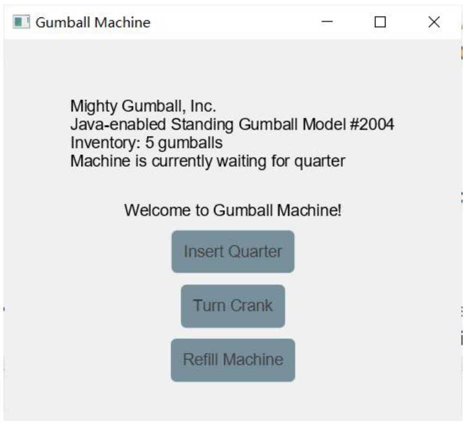
Homework：

1、　阅读 Gumballstate 源码并改写成你想的（GUI？）。

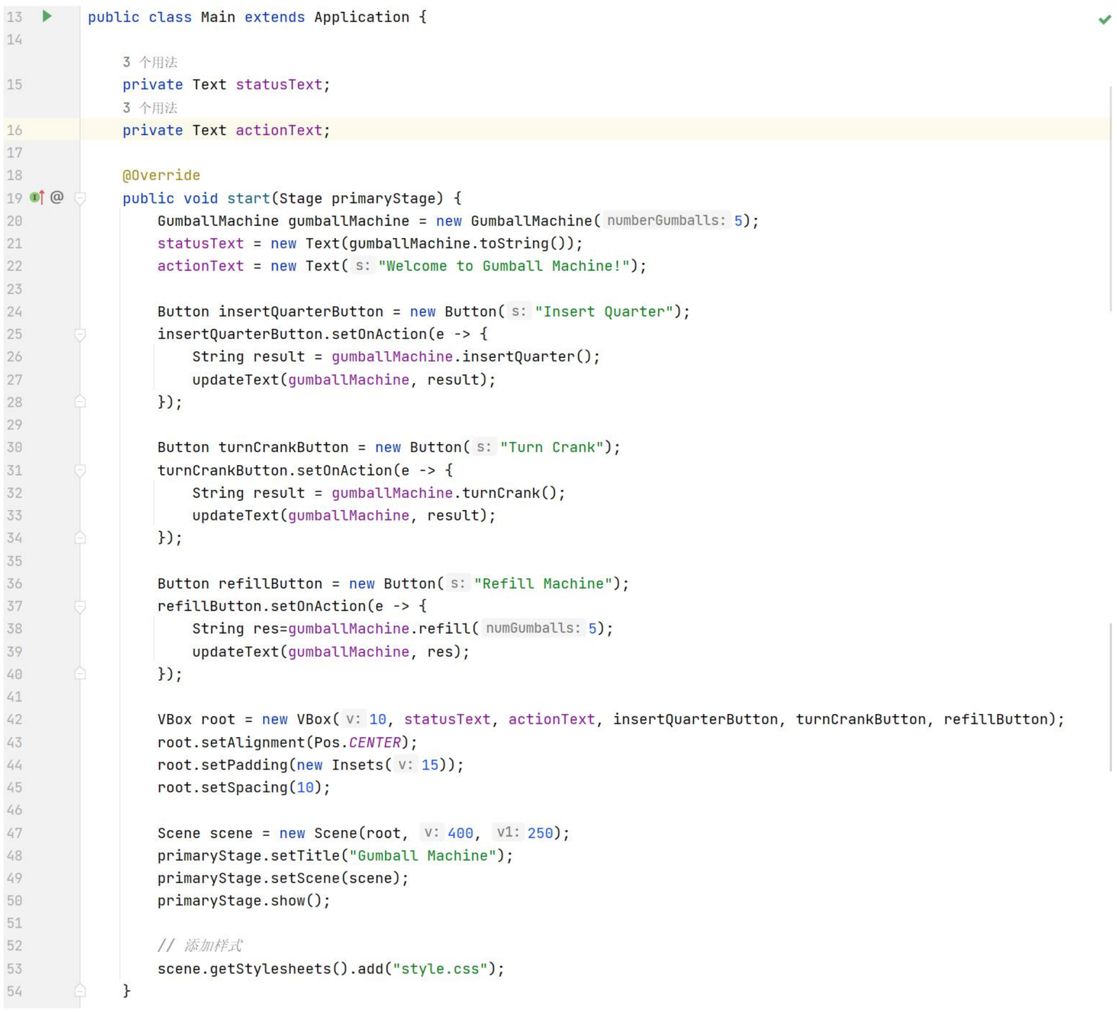
答：对于 GUI(图形化用户界面)，我通过 JavaFX 组件实现，首先创建几个按钮来模拟插入硬币、转动曲柄和重新填充糖果机的操作。而每个按钮的点击事件都会调用 GumballMachine 的相应方法，并更新文本状态显示当前的糖果机状态，设计的界面如下：



Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 5 gumballs
Machine is currently waiting for quarter

Welcome to Gumball Machine!

代码如下：

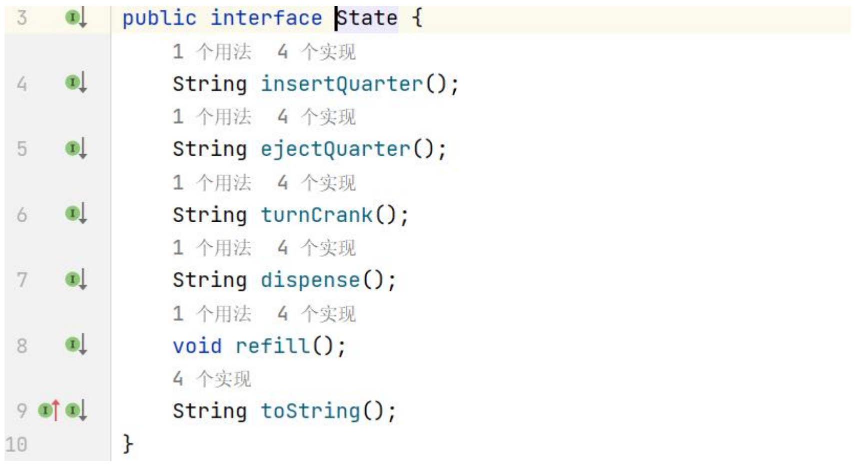


```java
public class Main extends Application {

    3 个用法
    private Text statusText;
    3 个用法
    private Text actionText;

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

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

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

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

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

        Scene scene = new Scene(root,  v: 400,  v1: 250);
        primaryStage.setTitle("Gumball Machine");
        primaryStage.setScene(scene);
        primaryStage.show();

        // 添加样式
        scene.getStylesheets().add("style.css");
    }
```

