软件体系结构作业 19

姓　名　：　洪伟鑫

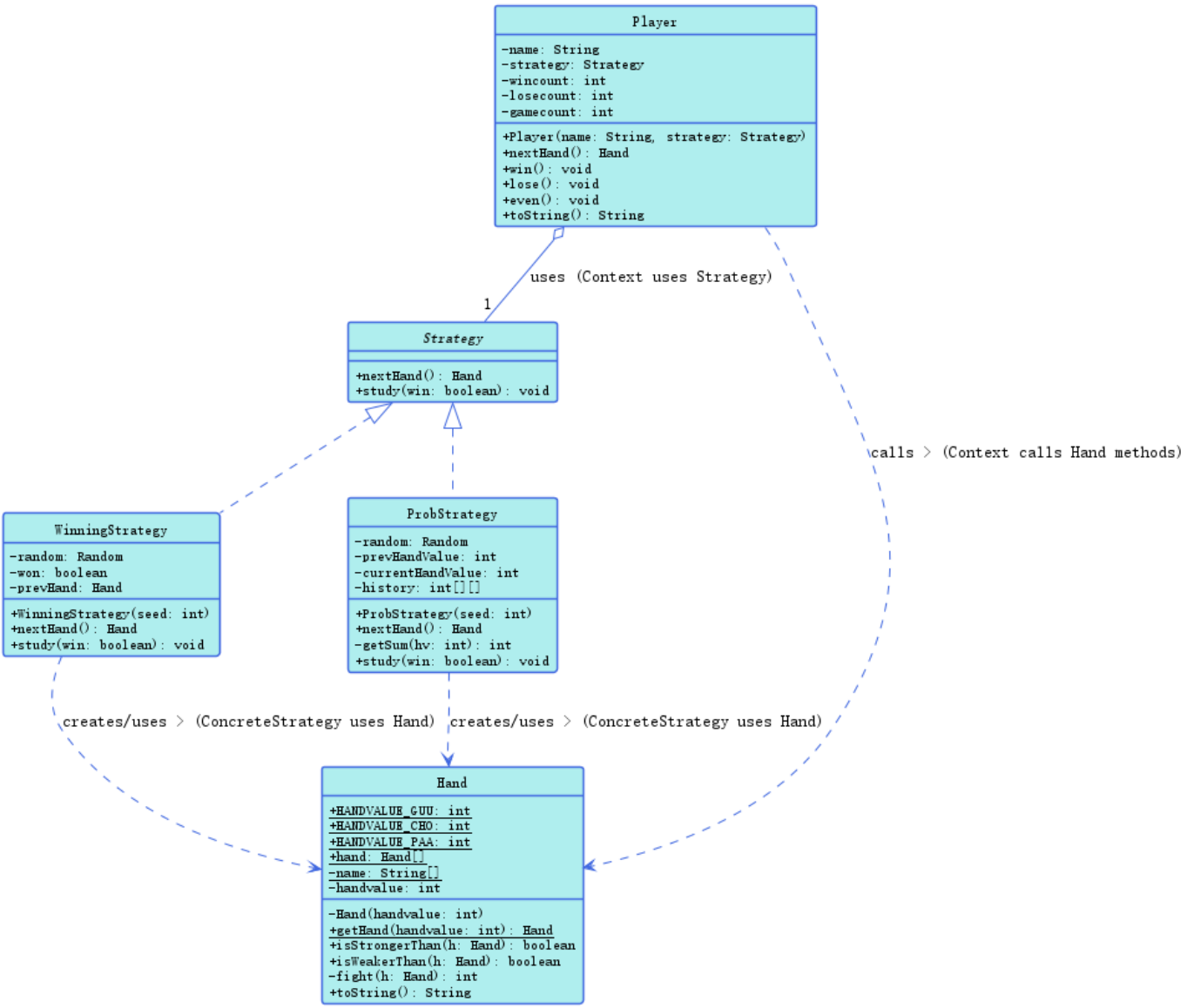
专　业　：　软件工程

年　级　：　2022 级

学　号　：　37220222203612

2025 年 5 月 1 7 日

1、用 GUI 改写策略模式的例子。



策略（Strategy）设计模式的核心结构，应用于一个石头剪刀布游戏的场景。

- 图中定义了一个 Strategy 接口，它规定了策略类必须实现的行为，即选择下一个手势（ nextHand ）和根据游戏结果进行学习（ study ）。
- WinningStrategy 和 ProbStrategy 是两个具体的策略实现类。
- Hand 类封装了游戏中的手势（如石头、剪刀、布）及其相互之间的克制关系。
- Player 类作为上下文（Context），持有一个 Strategy 对象的引用，并通过该策略来决定其出拳行为。

```java
// UI 组件
private JButton rockButton, paperButton, scissorsButton;
private JLabel playerChoiceLabel, computerChoiceLabel, resultLabel, scoreLabel;

// 游戏逻辑相关
private Player computerPlayer;
private Strategy computerStrategy;

// 记录分数
private int humanWins = 0;
private int computerWins = 0;
private int draws = 0;

// 手势名称 (与 Hand.java 中的 name 数组对应)
// Hand.HANDVALUE_GUU = 0;  // 石头
// Hand.HANDVALUE_CHO = 1;  // 剪刀
// Hand.HANDVALUE_PAA = 2;  // 布
// Hand.name = {"石头", "剪刀", "布"}
// 注意：为了确保一致性，这里直接使用 Hand 类中的常量和方法获取名称
private final String[] handNames = {"石头", "剪刀", "布"}; // 手势名称数组

// 定义一些颜色
private final Color bgColor = new Color(240, 240, 245); // 淡雅的背景色
private final Color panelBgColor = new Color(220, 220, 225); // 面板背景色
private final Color buttonTextColor = Color.WHITE;
private final Color rockButtonColor = new Color(255, 102, 102); // 红色系 - 石头
private final Color scissorsButtonColor = new Color(102, 178, 255); // 蓝色系 - 剪刀
private final Color paperButtonColor = new Color(102, 204, 102); // 绿色系 - 布
private final Color textColor = new Color(50, 50, 50); // 深灰色文本
private final Color winColor = new Color(0, 153, 51);    // 绿色 - 赢
private final Color loseColor = new Color(204, 0, 0);    // 红色 - 输
private final Color drawColor = new Color(0, 102, 204); // 蓝色 - 平局
```

导入相关的包，定义 GUI 要用到的各种颜色

```java
public GameGUI() {
    setTitle("石头剪刀布游戏-Strategy");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout(10, 10)); // 主布局，组件间距10像素
    getContentPane().setBackground(bgColor); // 设置JFrame的contentPane背景色

    // 初始化计算机玩家和策略
    // 使用 ProbStrategy，它能从游戏结果中学习
