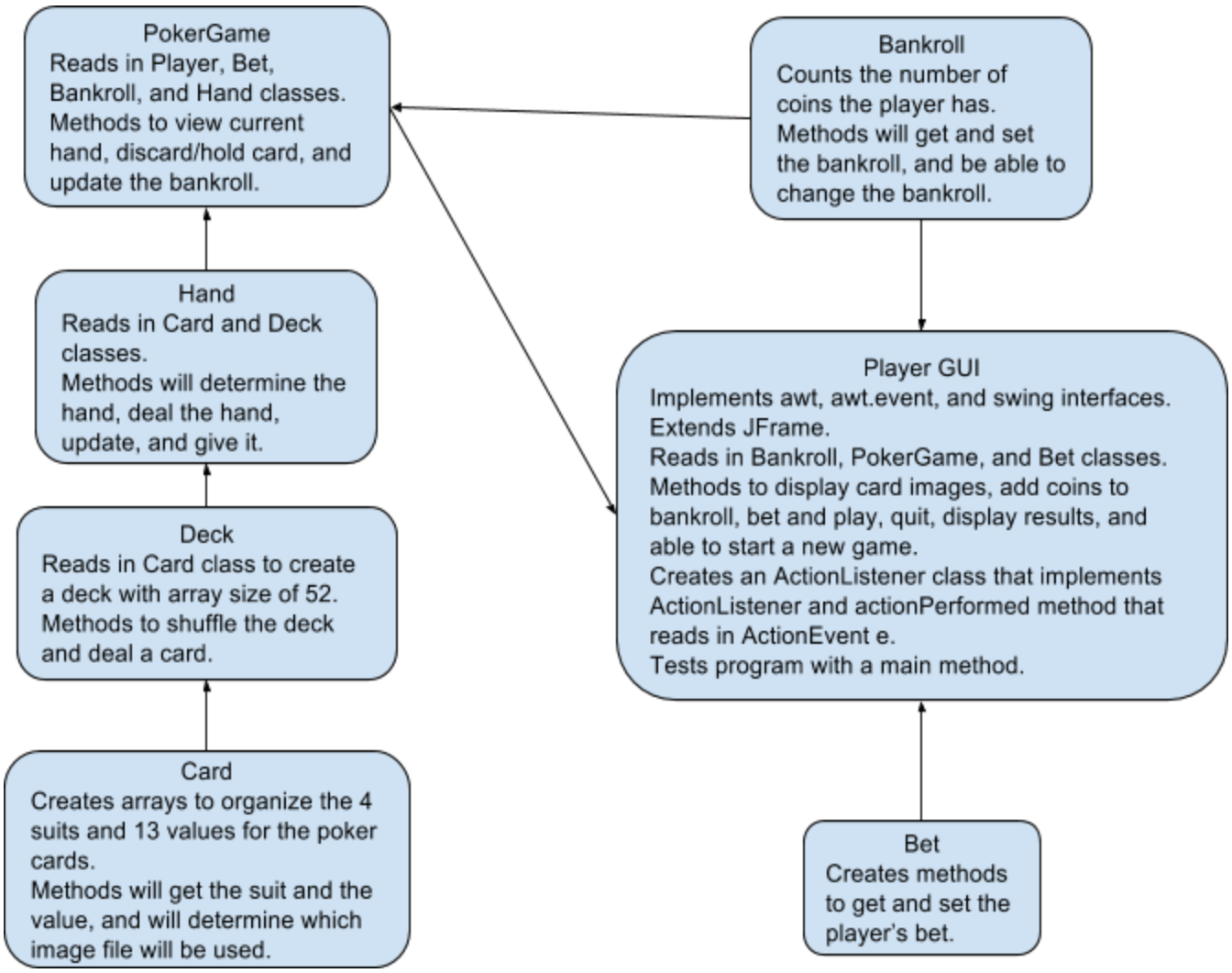
Poker Game Project with Java Language

Class Diagram:



Program Code:

//New Player Class - Image files are stored in src/ folder

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.util.logging.*;
import javax.swing.*;
public class Player extends JFrame{

	private JLabel resultLabel;
	private JLabel[] cardLabel;
	private JButton[] holdButton;
	private JButton add1Button;
	private JButton add5Button;
	private JLabel bankrollLabel;
	private JButton quitButton;
	private JButton dealButton;
	private JButton startNewButton;
	private JButton[] betAndPlayButton;
	private Bankroll bankroll;
	private PokerGame pokerGame;
	private Bet bet;
	private JMenuBar menuBar = new JMenuBar();
	private JMenu menu = new JMenu("Menu");
	private JMenuItem check = new JMenuItem("Check coins");
	private JMenuItem add = new JMenuItem("Add coins");
	private JMenuItem reset = new JMenuItem("Start New Game");
	private JMenuItem quit = new JMenuItem("Quit Game");
	/*private JMenu bonus = new JMenu("Bonus");
	private JMenuItem secret = new JMenuItem("");
	private JMenuItem HowToPlay = new JMenuItem("How to play");
	private JMenuItem MySon = new JMenuItem("Funny Picture");*/

	//Constructor
	public Player() {
		super("Video Poker");
		bet = new Bet();
		bankroll = new Bankroll();
		setBounds(0, 0, 800, 800);

		//Menu Bar
		menuBar.add(menu);
		menu.add(check);
		menu.add(add);
		menu.add(reset);
		menu.add(quit);
		setJMenuBar(menuBar);
```

```
check.addActionListener(new ButtonListener());
add.addActionListener(new ButtonListener());
reset.addActionListener(new ButtonListener());
quit.addActionListener(new ButtonListener());
/*menuBar.add(bonus);
bonus.add(HowToPlay);
bonus.add(MySon);
bonus.add(secret);
secret.addActionListener(new ButtonListener());
HowToPlay.addActionListener(new ButtonListener());
MySon.addActionListener(new ButtonListener());*/

