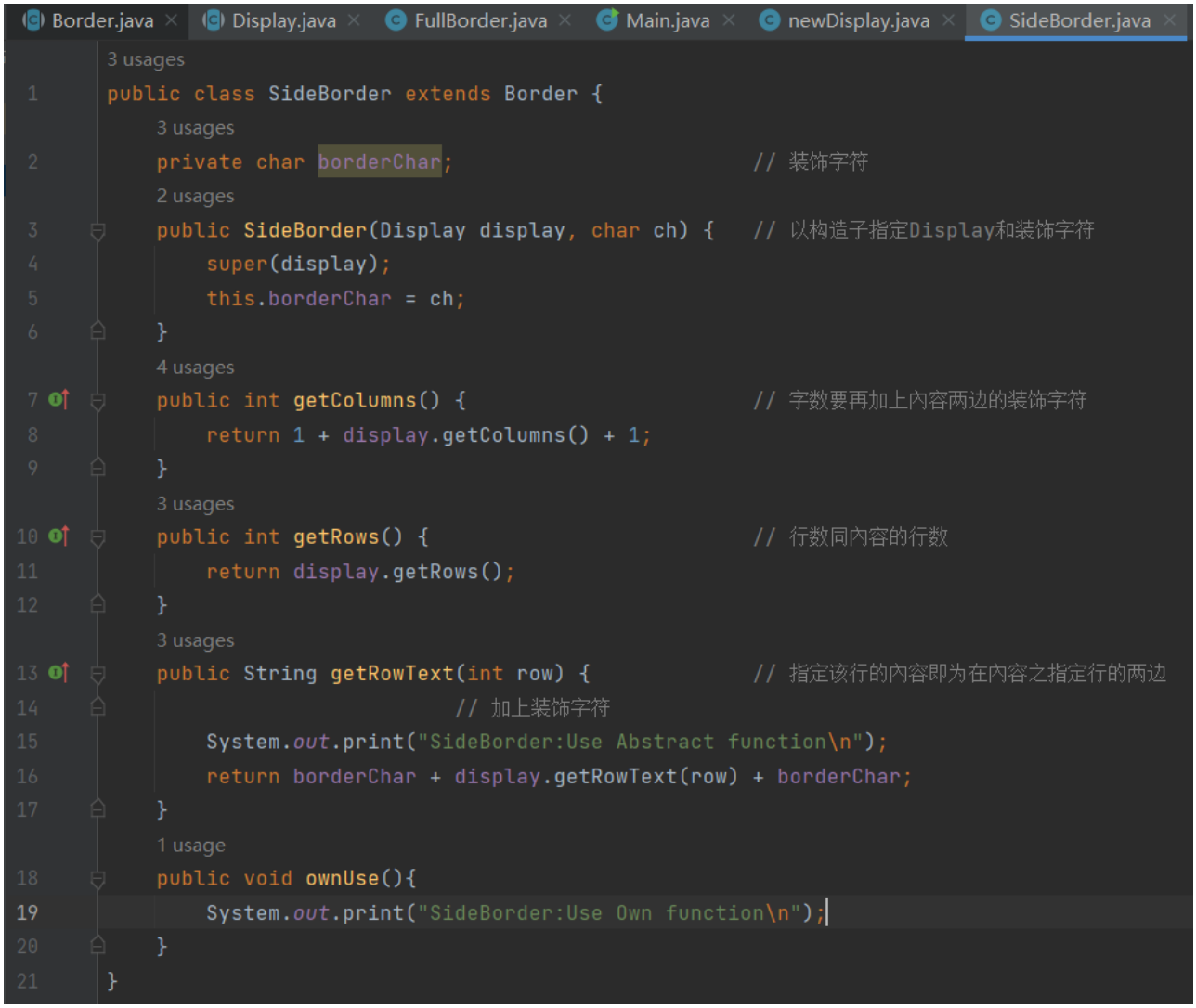


1. **抽象类装饰器Border**



2. **两个具体装饰类：FullBorder和SideBorder**

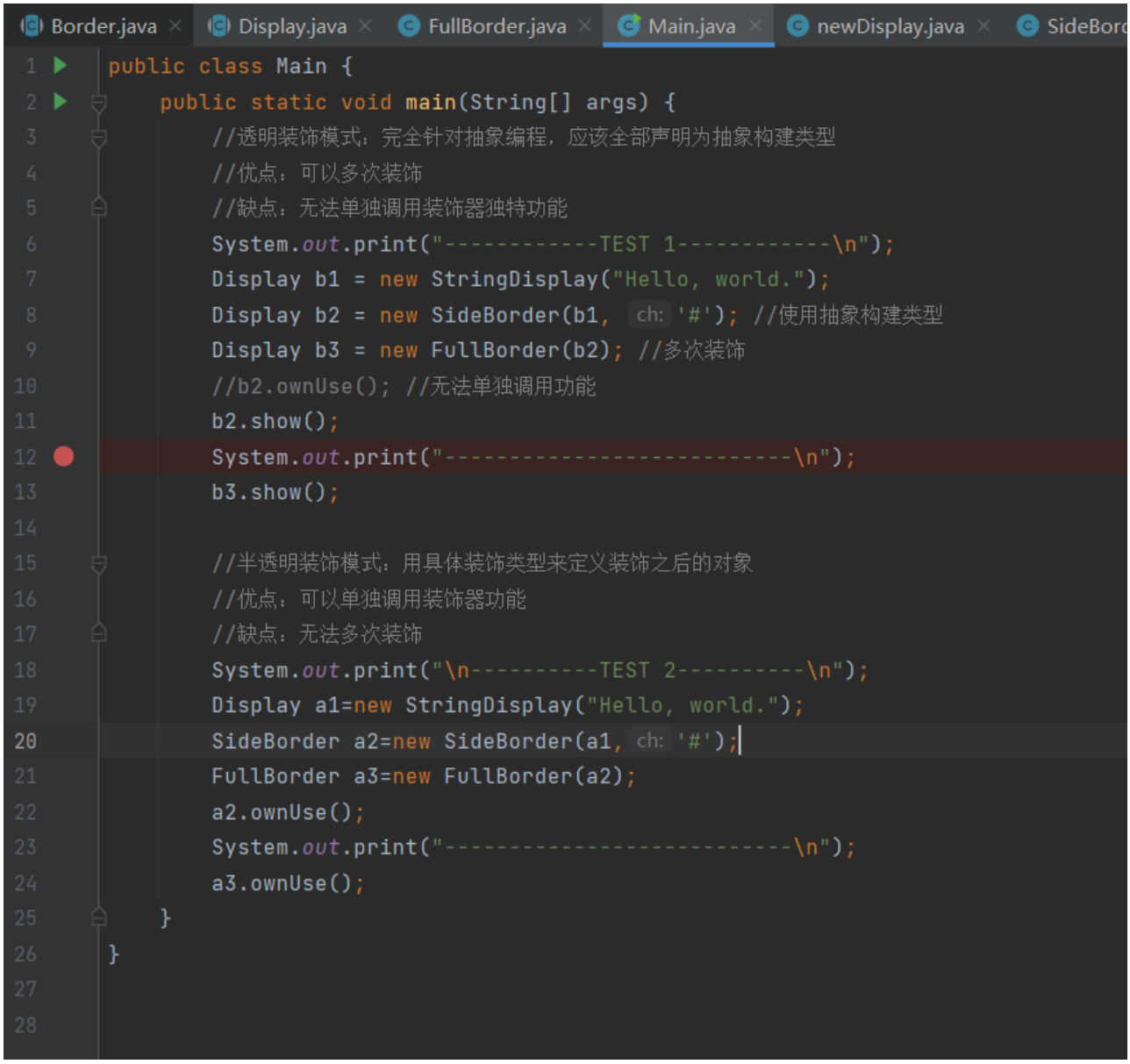

Border.java
Display.java
FullBorder.java
Main.java
newDisplay.java
SideBorder.java
StringDisplay.java
public class FullBorder extends Border {
2 usages
public FullBorder(Display display) { super(display); }
4 usages
public int getColumns() { // 字数＝内容字数＋内容两边的装饰字符
return 1 + display.getColumns() + 1;
}
3 usages
public int getRows() { // 行数＝内容行数＋上下的装饰字符
return 1 + display.getRows() + 1;
}
3 usages
public String getRowText(int row) { // 指定该行的内容
System.out.print("FullBorder:Use Abstract function\n");
if (row == 0) { // 外框顶端
return "+" + makeLine(ch: '-', display.getColumns()) + "+";
} else if (row == display.getRows() + 1) { // 外框底部
return "+" + makeLine(ch: '-', display.getColumns()) + "+";
} else { // 其他部分
return "|" + display.getRowText(row - 1) + "|";
}
}
2 usages
private String makeLine(char ch, int count) { // 以字符ch，建立重复count次的连续字串
StringBuffer buf = new StringBuffer();
for (int i = 0; i < count; i++) buf.append(ch);
return buf.toString();
}
1 usage
public void ownUse() { System.out.print("FullBorder:Use Own function\n");}
}



