软件体系结构 作业16

22920212204392 黄勖

1 阅读Gumballstate源码并改写成你想的（GUI？）。

改写了soldOutState（将其分为soldOutState和soldOutWithQuarterState），使得糖果机在没有quarter时也能接收投币，但是不能进行相应的摇杆操作

soldOutWithQuarterState如下

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

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

    public void insertQuarter() {
        System.out.println("You can't insert another quarter");
    }

    public void ejectQuarter() {
        System.out.println("Quarter returned");
        gumballMachine.setState(gumballMachine.getSoldOutState());
    }

    public void turnCrank() {
        System.out.println("You turned, but there are no gumballs");
    }

    public void dispense() {
        System.out.println("No gumball dispensed");
    }

    public void refill() {
        gumballMachine.setState(gumballMachine.getHasQuarterState());
    }

    public String toString() {
        return "sold out, but a quarter inside";
    }
}
```

在gumballMachine中也进行了同样的改动

```
2 个用法
State soldOutWithQuarterState;
```

```
soldOutWithQuarterState = new SoldOutWithQuarterState( gumballMachine: this);
```

```
1 个用法
public State getSoldOutWithQuarterState() { return soldOutWithQuarterState; }
```

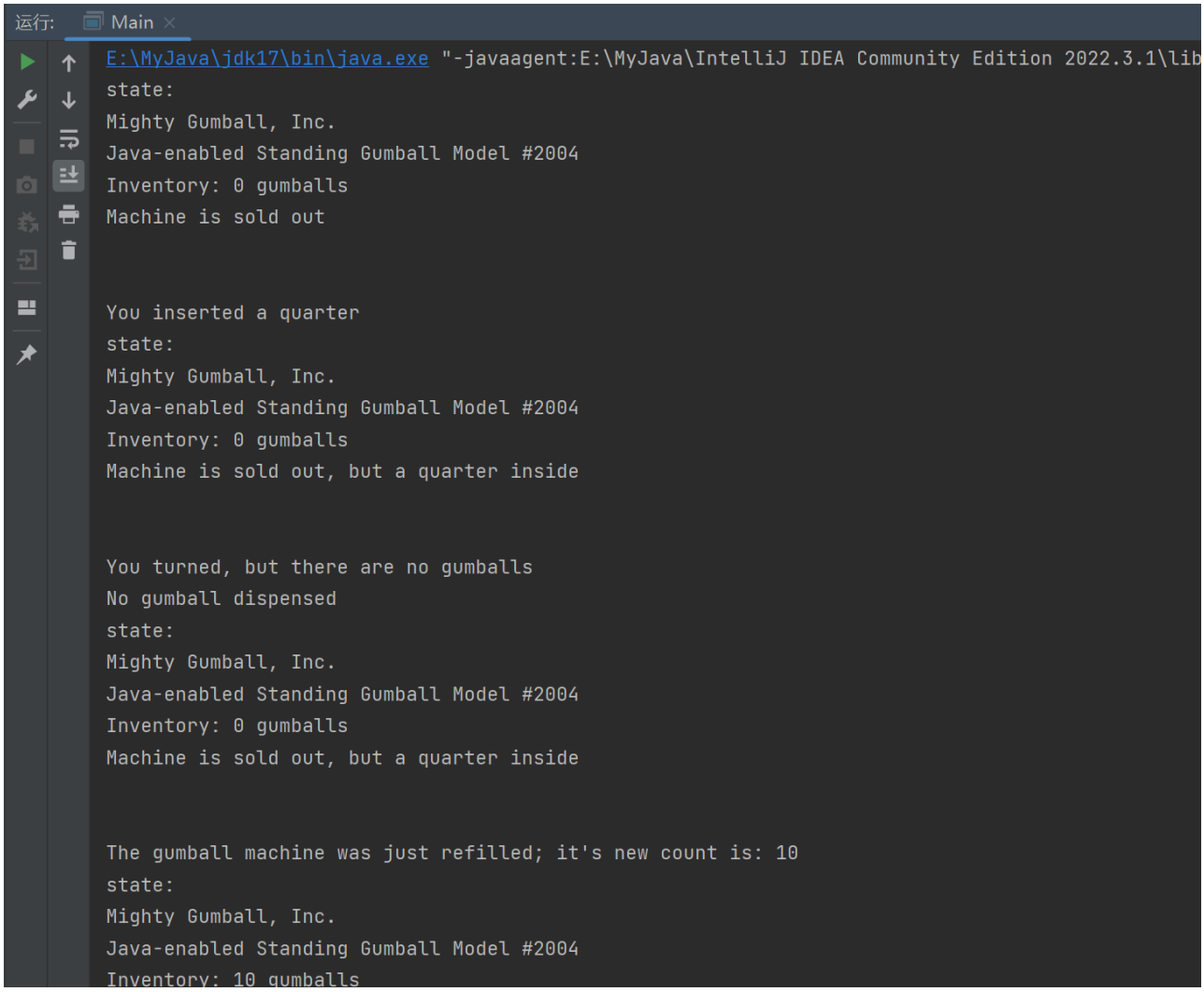
同样的，在soldOutState中，也有如下改动

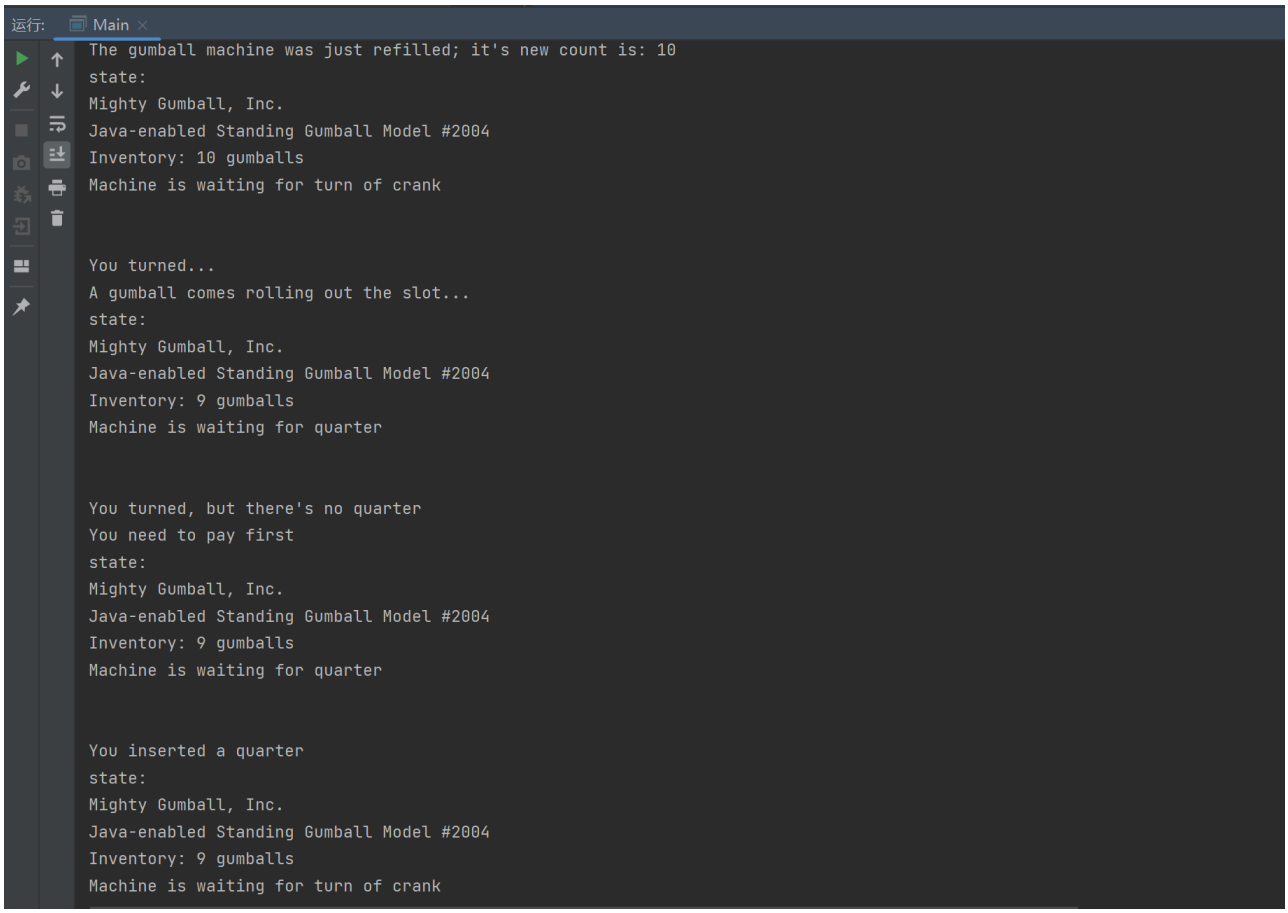
```
1 个用法
public void insertQuarter() {
    System.out.println("You inserted a quarter");
    gumballMachine.setState(gumballMachine.getSoldOutWithQuarterState());
}
```

Main函数如下：