    computerStrategy = new ProbStrategy((int) System.currentTimeMillis()); // 使用当前时间作为随机种子
    computerPlayer = new Player("电脑", computerStrategy);

    // --- UI 设置 ---

    // 顶部面板：玩家操作按钮
    JPanel actionPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 15)); // 增大间距
    actionPanel.setBorder(BorderFactory.createEmptyBorder(15, 10, 10, 10)); // 调整边距
    actionPanel.setBackground(panelBgColor); // 设置面板背景色

    // 根据 Hand.java 中的定义创建按钮
    rockButton = new JButton(handNames[Hand.HANDVALUE_GUU]);     // 石头
    scissorsButton = new JButton(handNames[Hand.HANDVALUE_CHO]); // 剪刀
    paperButton = new JButton(handNames[Hand.HANDVALUE_PAA]);    // 布

    Font buttonFont = new Font("SimSun", Font.BOLD, 16); // 按钮字体加粗，稍大
    rockButton.setFont(buttonFont);
    scissorsButton.setFont(buttonFont);
    paperButton.setFont(buttonFont);

    rockButton.setBackground(rockButtonColor);
    scissorsButton.setBackground(scissorsButtonColor);
    paperButton.setBackground(paperButtonColor);

    rockButton.setForeground(buttonTextColor);
    scissorsButton.setForeground(buttonTextColor);
    paperButton.setForeground(buttonTextColor);



    // 设置ActionCommand为手势对应的值，方便处理
    rockButton.setActionCommand(String.valueOf(Hand.HANDVALUE_GUU));
    scissorsButton.setActionCommand(String.valueOf(Hand.HANDVALUE_CHO));
    paperButton.setActionCommand(String.valueOf(Hand.HANDVALUE_PAA));

    rockButton.addActionListener(this);
    scissorsButton.addActionListener(this);
    paperButton.addActionListener(this);
```

```
    // 设置ActionCommand为手势对应的值，方便处理
    rockButton.setActionCommand(String.valueOf(Hand.HANDVALUE_GUU));
    scissorsButton.setActionCommand(String.valueOf(Hand.HANDVALUE_CHO));
    paperButton.setActionCommand(String.valueOf(Hand.HANDVALUE_PAA));

    rockButton.addActionListener(this);
    scissorsButton.addActionListener(this);
    paperButton.addActionListener(this);

    JLabel promptLabel = new JLabel("请出拳：");
    promptLabel.setFont(new Font("SimSun", Font.BOLD, 16)); // 提示字体加粗