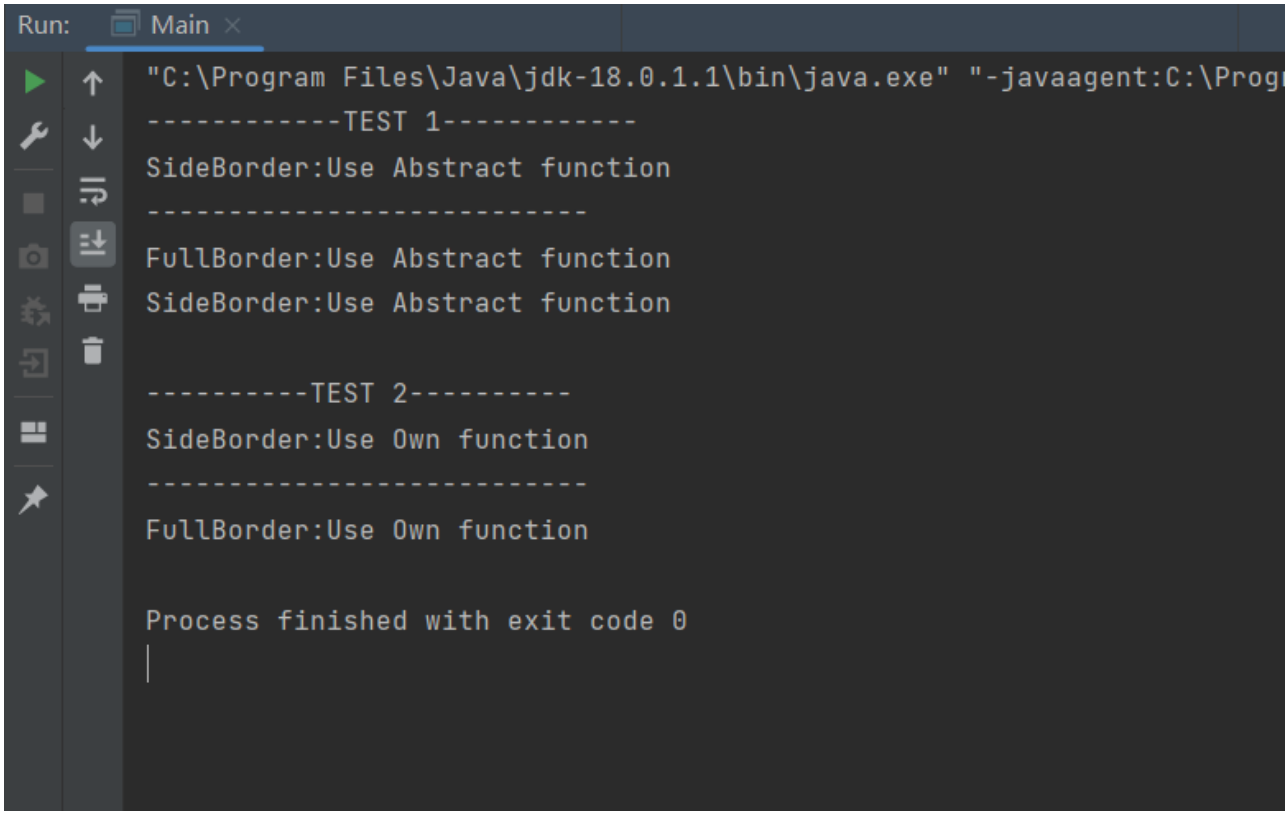
（5）运行结果说明

在Main中，编写了测试代码，当使用透明装饰模式时，由于完全针对抽象编程，当对b3使用display时不仅会调用FullBorder的display，还会调用SideBorder的display，从而实现多次装饰的效果。但是对于装饰器独有的功能ownUse，由于定义的时抽象类型，就无法直接调用了。

而在使用半透明装饰模式时，可以直接调用装饰器独有的功能ownUse，但是由于并没有直接使用抽象接口，因此这个方法是不能用于多次装饰的。



由输出结果也可以印证，Test1的透明装饰模式，对同一个对象用了两次装饰，而Test2的则会丢失，无法连续装饰。


Run:
Main
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe" "-javaagent:C:\Prog
------------TEST 1------------
SideBorder:Use Abstract function

FullBorder:Use Abstract function
SideBorder:Use Abstract function
----------TEST 2----------
SideBorder:Use Own function

FullBorder:Use Own function
Process finished with exit code 0