//Label for results
resultLabel = new JLabel();
resultLabel.setFont(new Font("Arial", Font.BOLD, 18));
resultLabel.setText("Video Poker");

//The five card images
cardLabel = new JLabel[5];
String backCard = "src/br.gif";
for (int i = 0; i < 5; i++){
        cardLabel[i] = new JLabel(new ImageIcon(backCard));
}

holdButton = new JButton[5];
for (int i = 0; i < 5; i++){
        holdButton[i] = new JButton(""  + (i +  1));
        holdButton[i].setFont(new Font("Arial", Font.BOLD, 18));
        holdButton[i].setEnabled(false);
}

betAndPlayButton = new JButton[5];
for (int i = 0; i <  5; i++){
        betAndPlayButton[i] = new JButton("Bet " + (i + 1));
        betAndPlayButton[i].setEnabled(false);
        betAndPlayButton[i].setFont(new Font("Arial", Font.BOLD, 15));
}

dealButton = (new JButton("Deal"));
dealButton.setFont(new Font("Arial", Font.BOLD, 18));
dealButton.setEnabled(false);

quitButton  = new JButton("Quit Game");
quitButton.setFont(new Font("Arial", Font.BOLD, 15));

startNewButton  = new JButton("Start New Game");
startNewButton.setFont(new Font("Arial", Font.BOLD, 15));
```

```
bankrollLabel = new JLabel();
bankrollLabel.setFont(new Font("Arial", Font.BOLD, 24));
bankrollLabel.setText("Coins remaining: " + 0);

add1Button = new JButton("Add 1 Coin");
add5Button = new JButton("Add 5 Coins");
add1Button.setFont(new Font("Arial", Font.BOLD, 15));
add5Button.setFont(new Font("Arial", Font.BOLD, 15));

JPanel centerPanel = new JPanel(new GridLayout(4,5));

//five bet buttons
for (int i  = 0; i <  5; i++){
        centerPanel.add(betAndPlayButton[i]);
}

//display five card labels/images
for (int i =  0; i <  5; i++){
        centerPanel.add(cardLabel[i]);
}

//add five hold buttons
for (int i = 0; i < 5; i++){
        centerPanel.add(holdButton[i]);
}

centerPanel.add(add1Button);
centerPanel.add(add5Button);
centerPanel.add(dealButton);
centerPanel.add(quitButton);
centerPanel.add(startNewButton);

add(resultLabel, BorderLayout.NORTH);//top text
add(bankrollLabel, BorderLayout.SOUTH);//bottom text
add(centerPanel, BorderLayout.CENTER);//centered buttons

add1Button.addActionListener(new ButtonListener());
add5Button.addActionListener(new ButtonListener());
dealButton.addActionListener(new ButtonListener());
quitButton.addActionListener(new ButtonListener());
startNewButton.addActionListener(new ButtonListener());

for(int i = 0; i < 5; i++){
        betAndPlayButton[i].addActionListener(new ButtonListener());
}

for(int i = 0; i < 5; i++){
        holdButton[i].addActionListener(new ButtonListener());
```

```
		}

		setResizable(false);
		setVisible(true);
	}

	//All images placed under src/ folder
	public void displayHand(Hand hand){
		String[] handString = hand.getHand();
		for(int i = 0; i < 5; i++){
			String name = "src/" + handString[i] + ".gif";
			cardLabel[i].setIcon(new ImageIcon(name));
		}
	}

	public void getDiscard(boolean[] holdCards){
		for(int i = 0; i < 5; i++){
			if(holdButton[i].isEnabled()){
				holdCards[i] = false;
			}
			else{
				holdCards[i] = true;
			}
		}
	}

	public void displayResults(int payoff, int winnings){
		String nameOfHand = "Lose";
		if(payoff == 250){
			nameOfHand = "Royal Flush";
		}
		else if(payoff == 50){
			nameOfHand = "Straight Flush";
		}
		else if(payoff == 25){
			nameOfHand = "Four of a Kind";
		}
		else if(payoff == 9){
			nameOfHand = "Full House";
		}
		else if(payoff == 6){
			nameOfHand = "Flush";
		}
		else if(payoff == 4){
			nameOfHand = "Straight";
		}
		else if(payoff == 3){
			nameOfHand = "Three of a Kind";
```

```
		}
		else if(payoff == 2){
			nameOfHand = "Two Pair";
		}
		else if(payoff == 1){
			nameOfHand = "Pair of Jacks or Better";
		}

		if(winnings > 0){
			resultLabel.setText("Winner: " + nameOfHand + " - pays " + winnings);
		}
		else{
			resultLabel.setText("You lost your bet of " + bet.getBet());
		}

		bankrollLabel.setText("Coins remaining: " + bankroll.getBankroll());
	}

	//Action Listener Class
	private class ButtonListener implements ActionListener{
		@Override
		public void actionPerformed(ActionEvent e) {
			//Adds more coins in coin balance
			if ((e.getSource() == add1Button) || (e.getSource() == add5Button)) {
				if (e.getSource() ==  add1Button){
					bankroll.alterBankroll(1);
				}
				else{
					bankroll.alterBankroll(5);
				}

				int br = bankroll.getBankroll();
				bankrollLabel.setText("Coins remaining: "  + br);
				for (int i = 0; i < 5; i++){
					if (br >= (i + 1)){
						betAndPlayButton[i].setEnabled(true);
					}
				}
				return;
			}

			//Quits the program
			if (e.getSource() == quitButton){
				int br = bankroll.getBankroll();
				System.exit(0);
			}

			//5 buttons for betting
```

```
for (int i = 0; i < 5; i++){
	if (e.getSource() == betAndPlayButton[i]) {
		bet = new Bet();
		bet.setBet(i + 1);
		resultLabel.setText("Bet is " + (i + 1));

		pokerGame = new PokerGame(bet, bankroll, Player.this);
		pokerGame.viewInitialHand();
		for(int j = 0; j < 5; j++) {
			holdButton[j].setText("" + (j + 1));
			holdButton[j].setEnabled(true);
		}

