SA第十六次作业

HomeWork16_1

1、阅读Gumballstate源码并改写成你想的（GUI？）。

(1)阅读代码并分析：

首先，定义了一个State接口，其中包含了糖果机可能的状态和对应的操作方法：插入硬币（insertQuarter）、退回硬币（ejectQuarter）、转动曲柄（turnCrank）、发放糖果（dispense），以及重新填充糖果（refill）的方法。

接着，定义了一个GumballMachine类来实现糖果机的具体功能。在构造方法中，初始化了糖果机的各种状态（售罄状态、无硬币状态、有硬币状态、售出状态），并根据糖果数量选择初始状态。insertQuarter()、ejectQuarter()、turnCrank()方法都委托给当前状态对象处理。releaseBall()方法用于释放一个糖果，并更新糖果数量。refill()方法用于重新填充糖果，并通知状态对象进行相应的处理。还提供了获取当前状态和各种状态对象的方法。

(2)改写：

新增一种新的状态类WinnerState，在此状态下，如果用户成功转动曲柄，会额外获得两颗糖果。

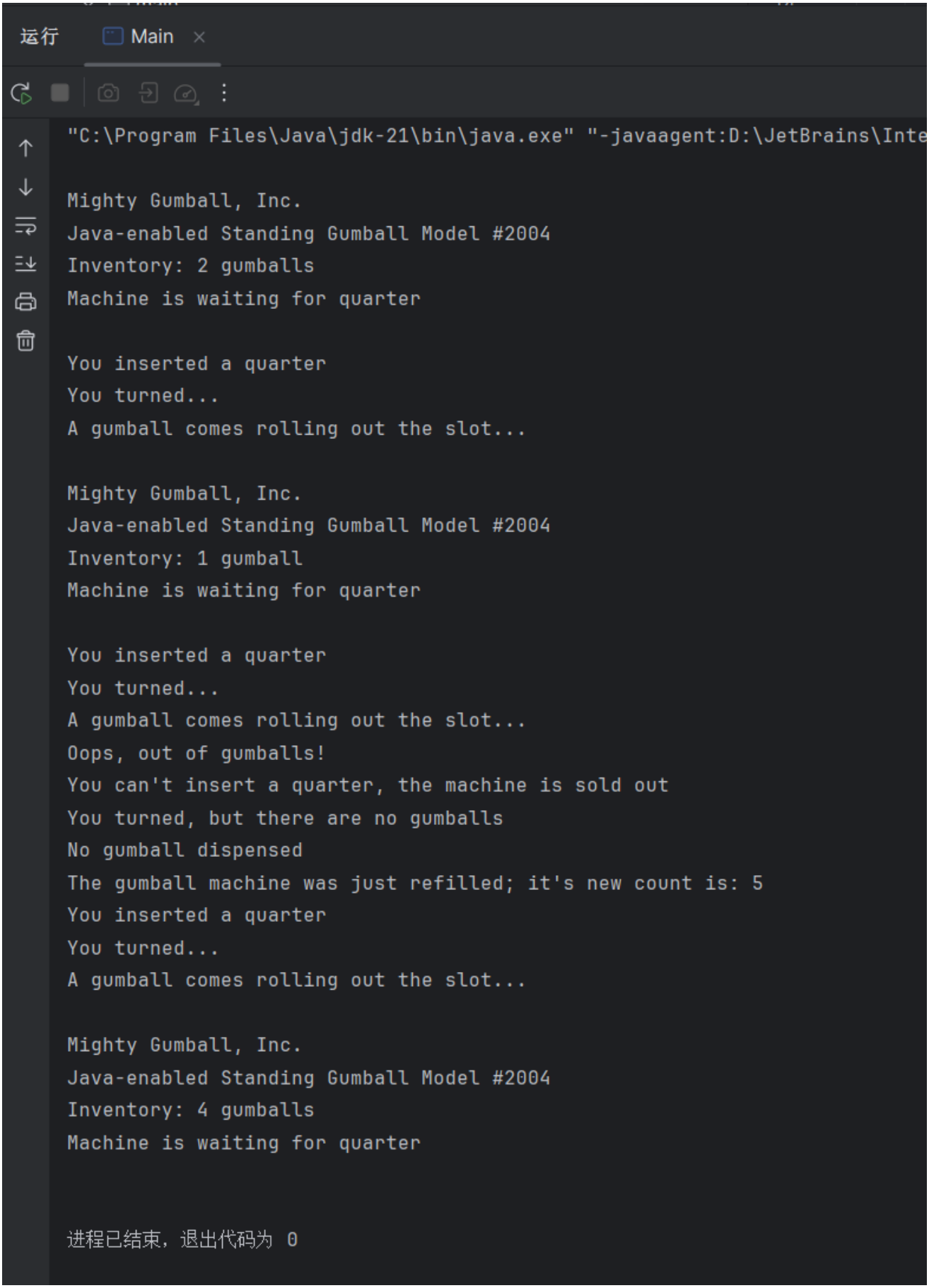
```
public class WinnerState implements State {
    GumballMachine gumballMachine;

    public WinnerState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

    @Override
    public void insertQuarter() {
        System.out.println("Please wait, we're already giving you a gumball");
    }

    @Override
    public void ejectQuarter() {
        System.out.println("Sorry, you already turned the crank");
    }

    @Override
    public void turnCrank() {
        System.out.println("Turning again doesn't get you another gumball!");
    }

    @Override
    public void dispense() {
        System.out.println("YOU'RE A WINNER! You get two gumballs for your quarter");
        gumballMachine.releaseBall();
        if (gumballMachine.getCount() == 0) {
            gumballMachine.setState(gumballMachine.getSoldOutState());
        } else {
```

```
            gumballMachine.releaseBall();
            if (gumballMachine.getCount() > 0) {
                gumballMachine.setState(gumballMachine.getNoQuarterState());
            } else {
                System.out.println("Oops, out of gumballs!");
                gumballMachine.setState(gumballMachine.getSoldOutState());
            }
        }
    }

    @Override
    public void refill() {
    }

    @Override
    public String toString() {
        return "despensing two gumballs for your quarter, because YOU'RE A WINNER!";
    }
}
```