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

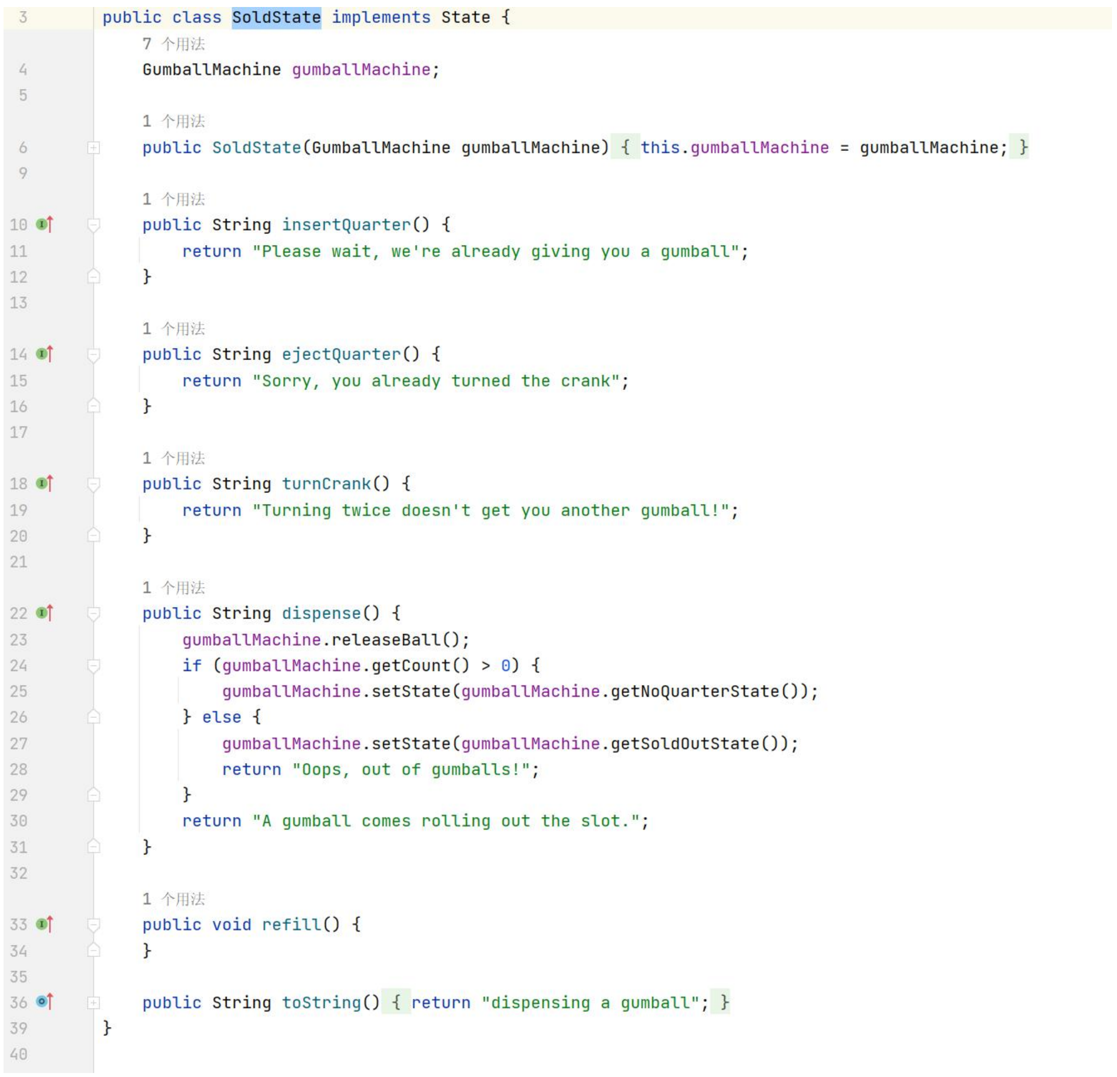
JavaFX 界面而非控制台上，需要更新 State 接口和实现，状态实现类应该能够返回关于它们操作的描述，而不是直接打印到控制台。

State 接口：



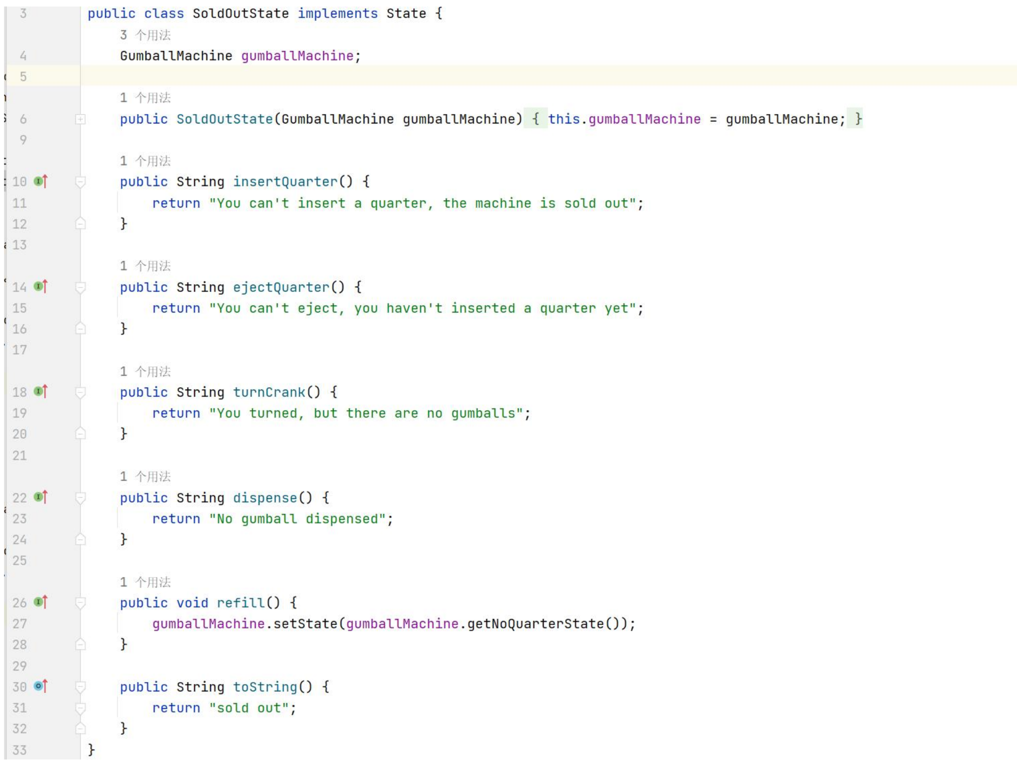
```
public interface State {
    1 个用法   4 个实现
    String insertQuarter();
    1 个用法   4 个实现
    String ejectQuarter();
    1 个用法   4 个实现
    String turnCrank();
    1 个用法   4 个实现
    String dispense();
    1 个用法   4 个实现
    void refill();
    4 个实现
    String toString();
}
```