    promptLabel.setForeground(textColor);
    actionPanel.add(promptLabel);
    actionPanel.add(rockButton);
    actionPanel.add(scissorsButton);
    actionPanel.add(paperButton);


    // 中部面板：显示选择、结果和分数
    JPanel resultsPanel = new JPanel();
    resultsPanel.setLayout(new BoxLayout(resultsPanel, BoxLayout.Y_AXIS)); // 垂直排列
    resultsPanel.setBorder(BorderFactory.createEmptyBorder(15, 25, 25, 25)); // 调整内边距
    resultsPanel.setBackground(panelBgColor); // 设置面板背景色

    playerChoiceLabel = new JLabel("你的选择：-");
    computerChoiceLabel = new JLabel("电脑的选择：-");
    resultLabel = new JLabel("结果：-");
    scoreLabel = new JLabel("比分：你 0 - 电脑 0 (平局：0)");

    Font labelFont = new Font("SimSun", Font.PLAIN, 16); // 标签字体稍大
    playerChoiceLabel.setFont(labelFont);
    computerChoiceLabel.setFont(labelFont);
    resultLabel.setFont(new Font("SimSun", Font.BOLD, 18)); // 结果加粗，更大
    scoreLabel.setFont(new Font("SimSun", Font.PLAIN, 15));

    playerChoiceLabel.setForeground(textColor);
    computerChoiceLabel.setForeground(textColor);
    resultLabel.setForeground(textColor); // 初始颜色
    scoreLabel.setForeground(textColor);

    // 左对齐组件
    playerChoiceLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
    computerChoiceLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
    resultLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
    scoreLabel.setAlignmentX(Component.LEFT_ALIGNMENT);

    resultsPanel.add(playerChoiceLabel);
    resultsPanel.add(Box.createRigidArea(new Dimension(0, 10))); // 垂直间距
    resultsPanel.add(computerChoiceLabel);
    resultsPanel.add(Box.createRigidArea(new Dimension(0, 10)));
    resultsPanel.add(resultLabel);
    resultsPanel.add(Box.createRigidArea(new Dimension(0, 20))); // 较大间距
    resultsPanel.add(scoreLabel);

    // 将面板添加到Frame
    add(actionPanel, BorderLayout.NORTH);
    add(resultsPanel, BorderLayout.CENTER);

    pack(); // 根据组件自动调整窗口大小
    setMinimumSize(new Dimension(420, 300)); // 调整最小尺寸以适应新样式
    setLocationRelativeTo(null); // 窗口居中显示
}
```