修改GumballMachine中的turnCrank()方法：如果是WinnerState，则不需要dispense()，因为dispense()方法中已经处理了

```
public void turnCrank() {
    state.turnCrank();
    if (state instanceof WinnerState) {
    } else {
       state.dispense();
    }
}
```

运行结果：



HomeWork16_2

1、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。

创建目标类RandomNumberGenerator作为被观察的Subject，该类需要继承java.util包中提供的Observable类，每次进行变更，需要使用setChanged()将自身标注为已变更，并将变更的内容通过notifyObservers()将变更的内容通知给观察者。

```
public class RandomNumberGenerator extends Observable {
    private Random random = new Random();
    private int number;

    private int getNumber() {
        return number;
    }

    public void execute() {
        for(int i=0;i<10;i++){
            number= random.nextInt(50);
            setChanged();
            notifyObservers(number);
        }
    }
}
```

创建两种观察者类DigitObserver和GraphObserver，该类需要实现Observer接口并重写其中的update()方法，该方法即观察者观测到变更后的行为。

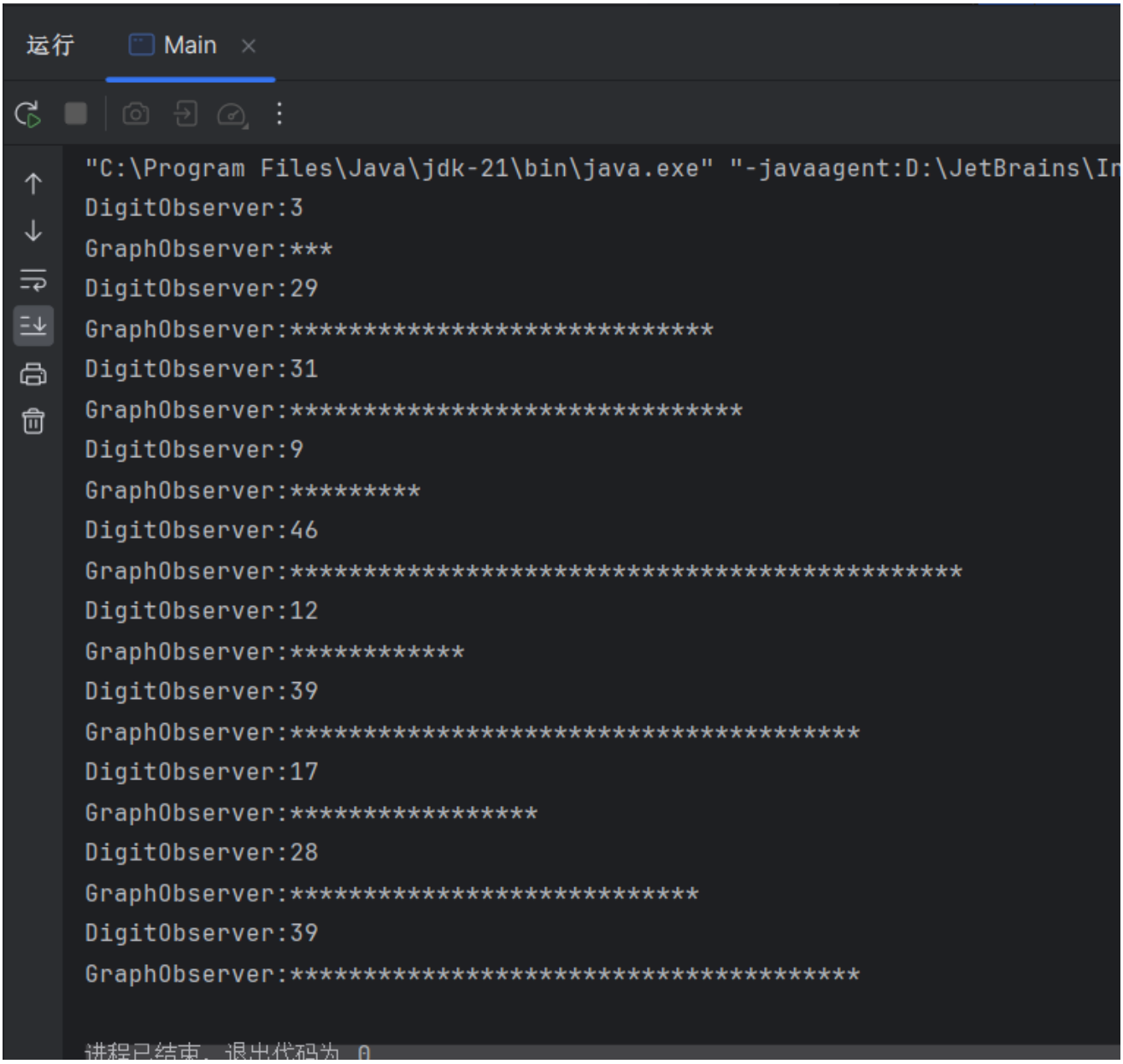
```
public class DigitObserver implements Observer {
    public void update(Observable o, Object arg) {
        System.out.println("DigitObserver:" + (int) arg);
    }
}
```

```
public class GraphObserver implements Observer {
    public void update(Observable o, Object arg) {
        System.out.print("GraphObserver:");
        for(int i=0;i<(int)arg;i++){
            System.out.print("*");
        }
        System.out.println();
    }
}
```

Main函数，创建一个目标对象和两个观察者对象，将观察者对象加入Subject的观察者序列中，调用Subject的运行方法。

```
public class Main {
    public static void main(String[] args) {
        RandomNumberGenerator randomNumberGenerator = new RandomNumberGenerator();
        DigitObserver digitObserver = new DigitObserver();
        GraphObserver graphObserver = new GraphObserver();
        randomNumberGenerator.addObserver(graphObserver);
        randomNumberGenerator.addObserver(digitObserver);
        randomNumberGenerator.execute();
    }
}
```

运行结果：


运行
Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\JetBrains\In
DigitObserver:3
GraphObserver:***
DigitObserver:29
GraphObserver:*****************************
DigitObserver:31
GraphObserver:*******************************
DigitObserver:9
GraphObserver:*********
DigitObserver:46
GraphObserver:**
DigitObserver:12
GraphObserver:************
DigitObserver:39
GraphObserver:***************************************
DigitObserver:17
GraphObserver:*****************
DigitObserver:28
GraphObserver:****************************
DigitObserver:39
GraphObserver:***************************************
进程已结束，退出代码为 0