SoldState 类：



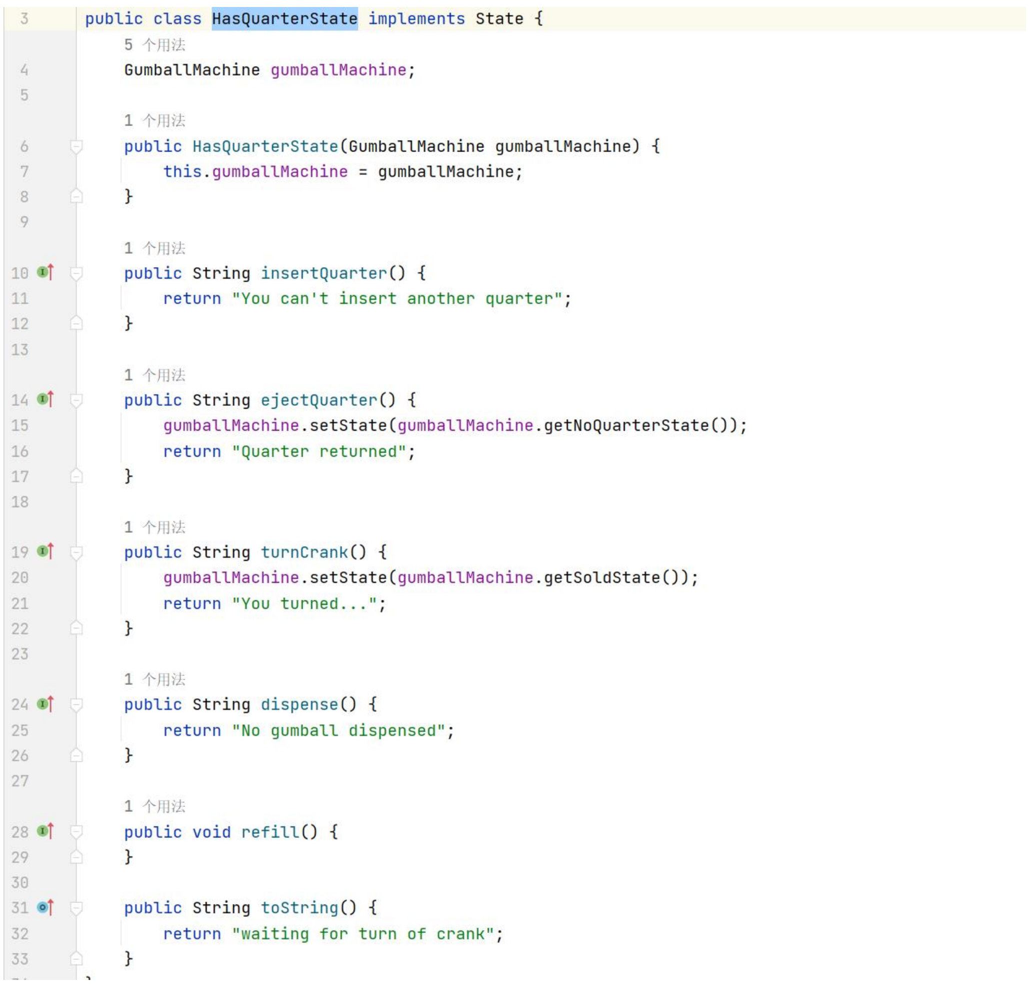
```
public class SoldState implements State {
    7 个用法
    GumballMachine gumballMachine;

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

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

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

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

    1 个用法
