**Java程序设计 LAB06**

**实验目的：**

* 简单了解Object类
* 掌握良好重写Object类中方法的能力
* 简单了解内部类、匿名类

实验题目

本实验假设你明白：

* 涉及内部类时的初始化过程
* 内部类能够访问其外部类的所有属性和方法
* 构造内部类必须先构造外部类对象
* 嵌套类（静态内部类）只能够访问外部类的静态属性和方法
* 嵌套类可以直接通过类名构造
* 匿名内部类和局部内部类中直接使用的外部数据必须是final的（除非是匿名类的方法的参数）

上述这些语法层面的东西，可以尝试自行证明。

本次实验不会涉及以下内部类的常用技巧：

* 回调与闭包
* 事件与控制框架

也不会涉及以下内部类的边角情况：

* 内部类的继承与覆盖
* 多重嵌套的内部类

1. **InnerClass1**

阅读下面这段代码：

|  |
| --- |
| *// Test.java*  **class** Outer {  **public** **int** num = 10;  **class** Inner {  **public** **int** num = 20;  **public** **void** show() {  **int** num = 30;  System.out.println(*/\*???\*/*);  System.out.println(*/\*???\*/*);  System.out.println(*/\*???\*/*);  }  }  }  **public** **class** Test {  **public** **static** **void** main(String[] args) {  Outer.Inner oi = */\*???\*/*;  oi.show();  }  } |

在oo/01/Test.java中的注释部分填代码，使程序先后输出30、20、10。

**不允许修改已经有的代码。**

**考察的是内部类的构造以及访问外部类的方法，请不要用加减运算这种操作。**

1. **InnerClass2**

阅读下面这段代码：

|  |
| --- |
| *// Test.java*  **interface** Inter { **void** show(); }  **class** Outer {*/\*\*/*}  **public class** Test {  **public** **static** **void** main(String[] args) {  Outer.method().show();  }  } |

在oo/02/Test.java中的注释部分填代码，使程序输出"oo"。

**不允许修改已经有的代码。**

**考察的是匿名类，但是用内部类也可以实现。**

1. **ShapeFactory3**

在前面实验的Shape的基础上，定义一个满足如下需求的IShapeFactory接口：

* + 具有方法Shape makeShape(double a, double b) ，返回一个由a和b指定大小的形状；
    - 参数不合法时，返回null或抛出异常

用单例模式+工厂方法模式的思想修改矩形、椭圆、菱形类：

* + 每一个类都增设一个private static IShapeFactory factory字段
    - 类中的factory用于生成该类的形状对象
      * 比如Rectangle类中的factory，其makeShape方法返回Rectangle对象
    - 直接使用匿名类为factory进行静态初始化，不允许像ShapeFactory2那样定义工厂类
  + 进行其他的修改，使外界的其他类能够获取到factory并成功构造形状对象

选择你认为合适的方式编写测试类：

* + 你的测试类应该能够覆盖到所有等价类。
  + 测试形式可以是单元测试，被测对象的形式可以参考之前实验中的ShapeFactoriesTest.makeShape方法。
  + 在代码注释中（或者与代码一起提交一个readme），描述你的测试计划

**题外话：**

使用匿名类，依然是为每一个形状创建了一个对应的工厂，因此本质上依然是工厂方法模式，区别在于不用显式定义新的类（据说编码过程中，起名字是最麻烦的事情）。

工厂方法模式的应用中，每一种工厂通常只有一个实例，因此它经常和单例模式一起被使用。

1. **ShapeSequence**

这是一个主动要求大家造轮子的题，因此不允许继承或组合任何java自带的容器（除了数组），并且本题还会出现在后续实验。

本题假设大家在算法上机时能够较为熟练的使用C++中STL容器，并至少对变长容器的迭代访问有一定了解。

**如果对题目描述有疑问，请先参阅压缩包内的样例**。

在Shape的基础上，定义一个满足如下需求的ShapeSequence类：

* + 具有属性private Shape[] shapes
  + 构造方法ShapeList(int size)
    - size用于指定shapes的最大长度，如果size是负数，那么按照0来处理。
    - 构造方法中应当对shapes进行初始化赋值，在其他过程中shapes的大小不应该被改变
  + 方法public void add(Shape shape)
    - 向shapes中添加一个新的元素
    - 当shapes被填满时，什么都不做
  + 方法public String toString()
    - 返回这个容器的字符串表达，格式为[Type, Type,...]
    - 格式中的Type是形状类型的全小写英文单词，比如rectangle、ellipse
  + 方法public SequenceIterator iterator() {return new SequenceIterator();}
  + 具有内部类private class SequenceIterator，它用于序列遍历的迭代器
    - 默认构造方法，在被构造时，迭代器指向的位置代表数组下标0
    - 方法public boolean isEnd()，迭代器完成遍历时，返回true
      * 完成遍历不代表迭代器指向了最后一个元素，而是指向了最后一个元素的下一个位置
    - 方法public Shape current()，返回当前迭代器指向位置的Shape对象
      * isEnd()是true时，访问current是非法操作
    - 方法public void moveNext()，使迭代器移动到下一个元素的位置
      * isEnd()是true时，什么都不做
    - 方法public boolean equals(Object o)，当o是SequenceItetator类型的、且o和this的外部类对象相同、且o和this的位置相同时，返回true

编写测试类并描述你的测试计划。

**注意：**

**toString和equals是override继承自Object的方法，如果这次再出现诸如ToString、equals的参数类型不是Object（这种情况会算作overload，详情参阅LAB05反馈）等情况，直接不给分了。**

题目中省略了一些实现上必要但是方式不唯一的属性：比如你可以在ShapeSequence类中声明一个int属性来表示当前容器被填充到了什么位置；给SequenceIterator一个int属性表示当前迭代器指向的位置，当使用isEnd时，判断迭代器位置和外部类容器的填充位置进行比较。

**题外话：**

本题的目的是进行内部类、Object相关的综合实践，通过使用自定义的迭代器和容器提前复习C++基础并理解一些基础概念。

后续容器和泛型课程中，会引入可变长容器以及匿名类实现的迭代器。

虽然我们上机并不评测程序的性能，但是在实现本题的toString时，还是推荐考虑使用StringBuilder。

你可以考虑一下不使用内部类时要如何实现迭代器，内部类为这种数据访问是否带来了足够的便利？

为什么SequenceIterator被限定为了private？