

## High/Low Card Game – Completed Classes and Applet

Attached is the completed code for the `Card`, `CardDeck` and `Hand` classes, as well as the code for the High/Low Applet.

### **Please note:**

The card images are not included. Card images can be downloaded from many Web sites including <http://www.waste.org/%7Eoxymoron/cards/>

For ease of programming, I named each card image suit name + face name + ".gif" (e.g., "spades5.gif", "heartsace.gif", "clubsking.gif"). If you want these named images you can email [evier49@aol.com](mailto:evier49@aol.com).

### The Card Class

```
import java.awt.*;

public class Card implements Comparable
{
    private String myFace;
    private String mySuit;
    private int myRank;
    //assumes that 2 is low card, ace is high card
    private String [] face={"2","3","4","5","6","7","8","9","10","Jack",
        "Queen","King","Ace"};
    private String [] suit={"hearts","clubs","spades","diamonds"};

    public Card(int f,int s)
    {
        myFace=face[f];
        mySuit=suit[s];
        myRank=f;
    }

    public String getFace()
    {
        return myFace;
    }

    public String getSuit()
    {
        return mySuit;
    }

    public int getRank()
    {
        return myRank;
    }

    public int compareTo(Object o)
    {
        Card other=(Card)o;
        return myRank-other.getRank();
    }
}
```

## The CardDeck Class

```
public class CardDeck
{
    private int numSuits=4, numFaces=13;
    private Card[][] myDeck=new Card[numSuits][numFaces];
    private int myNextCardSuit=0,myNextCardFace=0;

    public CardDeck()
    {
        int f,s;
        //this sets up an unshuffled deck of cards
        for(s=0;s<numSuits;s++)
            for (f=0;f<numFaces;f++)
                myDeck[s][f]=new Card(f,s);
        System.out.println("Here is the ordered deck:");
        printDeck(myDeck);
        System.out.println();
    }

    public void shuffle()
    {
        //random exchange; in-place shuffle
        int f,s,r_f,r_s;
        Card temp;
        for (s=0;s<numSuits;s++)
            for (f=0;f<numFaces;f++)
                {
                    r_s=(int)(Math.random()*numSuits);
                    r_f=(int)(Math.random()*numFaces);
                    //swap
                    temp=myDeck[s][f];
                    myDeck[s][f]=myDeck[r_s][r_f];
                    myDeck[r_s][r_f]=temp;
                }

        System.out.println("Here is the shuffled deck:");
        printDeck(myDeck);
    }

    public void printDeck(Card [] [] d)
    {
        int i,j;
        String face, suit;
        for (i=0;i<4;i++)
            for (j=0;j<13;j++)
                {
                    face=d[i][j].getFace();
                    suit=d[i][j].getSuit();
                    System.out.println(face + " of " +suit);
                }
    }

    public Card deal()
    {
        Card c1=myDeck[myNextCardSuit][myNextCardFace];
        myNextCardFace++;
    }
}
```

```

        if (myNextCardFace==numFaces)
        {
            myNextCardSuit++;
            myNextCardFace=0;
        }

        return cl;
    }
}

```

## **The Hand Class**

```

import java.util.*;
public class Hand
{
    private int mySize;
    private ArrayList<Card> cardList=new ArrayList();
    public Hand(int size)
    {
        mySize=size;
    }

    public int getSize()
    {
        return mySize;
    }

    public void addCard(Card c)
    {
        if (cardList.size()<mySize)
            cardList.add(c);
    }

    public Card getCard(int i)
    {
        return (Card)cardList.get(i);
    }

    public void printHand()
    {
        int i;
        Card c;
        for (i=0;i<cardList.size();i++)
        {
            c=(Card)cardList.get(i);
            System.out.println(c.getFace() + " of " + c.getSuit());
        }
    }

    private void changeCard(int pos,Card c)
    {
        cardList.set(pos,c);
    }

    public void sort()
    {
        //sort the Hand in non-decreasing order by face
        int lowIndex;
        Card c;
        int i;
        for (i=0;i<getSize();i++)
        {
            lowIndex=findLowestIndex(i);
            //swap

