

## Homework:

### 1、 阅读 Gumballstate 源码并改写成你想的（GUI？）。

答：对于 GUI (图形化用户界面)，我通过 JavaFX 组件实现，首先创建几个按钮来模拟插入硬币、转动曲柄和重新填充糖果机的操作。而每个按钮的点击事件都会调用 GumballMachine 的相应方法，并更新文本状态显示当前的糖果机状态，设计的界面如下：



代码如下：

```
13 public class Main extends Application {
14
15     3 个用法
16     private Text statusText;
17     3 个用法
18     private Text actionText;
19
20     @Override
21     public void start(Stage primaryStage) {
22         GumballMachine gumballMachine = new GumballMachine( numberGumballs: 5);
23         statusText = new Text(gumballMachine.toString());
24         actionText = new Text( s: "Welcome to Gumball Machine!");
25
26         Button insertQuarterButton = new Button( s: "Insert Quarter");
27         insertQuarterButton.setOnAction(e -> {
28             String result = gumballMachine.insertQuarter();
29             updateText(gumballMachine, result);
30         });
31
32         Button turnCrankButton = new Button( s: "Turn Crank");
33         turnCrankButton.setOnAction(e -> {
34             String result = gumballMachine.turnCrank();
35             updateText(gumballMachine, result);
36         });
37
38         Button refillButton = new Button( s: "Refill Machine");
39         refillButton.setOnAction(e -> {
40             String res=gumballMachine.refill( numGumballs: 5);
41             updateText(gumballMachine, res);
42         });
43
44         VBox root = new VBox( v: 10, statusText, actionText, insertQuarterButton, turnCrankButton, refillButton);
45         root.setAlignment(Pos.CENTER);
46         root.setPadding(new Insets( v: 15));
47         root.setSpacing(10);
48
49         Scene scene = new Scene(root, v: 400, v1: 250);
50         primaryStage.setTitle("Gumball Machine");
51         primaryStage.setScene(scene);
52         primaryStage.show();
53
54         // 添加样式
55         scene.getStylesheets().add("style.css");
56     }
57 }
```

源码中各种提示都输出在控制台上，为了将状态变化的提示信息显示在

JavaFX 界面而非控制台上，需要更新 State 接口和实现，状态实现类应该能够返回关于它们操作的描述，而不是直接打印到控制台。

State 接口：

```
3 public interface State {  
4     String insertQuarter();  
5     String ejectQuarter();  
6     String turnCrank();  
7     String dispense();  
8     void refill();  
9     String toString();  
10 }
```

SoldState 类：

```
3 public class SoldState implements State {  
4     GumballMachine gumballMachine;  
5  
6     public SoldState(GumballMachine gumballMachine) { this.gumballMachine = gumballMachine; }  
7  
8     public String insertQuarter() {  
9         return "Please wait, we're already giving you a gumball";  
10    }  
11  
12    public String ejectQuarter() {  
13        return "Sorry, you already turned the crank";  
14    }  
15  
16    public String turnCrank() {  
17        return "Turning twice doesn't get you another gumball!";  
18    }  
19  
20    public String dispense() {  
21        gumballMachine.releaseBall();  
22        if (gumballMachine.getCount() > 0) {  
23            gumballMachine.setState(gumballMachine.getNoQuarterState());  
24        } else {  
25            gumballMachine.setState(gumballMachine.getSoldOutState());  
26            return "Oops, out of gumballs!";  
27        }  
28        return "A gumball comes rolling out the slot.";  
29    }  
30  
31    public void refill() {  
32    }  
33  
34    public String toString() { return "dispensing a gumball"; }  
35  
36 }  
37  
38  
39  
40
```

SoldOutState 类：

```

3 public class SoldOutState implements State {
4     3 个用法
5     GumballMachine gumballMachine;
6
7     1 个用法
8     public SoldOutState(GumballMachine gumballMachine) { this.gumballMachine = gumballMachine; }
9
10    1 个用法
11    public String insertQuarter() {
12        return "You can't insert a quarter, the machine is sold out";
13    }
14
15    1 个用法
16    public String ejectQuarter() {
17        return "You can't eject, you haven't inserted a quarter yet";
18    }
19
20    1 个用法
21    public String turnCrank() {
22        return "You turned, but there are no gumballs";
23    }
24
25    1 个用法
26    public String dispense() {
27        return "No gumball dispensed";
28    }
29
30    1 个用法
31    public void refill() {
32        gumballMachine.setState(gumballMachine.getNoQuarterState());
33    }
34
35    public String toString() {
36        return "sold out";
37    }
38 }

