软件体系结构作业 17

姓　名：洪伟鑫

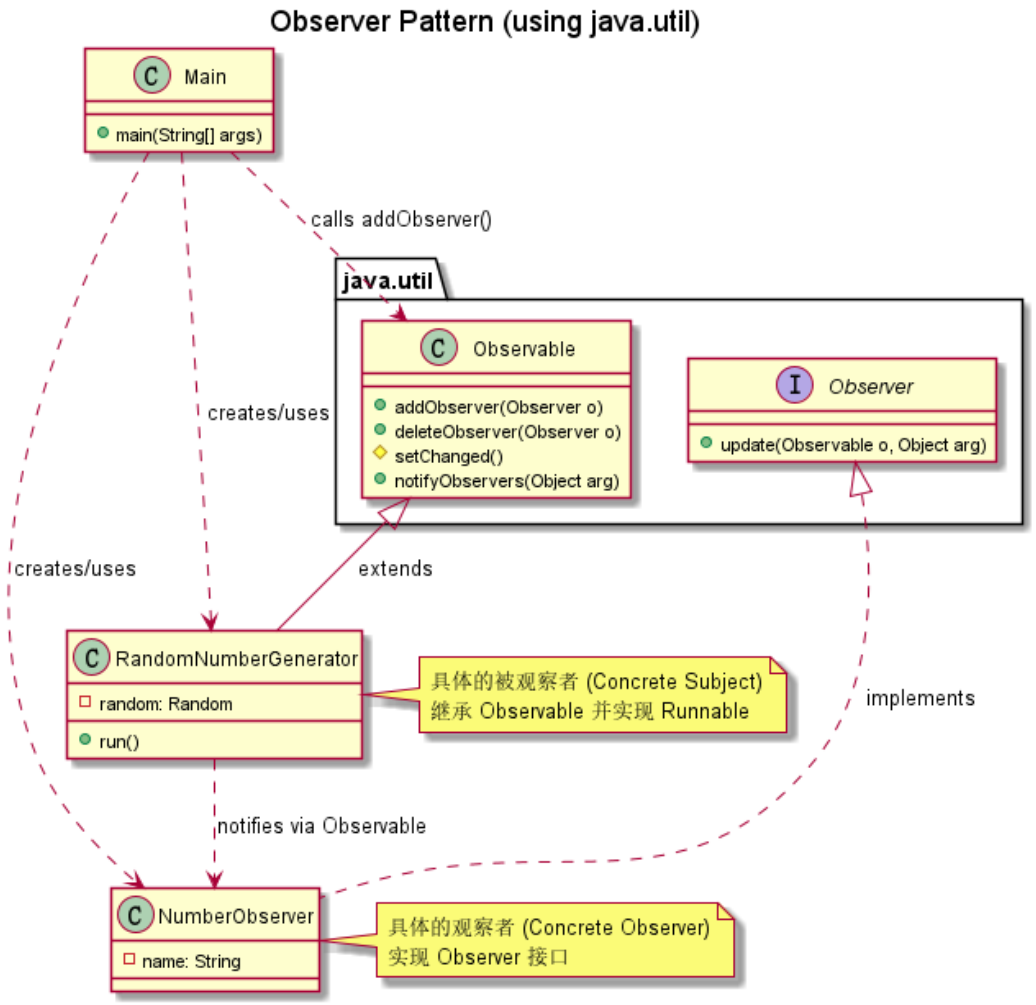
专　业：软件工程

年　级：2022 级

学　号：37220222203612

2025 年 5 月 4 日

利用 JDK 的 java.util 包中提供的 Observable 类以及 Observer 接口实现课堂的例子（对随机数的观察输出），将程序进行你要的修改或完善。



代码修改：

RandomNumberGenerator 继承 Observable 并实现了 Runnable 接口，以便在一个单独的线程中持续生成随机数。
在 run 方法中，每次生成随机数后，调用 setChanged() 表示状态已改变，然后调用 notifyObservers(randomNumber) 通知所有已注册的观察者，并将生成的随机数作为参数传递过去。
NumberObserver 实现 Observer 接口，其核心是 update 方法。当 notifyObservers 被调用时，每个注册的 NumberObserver 实例的 update 方法会被执行。
Main 类负责创建对象、建立观察关系并启动线程。

```java
import java.util.Observable;
import java.util.Observer;

/**
 * 观察者：负责观察随机数并输出
 * 实现 Observer 接口
 */
public class NumberObserver implements Observer {

    private String name;

    public NumberObserver(String name) {
        this.name = name;
    }

    /**
     * 当被观察者状态改变并调用 notifyObservers 时，此方法会被调用
     * @param o 被观察者对象（RandomNumberGenerator）
     * @param arg 被观察者传递过来的数据（随机数）
     */
    @Override
    public void update(Observable o, Object arg) {
        if (o instanceof RandomNumberGenerator) {
            if (arg instanceof Integer) {
                int randomNumber = (Integer) arg;
                System.out.println("观察者 [" + name + "] 收到新随机数: " + randomNumber);
                // 在这里可以根据随机数执行其他操作
            }
        }
    }
}
```

NumberObserver 观察者类

```java
import java.util.Vector;
import java.util.Iterator;

public abstract class NumberGenerator {
    private Vector observers = new Vector();
    public void addObserver(Observer observer) {
        observers.add(observer);
    }
    public void deleteObserver(Observer observer) {
        observers.remove(observer);
    }
    public void notifyObservers() {
        Iterator it = observers.iterator();

        while (it.hasNext()) {
            Observer o = (Observer)it.next();
            o.update(this);
        }
    }
    public abstract int getNumber();
    public abstract void execute();
}
```

被观察者类

```java
public class Main {
    public static void main(String[] args) {
        // 1. 创建被观察者（随机数生成器）
        RandomNumberGenerator generator = new RandomNumberGenerator();

        // 2. 创建观察者
        NumberObserver observer1 = new NumberObserver("观察者A");
        NumberObserver observer2 = new NumberObserver("观察者B");

        // 3. 注册观察者到被观察者
        generator.addObserver(observer1);
        generator.addObserver(observer2);

        // 4. 启动被观察者线程，开始生成随机数并通知观察者
        Thread generatorThread = new Thread(generator);
        System.out.println("启动随机数生成器...");
        generatorThread.start();

        // 让主线程等待一段时间，以便观察效果
        try {
            Thread.sleep(5000); // 运行 5 秒
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

Main 类

运行截图：

```
PS E:\大三资料\大三下课程资料\体系结构\PPT\Code\Observer> java Main
启动随机数生成器...
生成新随机数：82
观察者 [观察者B] 收到新随机数：82
观察者 [观察者A] 收到新随机数：82
生成新随机数：50
观察者 [观察者B] 收到新随机数：50
观察者 [观察者A] 收到新随机数：50
生成新随机数：18
观察者 [观察者B] 收到新随机数：18
观察者 [观察者A] 收到新随机数：18
生成新随机数：62
观察者 [观察者B] 收到新随机数：62
观察者 [观察者A] 收到新随机数：62
生成新随机数：9
观察者 [观察者B] 收到新随机数：9
观察者 [观察者A] 收到新随机数：9
生成新随机数：88
观察者 [观察者B] 收到新随机数：88
观察者 [观察者A] 收到新随机数：88
生成新随机数：12
观察者 [观察者B] 收到新随机数：12
观察者 [观察者A] 收到新随机数：12
```