



Code：

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.Random;  
  
// 手势类  
class Hand {  
 public static final int *HANDVALUE\_GUU* = 0;  
 public static final int *HANDVALUE\_CHO* = 1;  
 public static final int *HANDVALUE\_PAA* = 2;  
 public static final Hand[] *hand* = {  
 new Hand(*HANDVALUE\_GUU*),  
 new Hand(*HANDVALUE\_CHO*),  
 new Hand(*HANDVALUE\_PAA*)  
 };  
 private static final String[] *name* = {"石头", "剪刀", "布"};  
 private int handvalue;  
  
 private Hand(int handvalue) {  
 this.handvalue = handvalue;  
 }  
  
 public static Hand getHand(int handvalue) {  
 return *hand*[handvalue];  
 }  
  
 public boolean isStrongerThan(Hand h) {  
 return fight(h) == 1;  
 }  
  
 public boolean isWeakerThan(Hand h) {  
 return fight(h) == -1;  
 }  
  
 private int fight(Hand h) {  
 if (this == h) return 0;  
 else if ((this.handvalue + 1) % 3 == h.handvalue) return 1;  
 else return -1;  
 }  
  
 public String toString() {  
 return *name*[handvalue];  
 }  
}  
  
// 策略接口  
interface Strategy {  
 Hand nextHand();  
 void study(boolean win);  
}  
  
// 获胜策略  
class WinningStrategy implements Strategy {  
 private Random random;  
 private boolean won;  
 private Hand prevHand;  
  
 public WinningStrategy(int seed) {  
 random = new Random(seed);  
 }  
  
 public Hand nextHand() {  
 if (!won) {  
 prevHand = Hand.*getHand*(random.nextInt(3));  
 }  
 return prevHand;  
 }  
  
 public void study(boolean win) {  
 won = win;  
 }  
}  
  
// 概率策略  
class ProbStrategy implements Strategy {  
 private Random random;  
 private int prevHandValue = 0;  
 private int currentHandValue = 0;  
 private int[][] history = {  
 {1, 1, 1},  
 {1, 1, 1},  
 {1, 1, 1}  
 };  
  
 public ProbStrategy(int seed) {  
 random = new Random(seed);  
 }  
  
 public Hand nextHand() {  
 int bet = random.nextInt(getSum(currentHandValue));  
 int handvalue = 0;  
 if (bet < history[currentHandValue][0]) {  
 handvalue = 0;  
 } else if (bet < history[currentHandValue][0] + history[currentHandValue][1]) {  
 handvalue = 1;  
 } else {  
 handvalue = 2;  
 }  
 prevHandValue = handvalue;  
 currentHandValue = handvalue;  
 return Hand.*getHand*(handvalue);  
 }  
  
 private int getSum(int hv) {  
 int sum = 0;  
 for (int i = 0; i < 3; i++) {  
 sum += history[hv][i];  
 }  
 return sum;  
 }  
  
 public void study(boolean win) {  
 if (win) {  
 history[prevHandValue][currentHandValue]++;  
 } else {  
 history[prevHandValue][(currentHandValue + 1) % 3]++;  
 history[prevHandValue][(currentHandValue + 2) % 3]++;  
 }  
 }  
}  
  
// 玩家类  
class Player {  
 private String name;  
 private Strategy strategy;  
 private int winCount;  
 private int loseCount;  
 private int gameCount;  
  
 public Player(String name, Strategy strategy) {  
 this.name = name;  
 this.strategy = strategy;  
 }  
  
 public Hand nextHand() {  
 return strategy.nextHand();  
 }  
  
 public void win() {  
 strategy.study(true);  
 winCount++;  
 gameCount++;  
 }  
  
 public void lose() {  
 strategy.study(false);  
 loseCount++;  
 gameCount++;  
 }  
  
 public void even() {  
 gameCount++;  
 }  
  
 public String toString() {  
 return "[" + name + ": " + gameCount + " games, " + winCount + " win, " + loseCount + " lose]";  
 }  
}  
  
// 主GUI窗口  
class StrategyPatternGUI extends JFrame {  
 private JTextArea resultArea;  
 private JButton startButton;  
  
 public StrategyPatternGUI() {  
 setTitle("策略模式 - 剪刀石头布游戏");  
 setSize(600, 400);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(new BorderLayout());  
  
 resultArea = new JTextArea();  
 resultArea.setEditable(false);  
 add(new JScrollPane(resultArea), BorderLayout.*CENTER*);  
  
 startButton = new JButton("开始游戏（100轮）");  
 startButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 playGame();  
 }  
 });  
 add(startButton, BorderLayout.*SOUTH*);  
 }  
  
 private void playGame() {  
 Player player1 = new Player("Taro", new WinningStrategy(314));  
 Player player2 = new Player("Hana", new ProbStrategy(15));  
  
 resultArea.setText(""); // 清空之前的结果  
 for (int i = 0; i < 100; i++) {  
 Hand hand1 = player1.nextHand();  
 Hand hand2 = player2.nextHand();  
  
 String roundResult;  
 if (hand1.isStrongerThan(hand2)) {  
 roundResult = "Round " + (i + 1) + ": " + player1 + " 赢（" + hand1 + " vs " + hand2 + "）";  
 player1.win();  
 player2.lose();  
 } else if (hand2.isStrongerThan(hand1)) {  
 roundResult = "Round " + (i + 1) + ": " + player2 + " 赢（" + hand2 + " vs " + hand1 + "）";  
 player2.win();  
 player1.lose();  
 } else {  
 roundResult = "Round " + (i + 1) + ": 平局（" + hand1 + " vs " + hand2 + "）";  
 player1.even();  
 player2.even();  
 }  
 resultArea.append(roundResult + "\n");  
 }  
  
 resultArea.append("\nTotal Result:\n");  
 resultArea.append(player1.toString() + "\n");  
 resultArea.append(player2.toString() + "\n");  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> new StrategyPatternGUI().setVisible(true));  
 }  
}