    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.";
    }

    1 个用法
    public void refill() {
    }

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

SoldOutState 类：



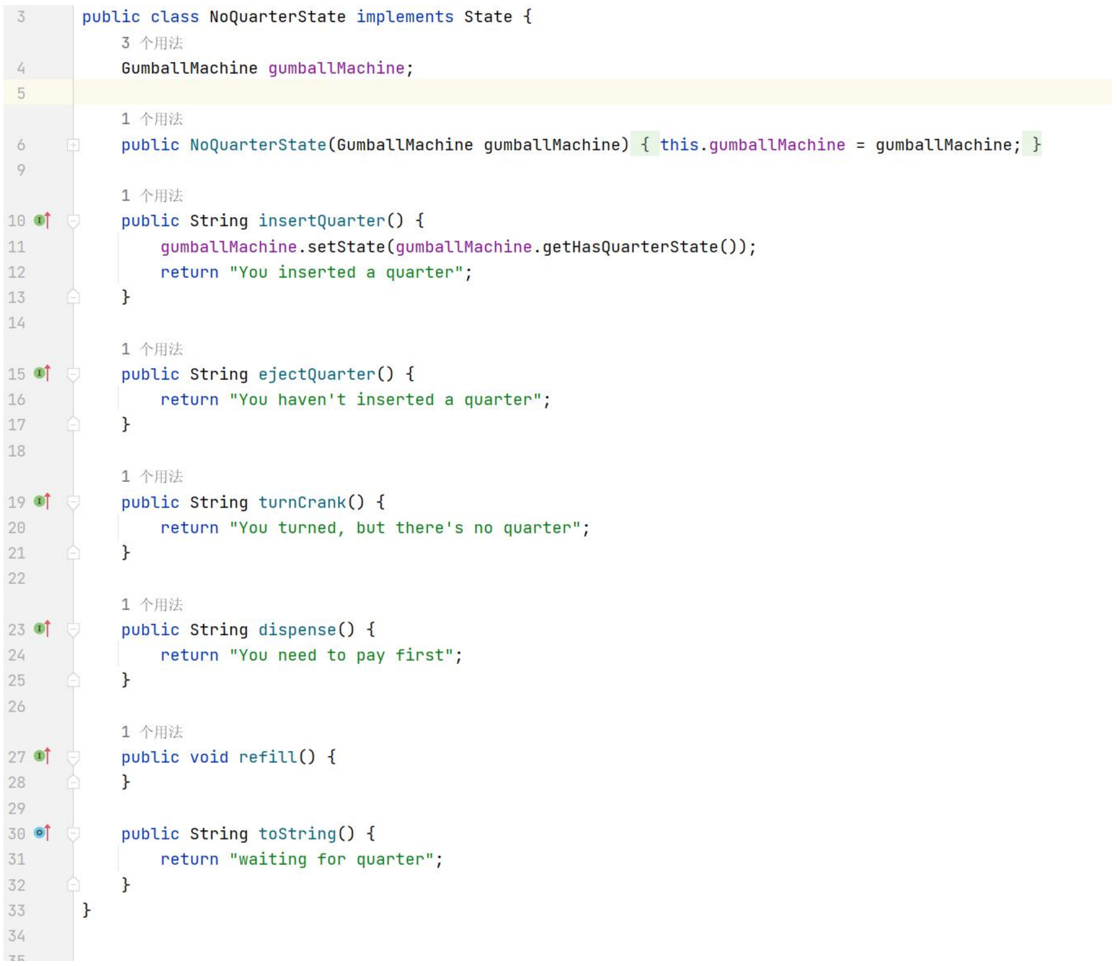
```java
public class SoldOutState implements State {
    3 个用法
    GumballMachine gumballMachine;

    1 个用法
    public SoldOutState(GumballMachine gumballMachine) { this.gumballMachine = gumballMachine; }