		// enable and disable other buttons
		add1Button.setEnabled(false);
		add5Button.setEnabled(false);
		quitButton.setEnabled(true);
		dealButton.setEnabled(true);
		startNewButton.setEnabled(true);
		for (int j = 0; j < 5; j++)
			betAndPlayButton[j].setEnabled(false);
		return;
	}
}

//5 buttons to hold cards
for (int i = 0; i < 5; i++){
	if (e.getSource() == holdButton[i]) {
		holdButton[i].setText("Hold");
		holdButton[i].setEnabled(false);
		return;
	}
}

//Deal button - Enables when player chooses a bet button.
if (e.getSource() == dealButton) {
	pokerGame.discardOrHoldCards();
	dealButton.setEnabled(false);
	for(int j = 0; j <  5; j++){
		holdButton[j].setEnabled(false);
	}
	for (int i = 0; i <  5; i++)
		if (bankroll.getBankroll()  >= (i + 1)){
			betAndPlayButton[i].setEnabled(true);
		}
	add1Button.setEnabled(true);
	add5Button.setEnabled(true);
	quitButton.setEnabled(true);
```

```
                }

                //Start button to start a new game. Resets cards and number of coins in balance.
                if(e.getSource() == startNewButton){
                        for(int i = 0; i < 5; i++){
                                String name = "src/br.gif";
                                cardLabel[i].setIcon(new ImageIcon(name));
                        }
                        resultLabel.setText("Video Poker");
                        bankroll = new Bankroll(0);
                        bankrollLabel.setText("Coins remaining: " + 0);
                        add1Button.setEnabled(true);
                        add5Button.setEnabled(true);
                        quitButton.setEnabled(true);
                        dealButton.setEnabled(false);
                        for(int j = 0; j < 5; j++)
                                betAndPlayButton[j].setEnabled(false);
                        for(int j = 0; j < 5; j++){
                                holdButton[j].setText("" + (j+1));
                                holdButton[j].setEnabled(false);
                        }

                        return;
                }

                //MenuBar items
                if(e.getSource() == check){
                        JOptionPane.showMessageDialog(null, "Your current bankroll: " +
bankroll.getBankroll());
                }
                else if(e.getSource() == add){
                        int amount = Integer.parseInt(JOptionPane.showInputDialog("Enter
amount of coins"));
                        bankroll.alterBankroll(amount);
                        int br = bankroll.getBankroll();
                        bankrollLabel.setText("Coins remaining: " + br);
                        for (int i = 0; i < 5; i++){
                                if (br >= (i + 1)){
                                        betAndPlayButton[i].setEnabled(true);
                                }
                        }
                }
                else if(e.getSource() == reset){
                        for(int i = 0; i < 5; i++){
                                String name = "src/br.gif";
                                cardLabel[i].setIcon(new ImageIcon(name));
                        }
                        resultLabel.setText("Video Poker");
```

```
                        bankroll = new Bankroll(0);
                        bankrollLabel.setText("Coins remaining: " + 0);
                        add1Button.setEnabled(true);
                        add5Button.setEnabled(true);
                        quitButton.setEnabled(true);
                        dealButton.setEnabled(false);
                        for(int j = 0; j < 5; j++)
                                betAndPlayButton[j].setEnabled(false);
                        for(int j = 0; j < 5; j++){
                                holdButton[j].setText("" + (j+1));
                                holdButton[j].setEnabled(false);
                        }
                }
                else if(e.getSource() == quit){
                        System.exit(0);
                }

                //Bonus menu items for fun
                /*if(e.getSource() == secret){
                        bankroll.alterBankroll(100);
                        int br = bankroll.getBankroll();
                        bankrollLabel.setText("Coins remaining: "  + br);
                        for (int i = 0; i < 5; i++){
                                if (br >= (i + 1)){
                                        betAndPlayButton[i].setEnabled(true);
                                }
                        }
                        ImageIcon image = new ImageIcon("src/Gc_EasterEgg_03_result.png");
                        JOptionPane.showMessageDialog(null, "Hidden easter egg
found!\nHere's 100 coins!", "SURPRISE!!!", JOptionPane.INFORMATION_MESSAGE,image);
                        secret.setEnabled(false);
                }

                if(e.getSource() == MySon){
                        ImageIcon image = new ImageIcon("src/IMG_2870.JPG");
                        JOptionPane.showMessageDialog(null, "", "",
JOptionPane.INFORMATION_MESSAGE,image);
                }

                if(e.getSource() == HowToPlay){
                        try{
                                Desktop.getDesktop().browse(new
URL("http://vegasclick.com/games/videopoker").toURI());
                        }
                        catch(MalformedURLException ex){

Logger.getLogger(callbrowser.class.getName()).log(Level.SEVERE, null, ex);
                        }
```

```
                                        catch(URISyntaxException ex){

                                        } catch (IOException e1) {
                                                e1.printStackTrace();
                                        }
                                }*/
                        }
                }

                //Extra coding for fun
                /*public class callbrowser extends javax.swing.JFrame{
                        public callbrowser(){
                                getComponents();
                        }
                }*/

                public static void main(String[] args){
                        Player pm = new Player();
                        pm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                }
}
```

//PokerGame Class

```
public class PokerGame {
	private Bankroll bankroll;
	private Bet bet;
	private Hand hand;
	private Player player;
	private boolean[] holdCards;

	public PokerGame(Bet coinsBet, Bankroll br, Player pl) {
		bankroll = br;
		bet = coinsBet;
		player = pl;
		hand = new Hand();
		holdCards = new boolean[5];
	}

	int updateBankroll(int payoff) {
		int winnings = payoff * (bet.getBet());   // negative for a loss
		bankroll.alterBankroll(winnings);
		return winnings;
	}

	public void viewInitialHand() {
		hand.newHand();
		player.displayHand(hand);
	}

	public void discardOrHoldCards() {
		player.getDiscard(holdCards);
		hand.updateHand(holdCards);
		player.displayHand(hand);
		int payoff = hand.evaluateHand();
		int winnings = updateBankroll(payoff);
		player.displayResults(payoff, winnings);
	}
}
```

//Bet Class