```

## HasQuarterState 类:

```

3 public class HasQuarterState implements State {
4     5 个用法
5     GumballMachine gumballMachine;
6
7     1 个用法
8     public HasQuarterState(GumballMachine gumballMachine) {
9         this.gumballMachine = gumballMachine;
10    }
11
12    1 个用法
13    public String insertQuarter() {
14        return "You can't insert another quarter";
15    }
16
17    1 个用法
18    public String ejectQuarter() {
19        gumballMachine.setState(gumballMachine.getNoQuarterState());
20        return "Quarter returned";
21    }
22
23    1 个用法
24    public String turnCrank() {
25        gumballMachine.setState(gumballMachine.getSoldState());
26        return "You turned...";
27    }
28
29    1 个用法
30    public String dispense() {
31        return "No gumball dispensed";
32    }
33
34    1 个用法
35    public void refill() {
36    }
37
38    public String toString() {
39        return "waiting for turn of crank";
40    }
41 }

```

## NoQuarterState 类:

```
3 public class NoQuarterState implements State {
4     3 个用法
5     GumballMachine gumballMachine;
6
7     1 个用法
8     public NoQuarterState(GumballMachine gumballMachine) { this.gumballMachine = gumballMachine; }
9
10    1 个用法
11    public String insertQuarter() {
12        gumballMachine.setState(gumballMachine.getHasQuarterState());
13        return "You inserted a quarter";
14    }
15
16    1 个用法
17    public String ejectQuarter() {
18        return "You haven't inserted a quarter";
19    }
20
21    1 个用法
22    public String turnCrank() {
23        return "You turned, but there's no quarter";
24    }
25
26    1 个用法
27    public String dispense() {
28        return "You need to pay first";
29    }
30
31    1 个用法
32    public void refill() {
33    }
34
35    public String toString() {
36        return "waiting for quarter";
37    }
38 }
```

同时, GumballMachine 的方法应修改为返回操作结果的字符串, 这些字符串将被 GUI 使用来更新显示。

```
3 public class GumballMachine {
4     3 个用法
5     State soldOutState;
6     3 个用法
7     State noQuarterState;
8     2 个用法
9     State hasQuarterState;
10    2 个用法
11    State soldState;
12    10 个用法
13    State state;
14    8 个用法
15    int count = 0;
16
17    1 个用法
18    public GumballMachine(int numberGumballs) {
19        soldOutState = new SoldOutState( gumballMachine: this);
20        noQuarterState = new NoQuarterState( gumballMachine: this);
21        hasQuarterState = new HasQuarterState( gumballMachine: this);
22        soldState = new SoldState( gumballMachine: this);
23        this.count = numberGumballs;
24        if (numberGumballs > 0) {
25            this.state = noQuarterState;
26        } else {
27            this.state = soldOutState;
28        }
29    }
30
31    1 个用法
32    public String insertQuarter() {
33        return state.insertQuarter();
34    }
35
36    0 个用法
```