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

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

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

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

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

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

HasQuarterState 类:



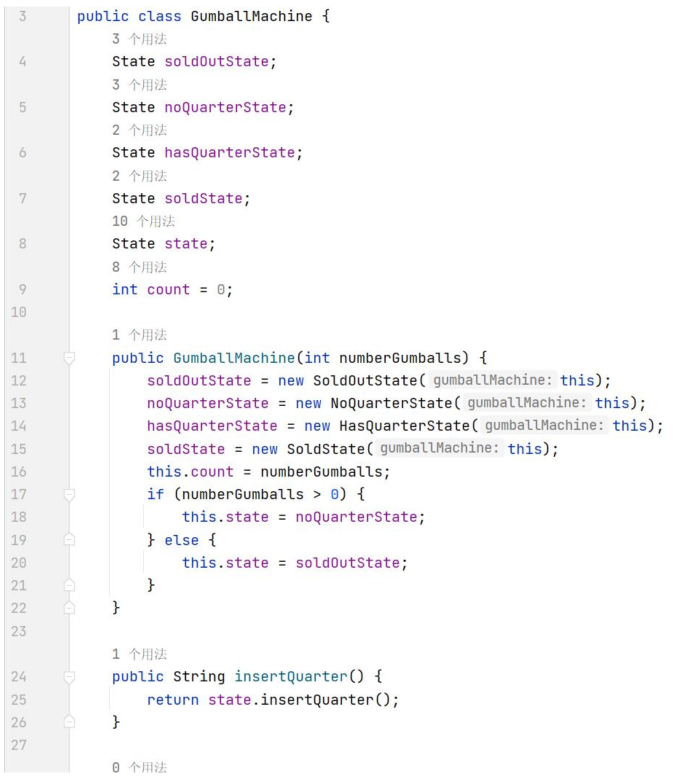
```java
public class HasQuarterState implements State {
    5 个用法
    GumballMachine gumballMachine;

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

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

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

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

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

    1 个用法
    public void refill() {
    }

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

NoQuarterState 类：

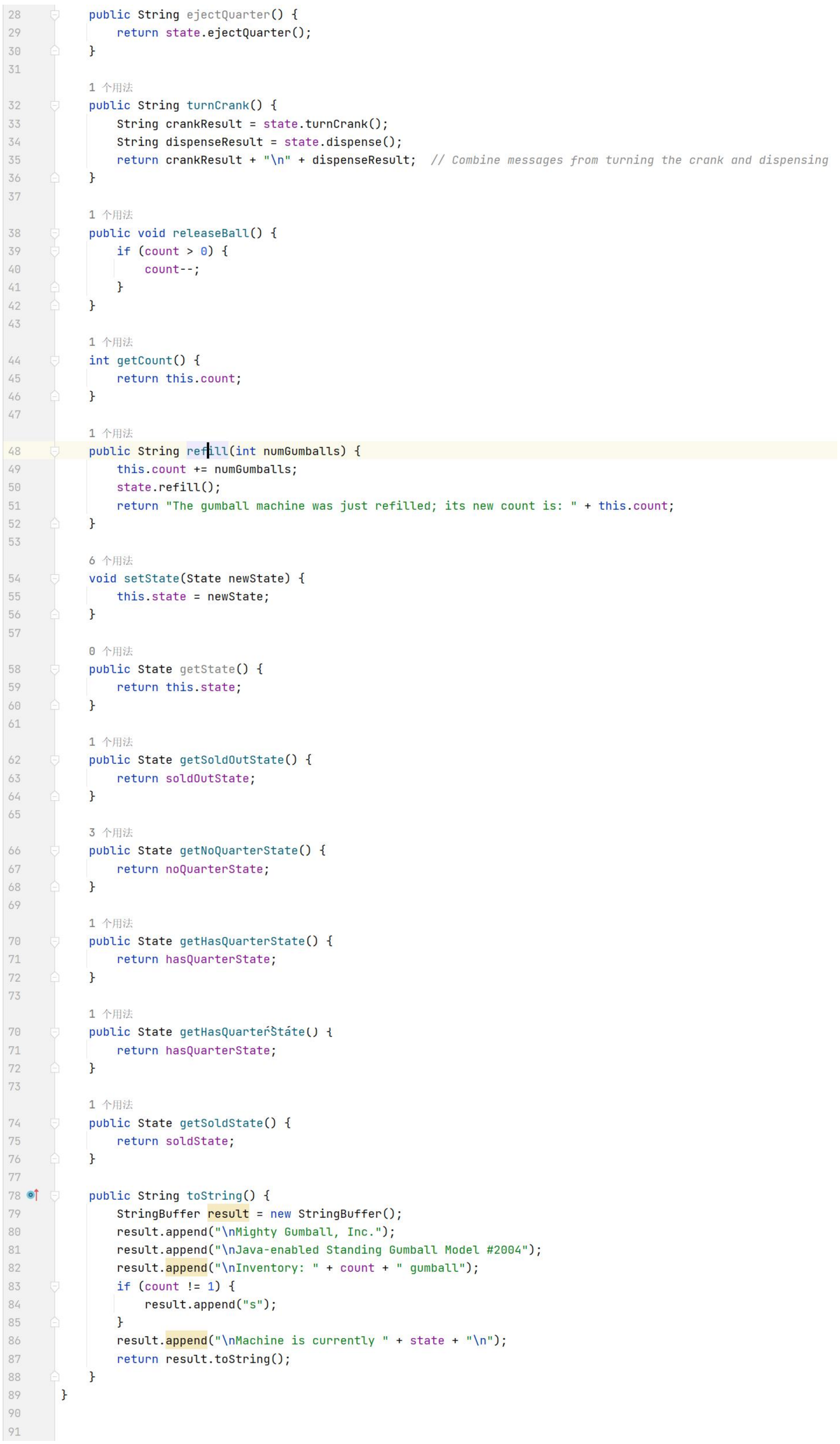