```

```
        c=getCard(i);
        changeCard(i,getCard(lowIndex));
        changeCard(lowIndex,c);
    }
}
private int findLowestIndex(int startIndex)
{
    int i;
    int low=startIndex;
    Card lowCard=getCard(low);
    for (i=startIndex+1;i<getSize();i++)
        if (getCard(i).compareTo(lowCard)<0)
            {
                low=i;
                lowCard=getCard(i);
            }

    return low;
}
}
```

## The Applet

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class highlow extends Applet implements ActionListener
{
    Label title=new Label("High Low Card Game");
    Image [] cardimages=new Image[5];
    CardDeck deck;
    Hand h;
    Label msg=new Label();
    Button higher=new Button("Higher");
    Button lower=new Button("Lower");
    Button newgame=new Button("New Game");
    int nextCard;

    public void update(Graphics g)
    {
        paint(g);
    }
    public void init()
    {
        int i;
        setLayout(null);
        add(title);
        title.setBounds(200,0,200,30);

        add(higher);
        higher.addActionListener(this);
        higher.setBounds(100,150,80,30);
        add(lower);
        lower.addActionListener(this);
        lower.setBounds(200,150,80,30);
        add(newgame);
        newgame.addActionListener(this);
        newgame.setBounds(300,150,80,30);
        newgame.setEnabled(false);
        add(msg);
        msg.setBounds(200,220,200,30);

        setup();
    }

    public void setup()
    {
        int i;
        deck=new CardDeck();
        deck.shuffle();
        h=new Hand(5);
        for (i=0;i<h.getSize();i++)
            h.addCard(deck.deal());
        nextCard=1;
        Card first=h.getCard(0);
        cardimages[0]=getImage(getDocumentBase(),first.getSuit()+first.getFace() +
".gif");
        for (i=1;i<cardimages.length;i++)
```

```

        {
            cardimages[i]=getImage(getDocumentBase(),"cardback.gif");
        }
    }

public void actionPerformed(ActionEvent e)
{
    Card c;
    int rank1,rank2;
    //3 ways to get here: Lower, Higher or Play Again button
    // add code to handle each of these
    if (e.getSource()==newgame)
        again();
    else
    {
        //turn over the next card

        c=h.getCard(nextCard);

cardimages[nextCard]=getImage(getDocumentBase(),c.getSuit()+c.getFace() + ".gif");
        rank1=((Card)(h.getCard(nextCard-1))).getRank();
        rank2=c.getRank();
        if (e.getSource()==lower)
            if (rank2<rank1)
                nextCard++;
            else
            {
                //JOptionPane.showMessageDialog(null,"Wrong. You lose");
                msg.setText("WRONG. YOU LOSE!!");
                higher.setEnabled(false);
                lower.setEnabled(false);
                newgame.setEnabled(true);
            }
        if (e.getSource()==higher)
            if (rank2>rank1)
                nextCard++;
            else
            {
                // JOptionPane.showMessageDialog(null,"Wrong. You lose");
                msg.setText("WRONG. YOU LOSE!!");
                higher.setEnabled(false);
                lower.setEnabled(false);
                newgame.setEnabled(true);
            }
    }

    repaint();
    if (nextCard==5)
    {
        //JOptionPane.showMessageDialog(null,"You Win!");
        msg.setText("YOU WIN!!");
        higher.setEnabled(false);
        lower.setEnabled(false);
        newgame.setEnabled(true);
    }
}

```

```
public void paint(Graphics g)
{
    int i,x,y;
    x=70;
    y=50;
    for (i=0;i<cardimages.length;i++)
    {
        g.drawImage(cardimages[i],x,y,60,90,this);
        x=x+70;
    }
}

public void again()
{
    lower.setEnabled(true);
    higher.setEnabled(true);
    newgame.setEnabled(false);
    msg.setText("");
    setup();
}
}
```