```
import java.util.Scanner;

public class Bet {
   private int bet;
   public Bet(){  //default constructor sets bet to 0
         bet = 0;
   }

   public Bet(int n) {       //one-argument constructor, sets bet to n
         bet = n;
   }

   public void setBet(int n) {
      bet = n;
   }

   public int getBet() {    //getter
      return bet;
   }

   //Test
   public static void main(String[] args){
         Scanner in = new Scanner(System.in);
         System.out.println("Enter an integer: ");
         int n = in.nextInt();
         Bet bet1 = new Bet();
         System.out.println("Getter" + bet1.getBet());
         bet1.setBet(n);
         System.out.println("After Setter" + bet1.getBet());
         Bet bet2 = new Bet(n);
         System.out.println("Getter;" + bet2.getBet());
         bet2.setBet(n + 10);
         System.out.println("Getter;" + bet1.getBet());
   }
}
```

//Deck Class

```
import java.util.*;
public class Deck {
   private Card[] deck;
   private int next;          //holds position of next card to be dealt

   public Deck() {
      deck = new Card[53]; // Does not use position 0, uses 1..52.

      for (int rank = 1; rank <= 13; rank++) {
         // Place cards in order in deck.
         deck[rank] = new Card(1, rank); // Hearts.
         deck[rank + 13] = new Card(2, rank); // Diamonds.
         deck[rank + 26] = new Card(3, rank); // Clubs.
         deck[rank + 39] = new Card(4, rank); // Spades.
      }
      next = 1;    //first card dealt is deck[next]
   }

   public void shuffle() {
      Random randomNumber = new Random();

      for (int card = 1; card <= 52; card++) {
         int rand = randomNumber.nextInt(52) + 1;
         // Swap deck[card] with deck[r].
         Card temp = deck[card];
         deck[card] = deck[rand];
         deck[rand] = temp;
      }

      next = 1; // Top card of the deck.
   }

   public Card deal() {
      if (next > 52) // If deck is depleted...
         shuffle();
      Card card = deck[next];
      next++;
      return card;
   }
}
```

//Card Class

```
import java.util.Scanner;
public class Card {
	private int suit;
	/**
	 * 1: Hearts
	 * 2: Diamonds
	 * 3: Clubs
	 * 4: Spades
	 */

	private int value;
	/**
	 * 1: Ace
	 * 2-10: 2-10
	 * 11: Jack
	 * 12: Queen
	 * 13: King
	 */

	public Card(){    //Ace of Hearts by Default
		suit = 1;
		value = 1;
	}

	public Card(int s, int v) {
		suit = s;
		value = v;
	}

	public int getSuit() {
		return suit;
	}

	public int getValue() {
		return value;
	}

	public void setSuit(int s) {
		suit = s;
	}

	public void setValue(int v) {
		value = v;
	}

	//Added images to the src/ folder
	public String getName() { // Returns string, e.g., "Ace of Hearts".
```

```
String name = "";
if (value == 1){
	if (suit == 1)
		name += "ha";//"Hearts";
	else if (suit == 2)
		name += "da";//"Diamonds";
	else if (suit == 3)
		name += "ca";//Clubs";
	else if(suit == 4)
		name += "sa";//"Spades";
}

else if (value == 2){
	if (suit == 1)
		name += "h2";//"Hearts";
	else if (suit == 2)
		name += "d2";//"Diamonds";
	else if (suit == 3)
		name += "c2";//Clubs";
	else if(suit == 4)
		name += "s2";//"Spades";
}

else if (value == 3){
	if (suit == 1)
		name += "h3";//"Hearts";
	else if (suit == 2)
		name += "d3";//"Diamonds";
	else if (suit == 3)
		name += "c3";//Clubs";
	else if(suit == 4)
		name += "s3";//"Spades";
}

else if (value == 4){
	if (suit == 1)
		name += "h4";//"Hearts";
	else if (suit == 2)
		name += "d4";//"Diamonds";
	else if (suit == 3)
		name += "c4";//Clubs";
	else if(suit == 4)
		name += "s4";//"Spades";
}

else if (value == 5){
	if (suit == 1)
		name += "h5";//"Hearts";
```

```
		else if (suit == 2)
			name += "d5";//"Diamonds";
		else if (suit == 3)
			name += "c5";//Clubs";
		else if(suit == 4)
			name += "s5";//"Spades";
	}

	else if (value == 6){
		if (suit == 1)
			name += "h6";//"Hearts";
		else if (suit == 2)
			name += "d6";//"Diamonds";
		else if (suit == 3)
			name += "c6";//Clubs";
		else if(suit == 4)
			name += "s6";//"Spades";
	}

	else if (value == 7){
		if (suit == 1)
			name += "h7";//"Hearts";
		else if (suit == 2)
			name += "d7";//"Diamonds";
		else if (suit == 3)
			name += "c7";//Clubs";
		else if(suit == 4)
			name += "s7";//"Spades";
	}

	else if (value == 8){
		if (suit == 1)
			name += "h8";//"Hearts";
		else if (suit == 2)
			name += "d8";//"Diamonds";
		else if (suit == 3)
			name += "c8";//Clubs";
		else if(suit == 4)
			name += "s8";//"Spades";
	}

	else if (value == 9){
		if (suit == 1)
			name += "h9";//"Hearts";
		else if (suit == 2)
			name += "d9";//"Diamonds";
		else if (suit == 3)
			name += "c9";//Clubs";
```