```java
public class NoQuarterState implements State {
    3 个用法
    GumballMachine gumballMachine;

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

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

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

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

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

    1 个用法
    public void refill() {
    }

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

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。

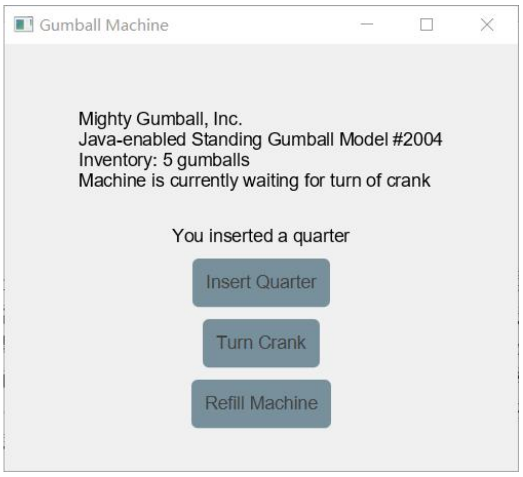


```java
public class GumballMachine {
    3 个用法
    State soldOutState;
    3 个用法
    State noQuarterState;
    2 个用法
    State hasQuarterState;
    2 个用法
    State soldState;
    10 个用法
    State state;
    8 个用法
    int count = 0;

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

    1 个用法
    public String insertQuarter() {
        return state.insertQuarter();
    }

    0 个用法
```



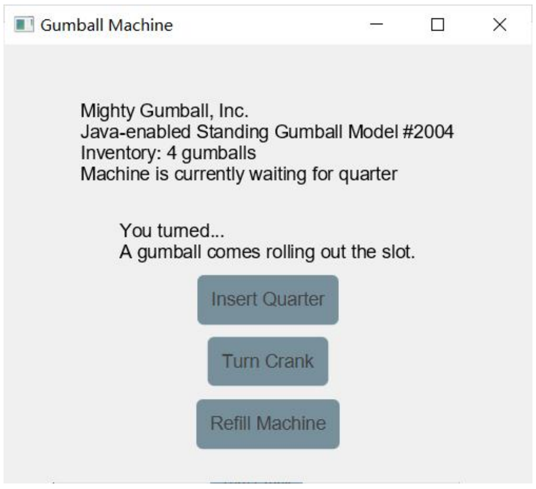
```
    public String ejectQuarter() {
        return state.ejectQuarter();
    }

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

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

    1 个用法
    int getCount() {
        return this.count;
    }

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

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

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

    1 个用法
    public State getSoldOutState() {
        return soldOutState;
    }

    3 个用法
    public State getNoQuarterState() {
        return noQuarterState;
    }

    1 个用法
    public State getHasQuarterState() {
        return hasQuarterState;
    }

    1 个用法
    public State getHasQuarterState() {
        return hasQuarterState;
    }

    1 个用法
    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();
    }
}
```