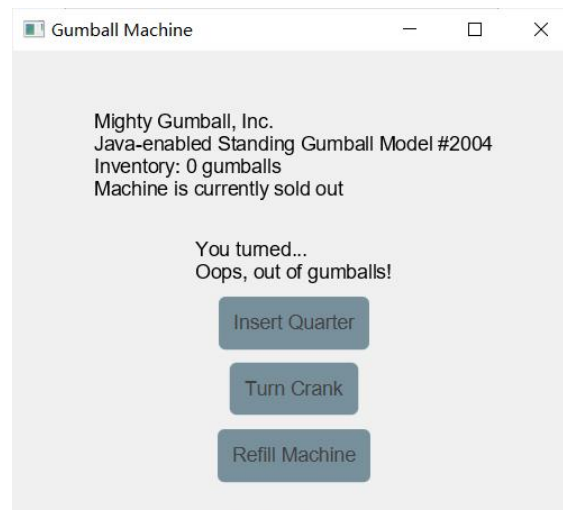
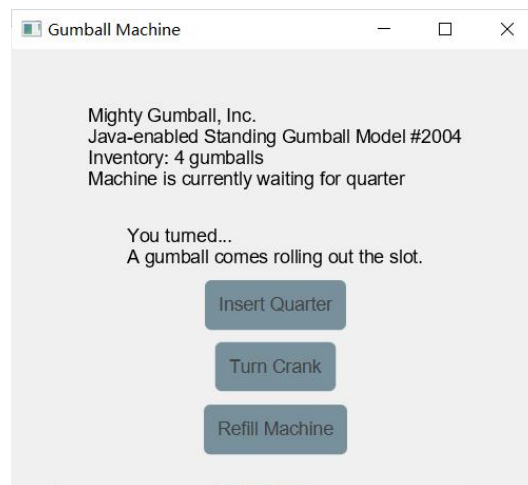
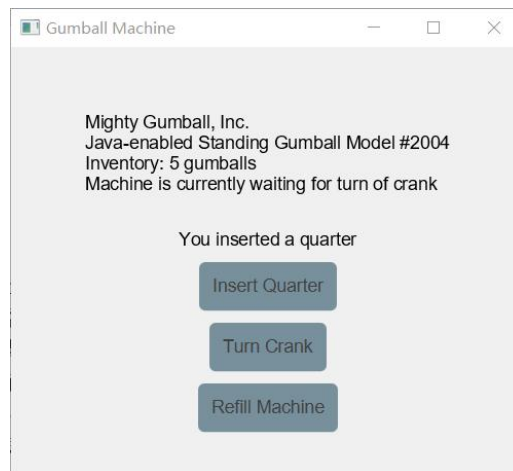
```
1 public interface State {
2     String insertQuarter();
3     String ejectQuarter();
4     String turnCrank();
5     String dispense();
6     void refill();
7     String toString();
8 }
```

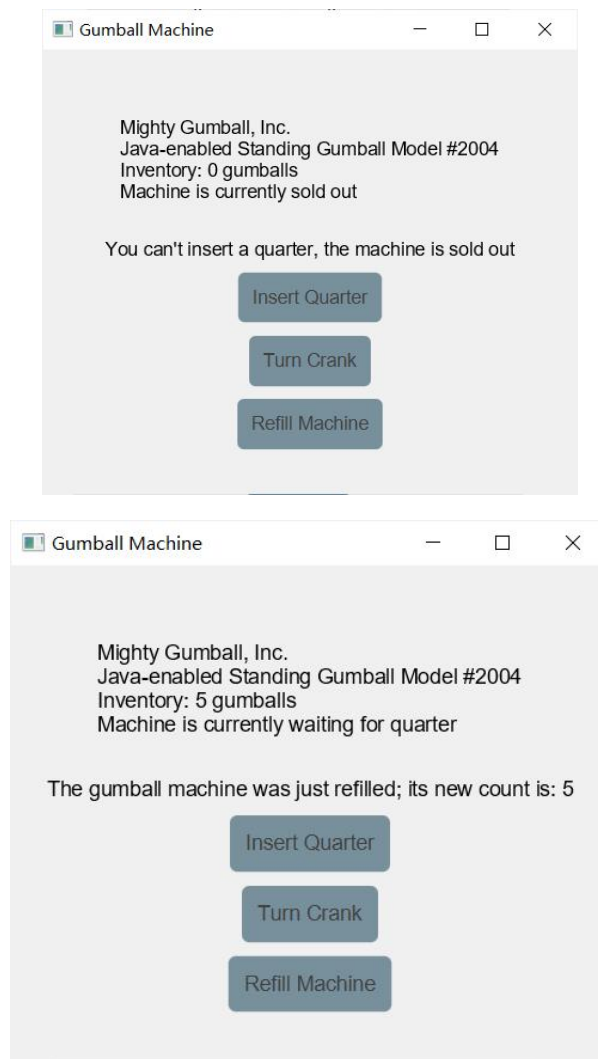
```

28 public String ejectQuarter() {
29     return state.ejectQuarter();
30 }
31
32 1 个用法
32 public String turnCrank() {
33     String crankResult = state.turnCrank();
34     String dispenseResult = state.dispense();
35     return crankResult + "\n" + dispenseResult; // Combine messages from turning the crank and dispensing
36 }
37
38 1 个用法
38 public void releaseBall() {
39     if (count > 0) {
40         count--;
41     }
42 }
43
44 1 个用法
44 int getCount() {
45     return this.count;
46 }
47
48 1 个用法
48 public String refill(int numGumballs) {
49     this.count += numGumballs;
50     state.refill();
51     return "The gumball machine was just refilled; its new count is: " + this.count;
52 }
53
54 6 个用法
54 void setState(State newState) {
55     this.state = newState;
56 }
57
58 0 个用法
58 public State getState() {
59     return this.state;
60 }
61
62 1 个用法
62 public State getSoldOutState() {
63     return soldOutState;
64 }
65
66 3 个用法
66 public State getNoQuarterState() {
67     return noQuarterState;
68 }
69
70 1 个用法
70 public State getHasQuarterState() {
71     return hasQuarterState;
72 }
73
74 1 个用法
74 public State getHasQuarterState() {
75     return hasQuarterState;
76 }
77
78 1 个用法
78 public String toString() {
79     StringBuffer result = new StringBuffer();
80     result.append("\nMighty Gumball, Inc.");
81     result.append("\nJava-enabled Standing Gumball Model #2004");
82     result.append("\nInventory: " + count + " gumball");
83     if (count != 1) {
84         result.append("s");
85     }
86     result.append("\nMachine is currently " + state + "\n");
87     return result.toString();
88 }
89 }
90
91