```
public class Main {

    public static void main(String[] args) {

        GumballMachine gumballMachine = new GumballMachine(0);

        System.out.println("state: " + gumballMachine.toString() + "\n");
        gumballMachine.insertQuarter();
        System.out.println("state: " + gumballMachine.toString() + "\n");
        gumballMachine.turnCrank();
        System.out.println("state: " + gumballMachine.toString() + "\n");
        gumballMachine.refill(10);
        System.out.println("state: " + gumballMachine.toString() + "\n");
        gumballMachine.turnCrank();
        System.out.println("state: " + gumballMachine.toString() + "\n");
        gumballMachine.turnCrank();
        System.out.println("state: " + gumballMachine.toString() + "\n");
        gumballMachine.insertQuarter();
        System.out.println("state: " + gumballMachine.toString() + "\n");
    }
}
```

测试结果：


运行:
Main
E:\MyJava\jdk17\bin\java.exe "-javaagent:E:\MyJava\IntelliJ IDEA Community Edition 2022.3.1\lib
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 0 gumballs
Machine is sold out
You inserted a quarter
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 0 gumballs
Machine is sold out, but a quarter inside
You turned, but there are no gumballs
No gumball dispensed
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 0 gumballs
Machine is sold out, but a quarter inside
The gumball machine was just refilled; it's new count is: 10
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 10 gumballs


运行:
Main
The gumball machine was just refilled; it's new count is: 10
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 10 gumballs
Machine is waiting for turn of crank
You turned...
A gumball comes rolling out the slot...
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 9 gumballs
Machine is waiting for quarter
You turned, but there's no quarter
You need to pay first
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 9 gumballs
Machine is waiting for quarter
You inserted a quarter
state:
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 9 gumballs
Machine is waiting for turn of crank

2　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。

我们定义的RandomNumberGenerator类如下

```
import java.util.Observable;
import java.util.Random;

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

    public int getNumber() {
        return number;
    }

    public void execute(){
        for(int i = 0; i < 10; i++){
            number = random.nextInt(30);
            setChanged();
            notifyObservers();
        }
    }

}
```

他有两个观察者，`DigitObserver`和`GraphObeserver`

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

public class DigitObserver implements Observer {
    @Override
    public void update(Observable observable, Object obj){
        System.out.println("Get number: " + ((RandomNumberGenerator)
observable).getNumber());
        try{
            Thread.sleep(1000);
        }
        catch (InterruptedException e){}
    }

    public void subscribe(Observable observable){
        observable.addObserver(this);
    }
```

```
}
```

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

public class GraphObserver implements Observer {
    @Override
    public void update(Observable observable, Object obj){
        int num = ((RandomNumberGenerator) observable).getNumber();
        System.out.print("Get graph: ");
        for(int i = 0; i < num; i++)
            System.out.print('*');
        System.out.println();
        try{
            Thread.sleep(1000);
        }
        catch (InterruptedException e){}
    }

    public void subscribe(Observable observable){
        observable.addObserver(this);
    }
}
```

我们用以下的main函数测试

```
    public static void main(String[] args){
        RandomNumberGenerator randomNumberGenerator = new
RandomNumberGenerator();
        DigitObserver digitObserver = new DigitObserver();
        GraphObserver graphObserver = new GraphObserver();

        graphObserver.subscribe(randomNumberGenerator);
        digitObserver.subscribe(randomNumberGenerator);

        randomNumberGenerator.execute();
    }
```

结果如下

```
Get number: 1
Get graph: *
Get number: 24
Get graph: ************************
Get number: 14
Get graph: **************
Get number: 23
Get graph: ***********************
Get number: 28
Get graph: ****************************
Get number: 9
Get graph: *********
Get number: 21
Get graph: *********************
Get number: 21
Get graph: *********************
Get number: 21
Get graph: *********************
Get number: 11
Get graph: ***********
```