运行程序：



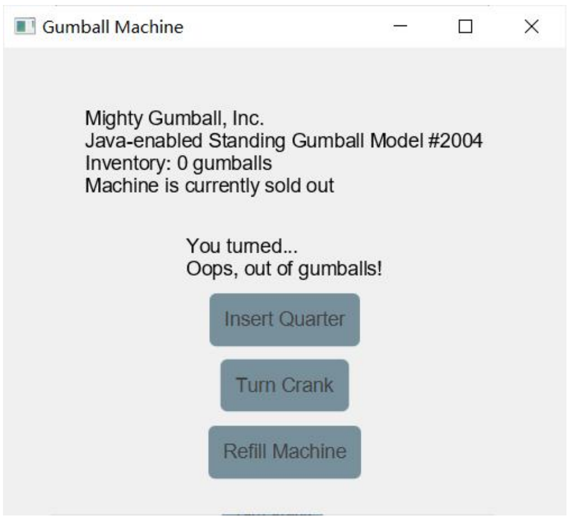
Gumball Machine

```
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 5 gumballs
Machine is currently waiting for turn of crank

You inserted a quarter
```



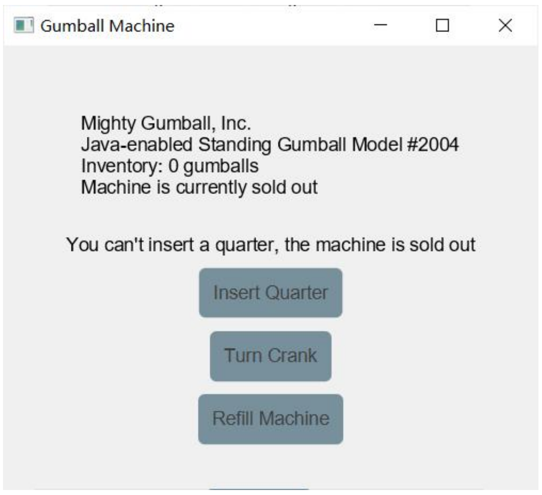
Gumball Machine

```
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 4 gumballs
Machine is currently waiting for quarter

You turned...
A gumball comes rolling out the slot.
```



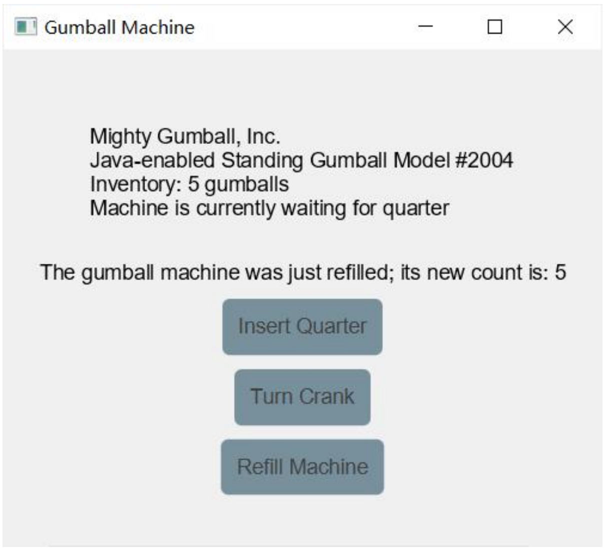
Gumball Machine

```
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 0 gumballs
Machine is currently sold out

You turned...
Oops, out of gumballs!
```


Gumball Machine

Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 0 gumballs
Machine is currently sold out

You can't insert a quarter, the machine is sold out


Gumball Machine

Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 5 gumballs
Machine is currently waiting for quarter

The gumball machine was just refilled; its new count is: 5

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) {
```

```java
        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**

```java
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;
```

```java
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);
    }
}
```