```

运行程序：





在本例中，State 设计模式非常适用，因为糖果机的行为直接依赖于其当前的状态。每个状态下糖果机的反应（比如投币、转动曲柄、补充糖果等）都不相同。通过使用 State 模式，我们可以将每种状态下的行为封装在各自的状态类中，避免在糖果机类中使用复杂的条件判断语句来决定行为，使得代码更加容易维护。

## 2、 附录

### 1) State

```
public interface State {  
    String insertQuarter();  
    String ejectQuarter();  
    String turnCrank();  
    String dispense();  
    void refill();  
    String toString();  
}
```

### 2) SoldState



```

public class SoldState implements State {
    GumballMachine gumballMachine;

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

    public String insertQuarter() {
        return "Please wait, we're already giving you a gumball";
    }

    public String ejectQuarter() {
        return "Sorry, you already turned the crank";
    }

    public String turnCrank() {
        return "Turning twice doesn't get you another gumball!";
    }

    public String dispense() {
        gumballMachine.releaseBall();
        if (gumballMachine.getCount() > 0) {
            gumballMachine.setState(gumballMachine.getNoQuarterState());
        } else {
            gumballMachine.setState(gumballMachine.getSoldOutState());
            return "Oops, out of gumballs!";
        }
        return "A gumball comes rolling out the slot.";
    }

    public void refill() {
    }

    public String toString() {
        return "dispensing a gumball";
    }
}

```

### 3) SoldOutState

```

public class SoldOutState implements State {
    GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gumballMachine) {

```



```

        this.gumballMachine = gumballMachine;
    }

    public String insertQuarter() {
        return "You can't insert a quarter, the machine is sold out";
    }

    public String ejectQuarter() {
        return "You can't eject, you haven't inserted a quarter yet";
    }

    public String turnCrank() {
        return "You turned, but there are no gumballs";
    }

    public String dispense() {
        return "No gumball dispensed";
    }

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

    public String toString() {
        return "sold out";
    }
}

```

#### 4) NoQuarterState

```

public class NoQuarterState implements State {
    GumballMachine gumballMachine;

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

    public String insertQuarter() {
        gumballMachine.setState(gumballMachine.getHasQuarterState());
        return "You inserted a quarter";
    }

    public String ejectQuarter() {
        return "You haven't inserted a quarter";
    }
}

```

```

    public String turnCrank() {
        return "You turned, but there's no quarter";
    }

    public String dispense() {
        return "You need to pay first";
    }

    public void refill() {
    }

    public String toString() {
        return "waiting for quarter";
    }
}

```

#### 5) HasQuarterState

```

public class HasQuarterState implements State {
    GumballMachine gumballMachine;

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

    public String insertQuarter() {
        return "You can't insert another quarter";
    }

    public String ejectQuarter() {
        gumballMachine.setState(gumballMachine.getNoQuarterState());
        return "Quarter returned";
    }

    public String turnCrank() {
        gumballMachine.setState(gumballMachine.getSoldState());
        return "You turned...";
    }

    public String dispense() {
        return "No gumball dispensed";
    }

    public void refill() {
    }
}