```
		else if(suit == 4)
			name += "s9";//"Spades";
	}

	else if (value == 10){
		if (suit == 1)
			name += "ht";//"Hearts";
		else if (suit == 2)
			name += "dt";//"Diamonds";
		else if (suit == 3)
			name += "ct";//Clubs";
		else if(suit == 4)
			name += "st";//"Spades";
	}

	else if (value == 11){
		if (suit == 1)
			name += "hj";//"Hearts";
		else if (suit == 2)
			name += "dj";//"Diamonds";
		else if (suit == 3)
			name += "cj";//"Clubs";
		else if(suit == 4)
			name += "sj";//"Spades";
	}

	else if (value == 12){
		if (suit == 1)
			name += "hq";//"Hearts";
		else if (suit == 2)
			name += "dq";//"Diamonds";
		else if (suit == 3)
			name += "cq";//"Clubs";
		else if(suit == 4)
			name += "sq";//"Spades";
	}

	else if (value == 13){
		if (suit == 1)
			name += "hk";//"Hearts";
		else if (suit == 2)
			name += "dk";//"Diamonds";
		else if (suit == 3)
			name += "ck";//"Clubs";
		else if(suit == 4)
			name += "sk";//"Spades";
	}
```

```
            return name;
        }

        //Test
        public static void main(String[] args){
            for (int s = 1; s <=  4; s++){ // 4 suits
                for (int val = 1; val <= 13; val++){ // 13 cards per suit {
                    Card cd = new Card(s, val);
                    System.out.println(s + "," + val + ": " + cd.getName());
                }
            }
            Scanner input = new Scanner(System.in);
            System.out.print ("Suit: ");
            int s = input.nextInt();
            System.out.print("Value: ");
            int val = input.nextInt();
            Card cd = new Card(s, val);
            System.out.println(s + "," + val + ": " + cd.getName());
        }
}
```

```
//Hand Class
public class Hand
{
	private Card[] cards;
	private Deck deck;
	private int suits[];  // holds the number of each suit in a hand
	private int values[]; // holds the number of each type card (A,2,3,4,...K)

	public Hand()
	{
		cards = new Card[5];
		suits = new int[5];	// uses indices 1..4
		values = new int[14];	// uses indices 1..13
		deck = new Deck();
	}

	public void newHand()
	{
		deck.shuffle();
		for (int i = 0; i < 5; i++)
		{
			cards[i] = deck.deal();
			suits[cards[i].getSuit()]++ ;
			values[cards[i].getValue()]++;
		}
		sort();
	}

	public void  updateHand(boolean[] x)
	{
		for (int i = 0; i < 5; i++)
			if (!x[i])
			{
				// remove card data for card i
				suits[cards[i].getSuit()]-- ;
				values[cards[i].getValue()]--;

				// get a new card
				cards[i] = deck.deal();

				// update data for card i
				suits[cards[i].getSuit()]++ ;
				values[cards[i].getValue()]++;
			}
		sort();
	}

	public String[] getHand()
```

```
{
        String[] cardsInHand = new String[5];
        for (int i = 0; i < 5; i++)
                cardsInHand[i] = cards[i].getName();
        return cardsInHand;
}

private void sort()  // orders cards by value field; a helper function
{
        int max; // holds the position of the highest valued card
        for (int place = 4; place > 0; place--)
        {
                max = 0;
                // find the position of the highest valued card between 0 ans place
                // the position of the high card is stored in max
                for (int i = 1; i <= place; i++)
                        if ( cards[i].getValue() > cards[max].getValue())
                                max = i;
                // swap the highest wlaued card with the card in position place
                Card temp = cards[place];
                cards[place] = cards[max];
                cards[max] = temp;
        }
}
public int  evaluateHand()
{
        if (royalFlush())  // royal flush pays 250:1
                return 250;
        else if (straightFlush()) // straight flush pays 50:1
                return 50;
        else if (fourOfAKind()) // four of a kind
                return 25; // four of a kind pays 25:1
        else if (fullHouse()) // full house
                return 9;
        else if (flush())
                return 6;
        else if (straight())
                return 4;
        else if (threeOfAKind()) // three of a kind
                return 3;
        else if (twoPair())
                return 2;
        else if (pair())  // Jacks or better
                return 1;
        return -1;   // losing hand
}

private boolean royalFlush()
```

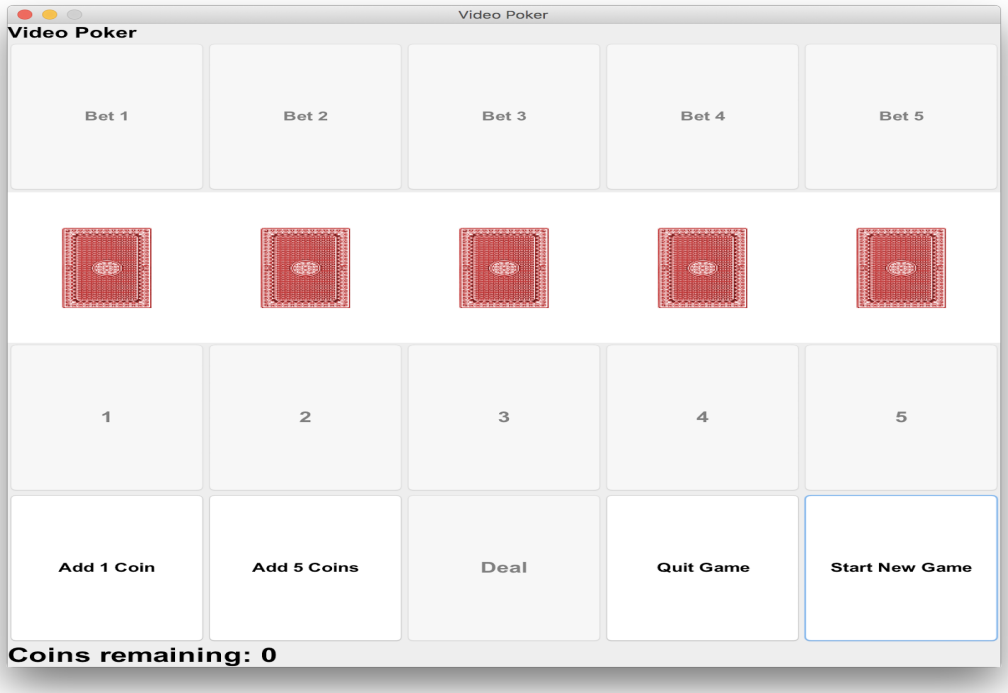
```
{
        //10, J,Q,K,A of the same suit
        boolean sameSuit= false;  // true if all same suit
        boolean isRoyalty= false; //  true if cards are 10,J,K,Q,A
        for(int i = 1; i <=4; i++)
                if (suits[i] == 5)  // all five cards of one suit?
                        sameSuit = true;
        isRoyalty = (values[1] == 1 &&
                        values[10] ==1 &&
                        values[11] ==1 &&
                        values[12] == 1 &&
                        values[13] == 1);
        return (sameSuit && isRoyalty); // true if both conditions are true
}
private boolean straightFlush()
{
        boolean sameSuit = false;
        boolean ranksInOrder = false;
        for(int i = 1; i <=4; i++)
                if (suits[i] == 5)
                        sameSuit = true;  // same suit?
        // cards in sequence?
        ranksInOrder =
                        cards[1].getValue() == (cards[0].getValue() + 1) &&
                        cards[2].getValue() == (cards[0].getValue() + 2) &&
                        cards[3].getValue() == (cards[0].getValue() + 3) &&
                        cards[4].getValue() == (cards[0].getValue() + 4);
        return (sameSuit && ranksInOrder);
}