这部分代码是 GameGUI 类的构造函数，它负责初始化整个游戏的用户界面。
首先，它设置了窗口的基本属性，如标题、关闭操作、布局和背景色。接着，它初始化了计算机玩家，为其分配了一个基于概率学习的策略（ ProbStrategy ），并使用当前时间作为随机种子。
然后，它开始构建界面元素：创建了一个顶部面板用于放置玩家的操作按钮（石头、剪刀、布），并为这些按钮设置了文字、字体、颜色、以及点击时触发的动作指令；同时，它还创建了一个中部面板，用于显示玩家的选择、电脑的选择、单局结果以及累计得分，并对这些显示标签的字体和颜色进行了设定。
最后，它将这两个主要面板添加到窗口中，并调用 pack() 方法自动调整窗口大小以适应内容，设置了窗口的最小尺寸，并让窗口在屏幕上居中显示。

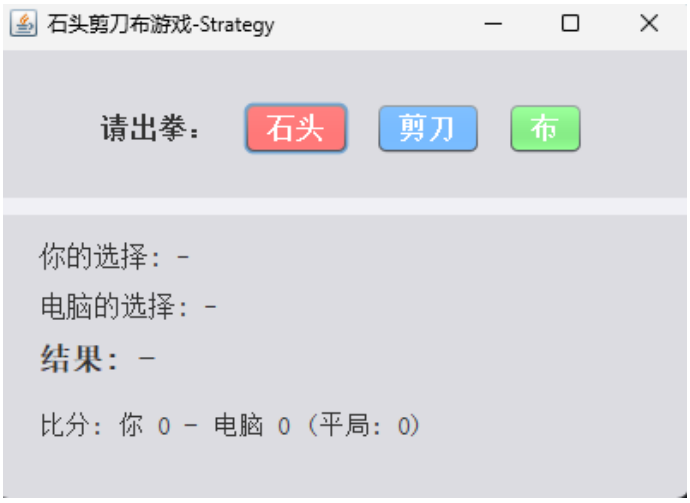
```java
@Override
public void actionPerformed(ActionEvent e) {
    // 获取玩家选择的手势值
    int playerHandValue = Integer.parseInt(e.getActionCommand());
    Hand humanHand = Hand.getHand(playerHandValue);

    // 计算机出拳
    Hand computerHand = computerPlayer.nextHand();

    // 更新界面显示
    playerChoiceLabel.setText("你的选择: " + humanHand.toString());
    computerChoiceLabel.setText("电脑的选择: " + computerHand.toString());

    // 判断胜负并更新结果
    if (humanHand.isStrongerThan(computerHand)) {
        resultLabel.setText("结果: 你赢了!");
        resultLabel.setForeground(winColor); // 设置赢的颜色
        humanWins++;
        computerPlayer.lose(); // 计算机输了，策略学习
    } else if (humanHand.isWeakerThan(computerHand)) {
        resultLabel.setText("结果: 电脑赢了!");
        resultLabel.setForeground(loseColor); // 设置输的颜色
        computerWins++;
        computerPlayer.win(); // 计算机赢了，策略学习
    } else {
        resultLabel.setText("结果: 平局!");
        resultLabel.setForeground(drawColor); // 设置平局的颜色
        draws++;
        computerPlayer.even(); // 平局，策略学习
    }

    updateScoreLabel();
}

private void updateScoreLabel() {
    scoreLabel.setText(String.format("比分: 你 %d - 电脑 %d (平局: %d)", humanWins, computerWins, draws));
}
```

这段代码主要处理用户在石头剪刀布游戏中点击按钮后的交互逻辑以及更新得分显示。
当用户点击代表石头、剪刀或布的按钮时， actionPerformed 方法会被调用。
它首先根据按钮的 ActionCommand 获取用户选择的手势，然后调用 computerPlayer.nextHand() 来获取计算机根据其当前策略所出的手势。
接着，它会更新界面上显示用户选择和计算机选择的标签。之后，通过比较用户和计算机的手势，判断本局的胜负平关系，并相应地更新结果标签的文本和颜色（赢为绿色，输为红色，平局为蓝色）。
同时，它会累加用户赢、计算机赢或平局的次数，并调用计算机玩家的 win() 、 lose() 或 even() 方法，这些方法内部会调用策略对象的 study() 方法，以便策略能够根据游戏结果进行学习和调整。最后，它调用 updateScoreLabel 方法。 updateScoreLabel 方法则负责将最新的得分情况（包括用户赢的次数、计算机赢的次数以及平局次数）格式化并显示在界面上的得分标签中。

```java
public static void main(String[] args) {
    // 确保GUI操作在事件调度线程中执行
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            try {
                for (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
                    if ("Nimbus".equals(info.getName())) {
                        UIManager.setLookAndFeel(info.getClassName());
                        break;
                    }
                }
            } catch (Exception e) {
                try {
                    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
            new GameGUI().setVisible(true);
        }
    });
}
```

Ｍａｉｎ函数创建 GameGUI 类的一个新实例，并通过调用 setVisible(true) 方法来显示游戏窗口，从而启动整个应用程序。

运行结果：

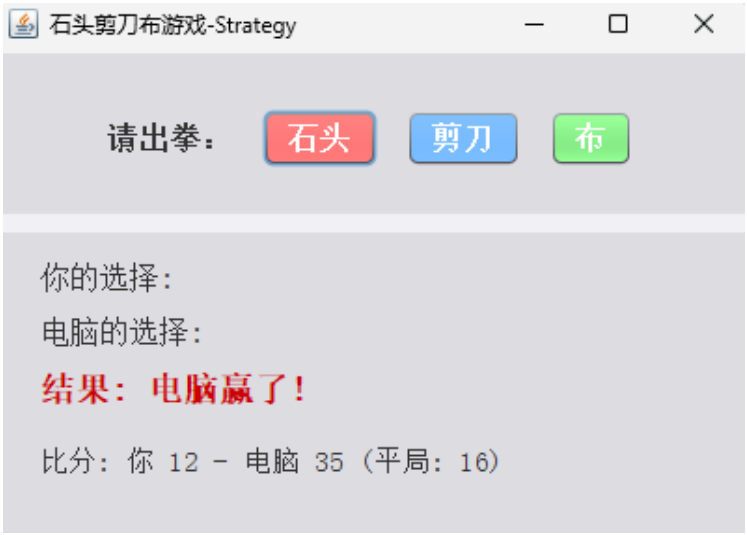


初始界面

电脑获胜

平局

玩家获胜



经过若干次出拳后的比分