```

```

    }

    public String toString() {
        return "waiting for turn of crank";
    }
}

```

## 6) GumballMachine

```

public class GumballMachine {
    State soldOutState;
    State noQuarterState;
    State hasQuarterState;
    State soldState;
    State state;
    int count = 0;

    public GumballMachine(int numberGumballs) {
        soldOutState = new SoldOutState(this);
        noQuarterState = new NoQuarterState(this);
        hasQuarterState = new HasQuarterState(this);
        soldState = new SoldState(this);
        this.count = numberGumballs;
        if (numberGumballs > 0) {
            this.state = noQuarterState;
        } else {
            this.state = soldOutState;
        }
    }

    public String insertQuarter() {
        return state.insertQuarter();
    }

    public String ejectQuarter() {
        return state.ejectQuarter();
    }

    public String turnCrank() {
        String crankResult = state.turnCrank();
        String dispenseResult = state.dispense();
        return crankResult + "\n" + dispenseResult; // Combine messages
        from turning the crank and dispensing
    }
}

```

```

public void releaseBall() {
    if (count > 0) {
        count--;
    }
}

int getCount() {
    return this.count;
}

public String refill(int numGumballs) {
    this.count += numGumballs;
    state.refill();
    return "The gumball machine was just refilled; its new count is:
" + this.count;
}

void setState(State newState) {
    this.state = newState;
}

public State getState() {
    return this.state;
}

public State getSoldOutState() {
    return soldOutState;
}

public State getNoQuarterState() {
    return noQuarterState;
}

public State getHasQuarterState() {
    return hasQuarterState;
}

public State getSoldState() {
    return soldState;
}

public String toString() {
    StringBuffer result = new StringBuffer();
    result.append("\nMighty Gumball, Inc.");

```

```

        result.append("\nJava-enabled Standing Gumball Model #2004");
        result.append("\nInventory: " + count + " gumball");
        if (count != 1) {
            result.append("s");
        }
        result.append("\nMachine is currently " + state + "\n");
        return result.toString();
    }
}

```

## 7) Style.css

```

.root {
    -fx-font-family: 'Arial';
    -fx-font-size: 14px;
    -fx-background-color: #f0f0f0;
}

.button {
    -fx-padding: 10;
    -fx-background-color: #78909C;
    -fx-text-fill: white;
    -fx-border-radius: 5;
    -fx-background-radius: 5;
}

.button:hover {
    -fx-background-color: #546E7A;
}

.text {
    -fx-fill: #424242;
}

```

## 8) Main

```

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.Stage;

```

```

public class Main extends Application {

    private Text statusText;
    private Text actionText;

    @Override
    public void start(Stage primaryStage) {
        GumballMachine gumballMachine = new GumballMachine(5);
        statusText = new Text(gumballMachine.toString());
        actionText = new Text("Welcome to Gumball Machine!");

        Button insertQuarterButton = new Button("Insert Quarter");
        insertQuarterButton.setOnAction(e -> {
            String result = gumballMachine.insertQuarter();
            updateText(gumballMachine, result);
        });

        Button turnCrankButton = new Button("Turn Crank");
        turnCrankButton.setOnAction(e -> {
            String result = gumballMachine.turnCrank();
            updateText(gumballMachine, result);
        });

        Button refillButton = new Button("Refill Machine");
        refillButton.setOnAction(e -> {
            String res=gumballMachine.refill(5);
            updateText(gumballMachine, res);
        });

        VBox root = new VBox(10, statusText, actionText,
insertQuarterButton, turnCrankButton, refillButton);
        root.setAlignment(Pos.CENTER);
        root.setPadding(new Insets(15));
        root.setSpacing(10);

        Scene scene = new Scene(root, 400, 250);
        primaryStage.setTitle("Gumball Machine");
        primaryStage.setScene(scene);
        primaryStage.show();
        scene.getStylesheets().add("style.css");
    }

    private void updateText(GumballMachine gumballMachine, String
actionResult) {

```

```
        statusText.setText(gumballMachine.toString());  
        actionText.setText(actionResult);  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```