private boolean flush()
{
        for(int i = 1; i <=4; i++)
                if (suits[i] == 5)  // all the same suit?
                        return true;
        return false;
}

private boolean fourOfAKind()
{
        for(int i =1 ; i <= 13; i++)
                if (values[i] == 4)
                        return true;
        return false;
}

private boolean fullHouse()
{
```

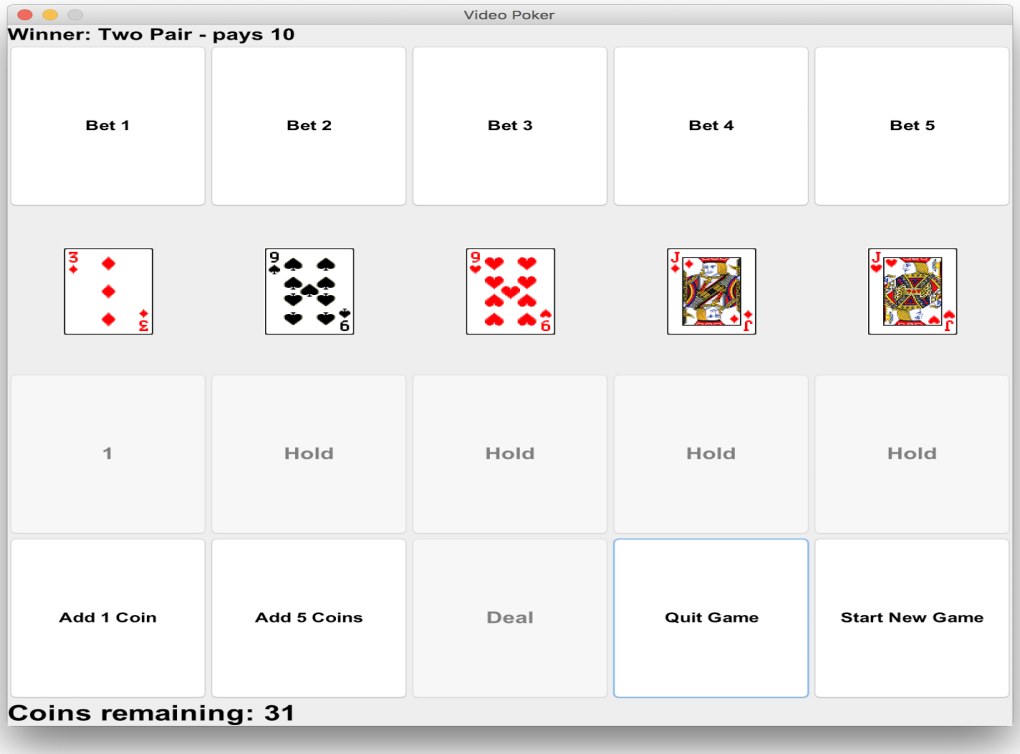
```
        boolean three= false;
        boolean two= false;
        for(int i =1 ; i <= 13; i++)
                if (values[i] == 3)  // three of one kind
                        three= true;
                else if (values[i] ==2) // two of another kind
                        two = true;
        return  two && three; // both conditions
}
private boolean straight()
{
        // cards in sequence?
        return
                        // Ace precedes 2
                        (cards[1].getValue() == (cards[0].getValue() + 1) &&
                        cards[2].getValue() == (cards[0].getValue() + 2) &&
                        cards[3].getValue() == (cards[0].getValue() + 3) &&
                        cards[4].getValue() == (cards[0].getValue() + 4))  ||
                        //Ace follows King
                        (values[1] == 1 && //Ace
                        values[10] ==1 &&  //Ten
                        values[11]==1 &&  //Jack
                        values[12] == 1 && //Queen
                        values[13] == 1); //King
}

private boolean threeOfAKind()
{
        for(int i =1 ; i <= 13; i++)
                if (values[i] == 3)
                        return true;
        return false;
}

