软件体系结构作业 16

姓　名：洪伟鑫

专　业：软件工程

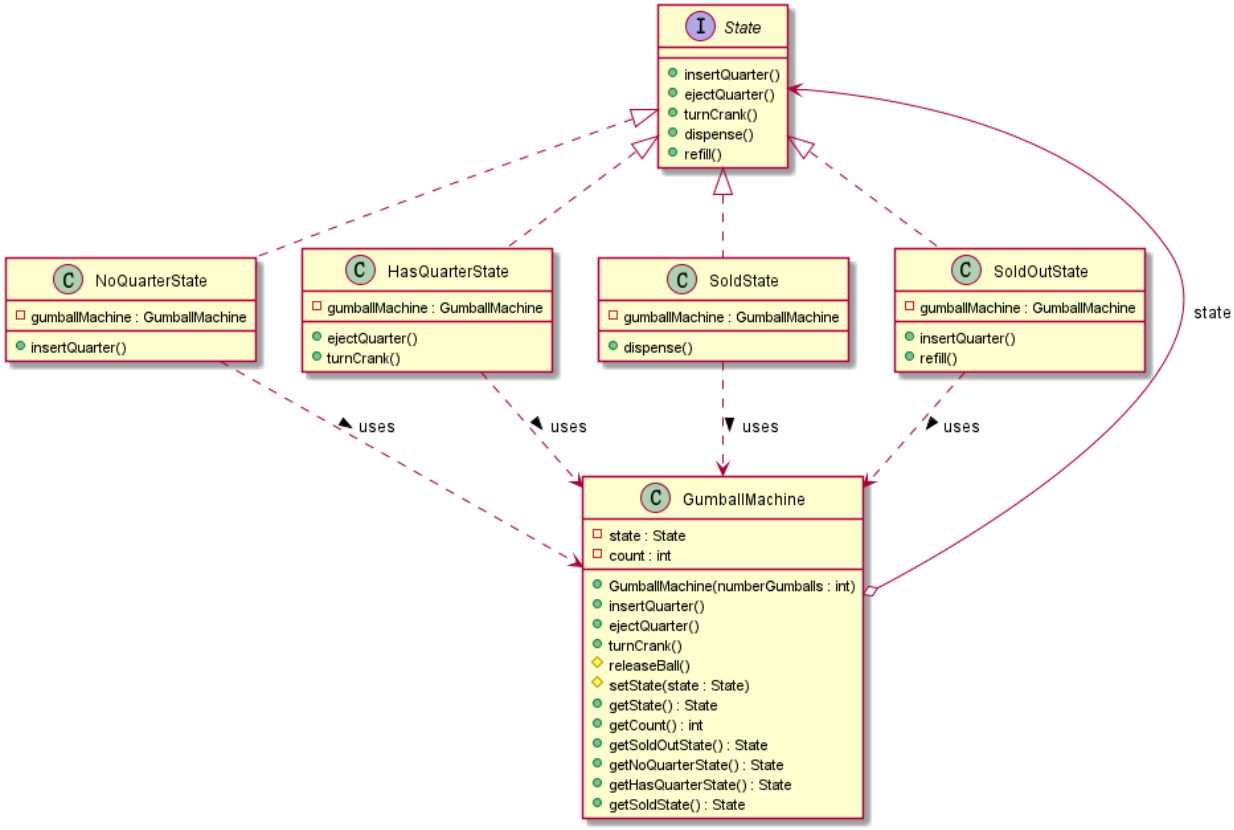
年　级：2022 级

学　号：37220222203612

2025 年 5 月 4 日

1.阅读 Gumballstate 源码并改写成你想的（GUI？）。

State 模式分析:



该扭蛋机项目通过状态模式（State Pattern）将扭蛋机的不同状态（如无硬币、有硬币、售出糖果、售罄）封装成独立的对象，从而实现了状态与行为的分离。

核心类 `GumballMachine.java` 作为 Context，负责维护当前状态以及所有具体状态对象的实例（ noQuarterState，hasQuarterState，soldState，soldOutState ）。它不直接处理用户的操作请求，而是将这些请求委托给当前的 state 对象。

所有具体状态类（如 `NoQuarterState.java`，`HasQuarterState.java` 等）均实现了统一的 `State.java` 接口，该接口定义了所有可能的操作。每个具体状态类根据自身状态实现了这些操作接口，并负责在执行操作后，通过调用 `GumballMachine.java` 的 `setState` 方法来改变扭蛋机的当前状态。这种设计使得 `GumballMachine.java` 类避免了复杂的条件判断逻辑，将状态相关的行为局部化到各个状态类中，使得系统更加灵活，易于理解和扩展。当扭蛋机的内部状态改变时，其行为也随之改变，这正是状态模式的核心思想。