private boolean twoPair()
{
        int count = 0;
        for( int i = 1; i <= 13; i++)
                if(values[i] == 2)   // count the number of pairs
                        count++;
        return (count == 2);
}
private boolean pair() // Jacks or Higher
{
        if (values[1] == 2) //pair of aces
                return true;
        for( int i = 11; i <= 13; i++) // pair of Jacks or higher
                if(values[i] ==2)
```

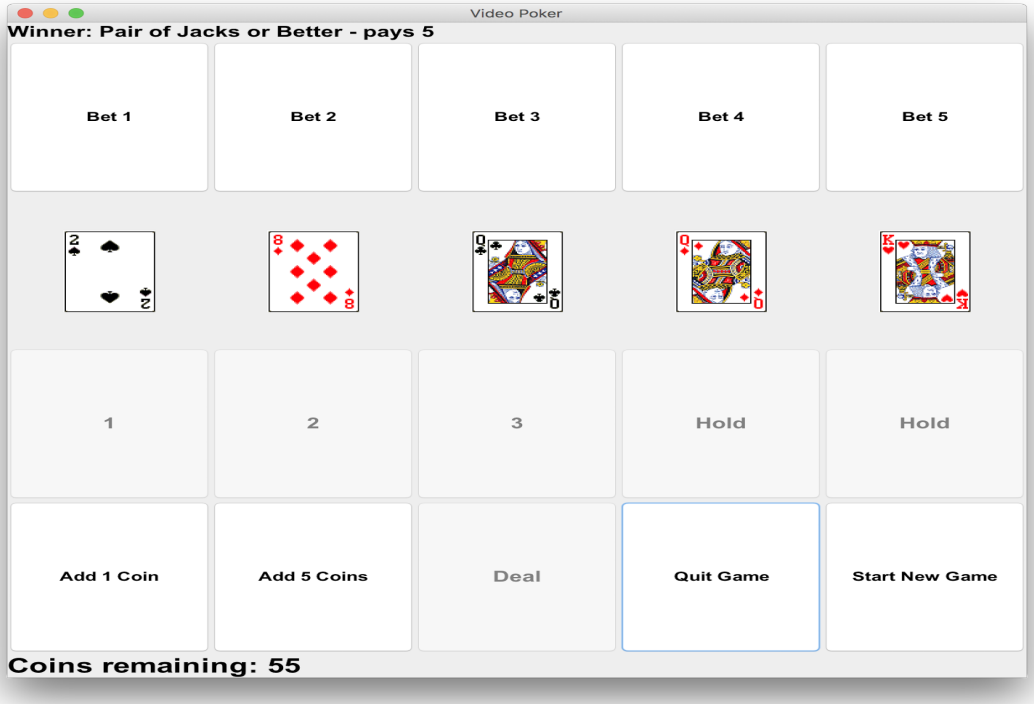
```
                                    return true;
                        return false;
            }
}
```

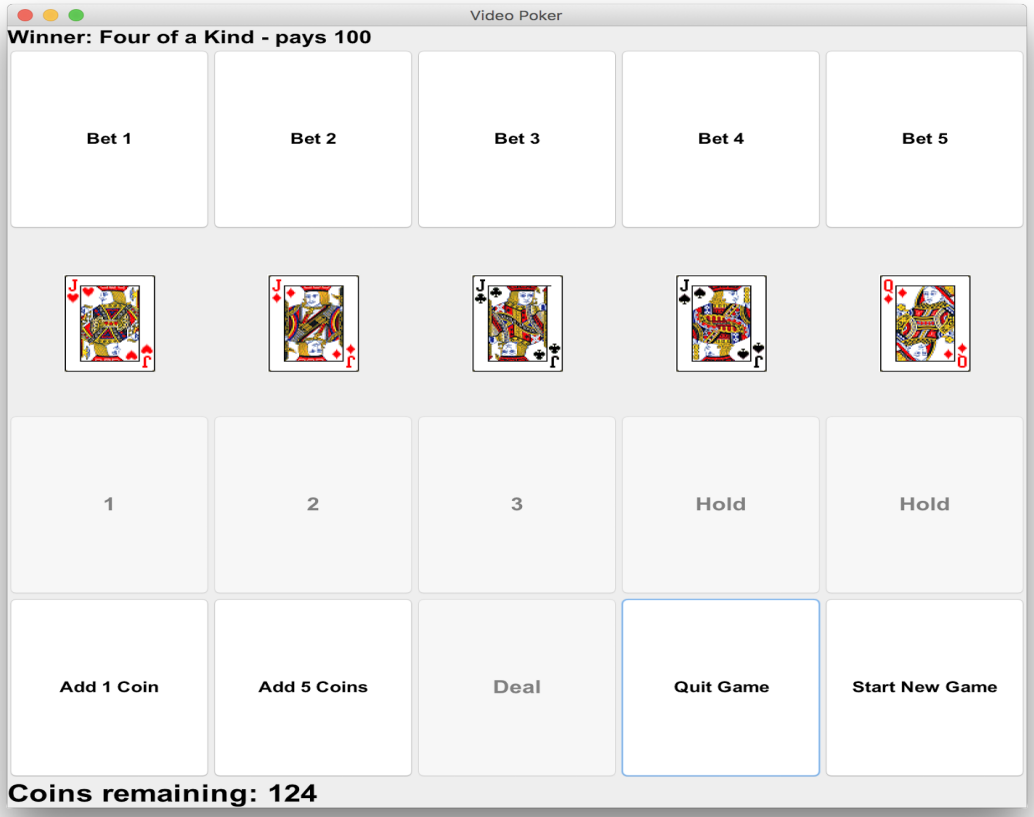
```
//Bankroll Class
import java.util.Scanner;
public class Bankroll {
	private int bankroll;