GUI 设计：

GUI 使用 Java Swing 构建，主窗口采用 `BorderLayout` 布局，顶部放置了显示当前状态和剩余扭蛋数量的 `JLabel`，中部是一个带滚动条的 `JTextArea` 用于显示操作反馈和机器状态信息，底部则通过 `FlowLayout` 居中排列了“投入硬币”、“退回硬币”、“转动曲柄”和数量输入框。所有按钮共享同一个 `ActionListener`，在 `actionPerformed` 方法中根据事件来源调用 `GumballMachine` 相应的方法来触发状态转换，并通过调用 `updateGUI` 方法实时更新界面上的状态和数量标签，从而直观地展示了状态模式下扭蛋机随操作变化的行为。

```
public class GumballGUI extends JFrame implements ActionListener {

    private GumballMachine gumballMachine;

    // GUI 组件
    private JLabel stateLabel; // 状态标签
    private JLabel countLabel; // 数量标签
    private JTextArea messageArea; // 消息显示区域
    private JButton insertQuarterButton; // 投币按钮
    private JButton ejectQuarterButton; // 退币按钮
    private JButton turnCrankButton; // 转动曲柄按钮
    private JButton refillButton; // (可选) 补充按钮
    private JTextField refillAmountField; // (可选) 补充数量输入框

    public GumballGUI(int initialCount) {
        gumballMachine = new GumballMachine(initialCount);

        setTitle(title:"扭蛋机 - 状态模式"); // 设置窗口标题
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 设置默认关闭操作
        setLayout(new BorderLayout(hgap:10, vgap:10)); // 使用 BorderLayout 布局，组件间距为 10

        // --- 顶部面板，用于显示状态和数量 ---
        JPanel infoPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap:20, vgap:5)); // 居中对齐，水平
        stateLabel = new JLabel(text:"状态: 初始化中...");
        countLabel = new JLabel("扭蛋数量: " + initialCount);
        infoPanel.add(stateLabel);
        infoPanel.add(countLabel);
        add(infoPanel, BorderLayout.NORTH); // 添加到窗口北部
```

```
// --- 中部面板，用于显示消息 ---
messageArea = new JTextArea(rows:10, columns:30); // 10 行 30 列
messageArea.setEditable(b:false); // 设置为不可编辑
JScrollPane scrollPane = new JScrollPane(messageArea); // 添加滚动条
add(scrollPane, BorderLayout.CENTER); // 添加到窗口中部

// --- 底部面板，用于放置按钮 ---
JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap:10, vgap:5)); // 居中
insertQuarterButton = new JButton(text:"投入硬币");
ejectQuarterButton = new JButton(text:"退回硬币");
turnCrankButton = new JButton(text:"转动曲柄");

insertQuarterButton.addActionListener(this); // 注册监听器
ejectQuarterButton.addActionListener(this); // 注册监听器
turnCrankButton.addActionListener(this); // 注册监听器

buttonPanel.add(insertQuarterButton);
buttonPanel.add(ejectQuarterButton);
buttonPanel.add(turnCrankButton);

refillAmountField = new JTextField(text:"5", columns:3); // 默认补充 5 个，宽度为 3 个字符
refillButton = new JButton(text:"补充");
refillButton.addActionListener(this); // 注册监听器
buttonPanel.add(new JLabel(text:" 数量:")); // 数量标签
buttonPanel.add(refillAmountField);
buttonPanel.add(refillButton);
```

```
    add(buttonPanel, BorderLayout.SOUTH); // 添加到窗口南部

    // 初始化状态更新
    updateGUI();

    pack(); // 根据组件调整窗口大小
    setLocationRelativeTo(c:null); // 窗口居中显示
    setVisible(b:true); // 设置窗口可见
}
```

```java
@Override
public void actionPerformed(ActionEvent e) {
    String actionMessage = ""; // 操作消息
    if (e.getSource() == insertQuarterButton) {
        actionMessage = "--- 操作: 投入硬币 ---";
        gumballMachine.insertQuarter();
    } else if (e.getSource() == ejectQuarterButton) {
        actionMessage = "--- 操作: 退回硬币 ---";
        gumballMachine.ejectQuarter();
    } else if (e.getSource() == turnCrankButton) {
        actionMessage = "--- 操作: 转动曲柄 ---";
        gumballMachine.turnCrank();
    } else if (e.getSource() == refillButton) {
         try {
            int amount = Integer.parseInt(refillAmountField.getText());
            if (amount > 0) {
                actionMessage = "--- 操作: 补充 (" + amount + ") ---";
                gumballMachine.refill(amount);
            } else {
                actionMessage = "--- 补充数量必须为正数 ---";
            }
        } catch (NumberFormatException ex) {
            actionMessage = "--- 无效的补充数量 ---";
        }
    }
```

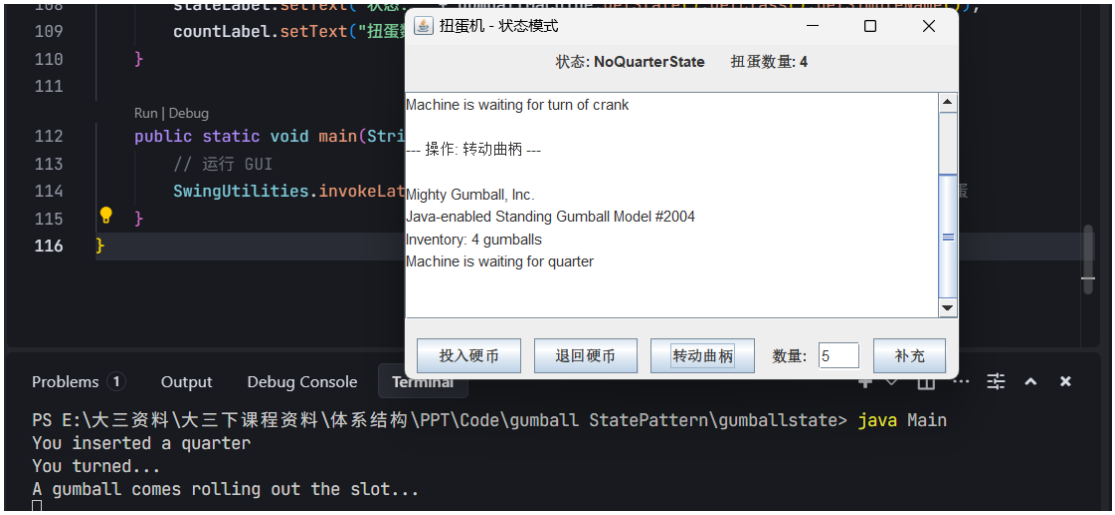
```java
private void updateGUI() {
    // 根据 GumballMachine 的当前状态和数量更新标签
    stateLabel.setText("状态: " + gumballMachine.getState().getClass().getSimpleName());
    countLabel.setText("扭蛋数量: " + gumballMachine.getCount());
}

Run | Debug
public static void main(String[] args) {
    // 运行 GUI
    SwingUtilities.invokeLater(() -> new GumballGUI(initialCount:5)); // 初始有 5 个扭蛋
}
```

运行截图：

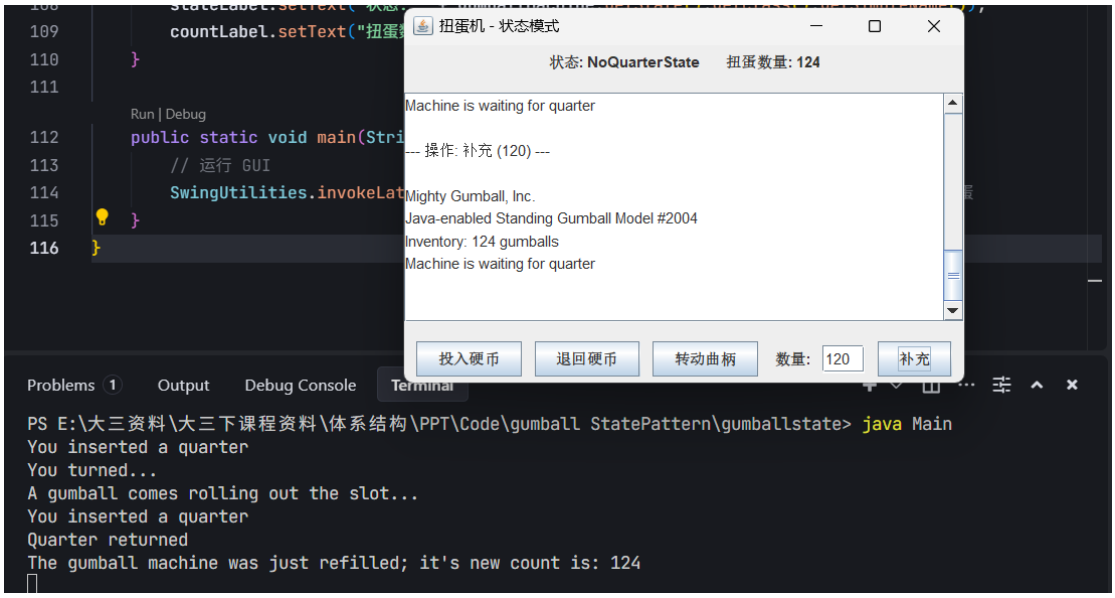
初始状态

投入一枚硬币



转动曲柄

投入后再退回



补充扭蛋币