	public Bankroll(){		//default constructor
		bankroll = 0;
	}

	public Bankroll(int n) { //one argument
		bankroll = n;
	}

	public int getBankroll() {
		return bankroll;
	}

	public void setBankroll(int n){
		bankroll = n;
	}

	public void alterBankroll(int n) {
		bankroll += n;
	}

	//Test
	public static void main(String[] args){
		Scanner in = new Scanner(System.in);
		System.out.println("What is your bankroll?");
		int bank = in.nextInt();
		Bankroll bankroll = new Bankroll();
		bankroll.setBankroll(bank);
		System.out.println("Bankroll: " + bankroll.getBankroll());
	}
}
```

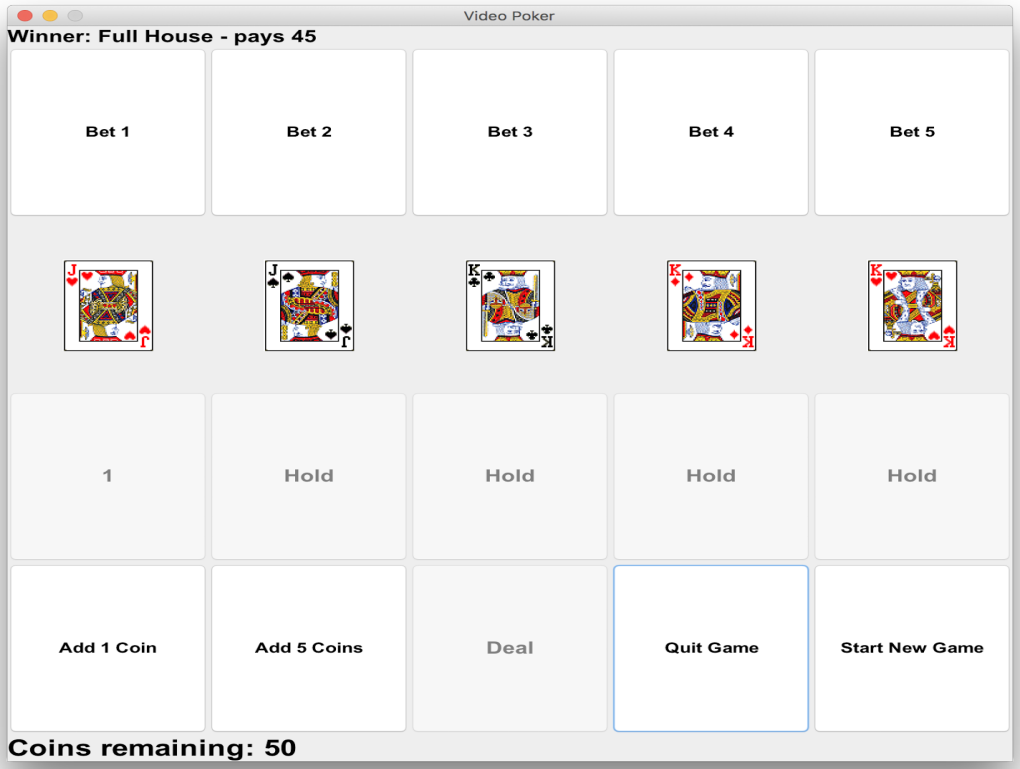
Some Examples of My Video Poker Program:

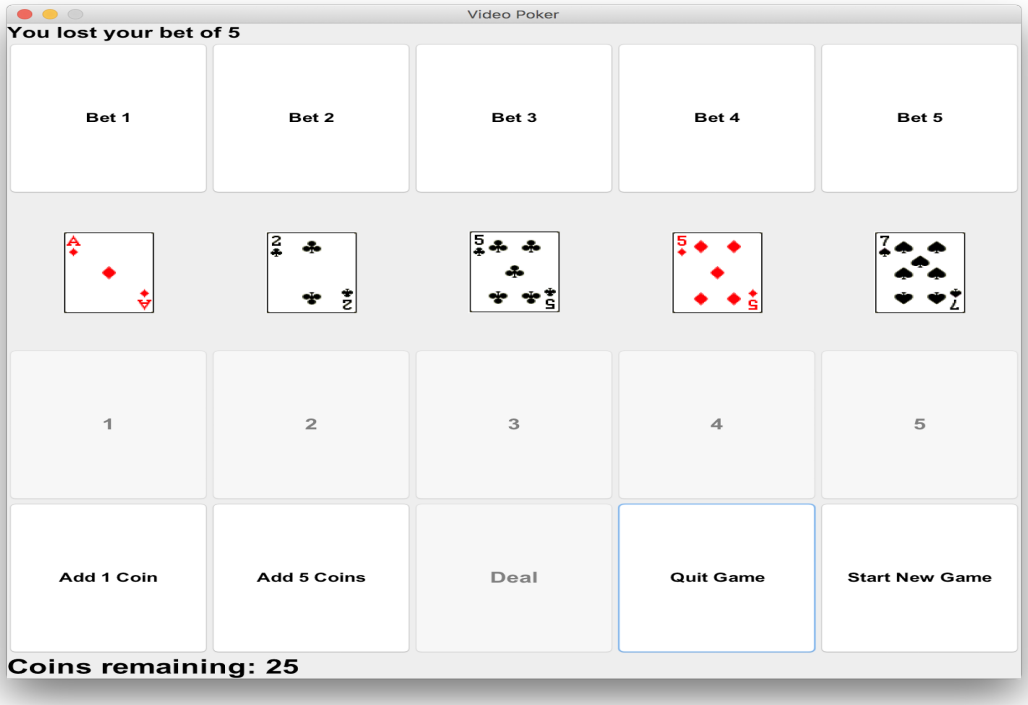





Video Poker
Winner: Pair of Jacks or Better - pays 5
Bet 1
Bet 2
Bet 3
Bet 4
Bet 5
1
2
3
Hold
Hold
Add 1 Coin
Add 5 Coins
Deal
Quit Game
Start New Game
Coins remaining: 55


Video Poker
Winner: Four of a Kind - pays 100
Bet 1
Bet 2
Bet 3
Bet 4
Bet 5
1
2
3
Hold
Hold
Add 1 Coin
Add 5 Coins
Deal
Quit Game
Start New Game
Coins remaining: 124


Video Poker
Winner: Full House - pays 45
Bet 1
Bet 2
Bet 3
Bet 4
Bet 5
1
Hold
Hold
Hold
Hold
Add 1 Coin
Add 5 Coins
Deal
Quit Game
Start New Game
Coins remaining: 50


Video Poker
You lost your bet of 5
Bet 1
Bet 2
Bet 3
Bet 4
Bet 5
1
2
3
4
5
Add 1 Coin
Add 5 Coins
Deal
Quit Game
Start New Game
Coins remaining: 25