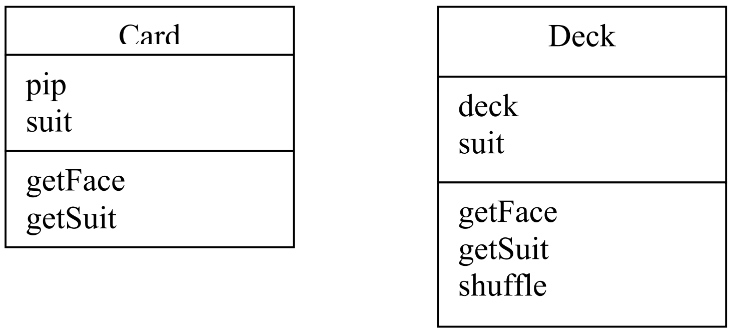
Java – Inheritance 2

C&G criteria: 2.2.6, 2.2.7, 2.3.1, 2.3.2, 3.5.3.

So far user defined Classes have been created as individual Class files, one Class to each file, individually compiled to separate Java files. It is possible to have more than 1 Class within a single file although only 1 of those Classes may be public. In other words if a file contains more than 1 Class only the single public Class is available to create objects within other Classes. This is not important where a Class is designed specifically to create a number of objects within a single other Class.

In this example a deck of cards is created to model playing card games. The deck itself is a useful Class that could turn up as an object in many card-playing Classes. Each deck is made up of 52 card objects. Another useful application of the deck is a hand of cards chosen from the deck. Unlike the card and deck the hand Class might have to be adapted for different card games.



Deck extends card and overrides the methods getFace and getSuit

Here are the full Class codes for Card and Deck. Note that these can be saved as a single Class file.

```
public class Card
{
        protected int pip;
        protected String suit;

        public int getFace()
        {
                return pip;
        }
        public String getSuit()
        {
                return suit;
        }
}
```

```
class Deck extends Card  //inherit all the properties of the public Card
{
            protected Card [] deck = new Card [52]; //array of 52 cards – 0 to 51
            final String [] suit = {"Hearts","Diamonds","Spades","Clubs"};

            public Deck() //constructor – set up a standard deck
            {
                        int number = 0;
                        for (int s =0;s<4;s++)
                        {// 2 nested for loops to new all the cards in each of 4 suits
                               for( int pips = 1; pips<14;pips++)
                              {
                                  deck[number]=new Card();
                                  deck[number].suit=suit[s]; //suit[0]="Hearts" etc.
                                  deck[number].pip=pips;  //court cards are 11+
                                  number ++;
                              }
                         }
                        shuffle(deck);
            }
             public int getFace(int num)
            {//override card method
                       return deck[num].pip;
            }
            public String getSuit(int num)
            {//override card method
                       return deck[num].suit;
            }
            protected void shuffle (Card [] array)
            {
                       Card temp =new Card();
                       int rand=array.length;
                      for (int pos =0;pos<array.length-1;pos++)
                      { //bubble swap each card in turn with Card temp
                           rand = (int)(java.lang.Math.random()*52);
                          temp = array[pos];
                          array[pos]=array[rand];
                          array[rand]=temp;
                       }
            }
}
```

Objects from these Classes will be used to play pontoon. Create an interface to check that these Classes work.

To play pontoon a specific hand object geared for this game must be created. This can be of any number of cards (ignoring the bonus for 5 or 6 card hands). There must be a method to know when the pip total reaches or exceeds 21. To add some utility other methods will allow the computer to choose another card automatically. As a limited artificial intelligence the computer will pick another card if its current score is less than or equal to 19. As another simplification an ace will be used as 1 only. Note that in pontoon the card suit does not affect the play of the game. It does add flavour and the suit method might be used by another game with Card objects.

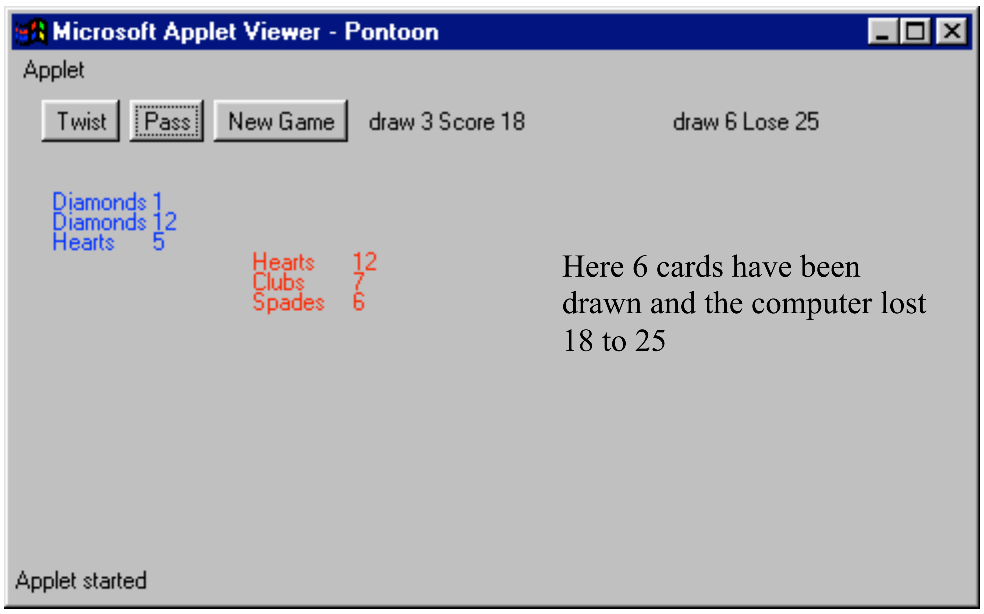
This is a suggested outline for the pontoon Hand Class. The computer must have a Hand. The human can use a Hand object or more simply pick cards and work out the score from the screen output.

Hand
score cardNo winning
newScore setScore setCard getCardNo getScore getwin playPontoon

Hand Class extends Deck.

- newScore resets the score to 0
- setScore increments the current score
- setCard draws the next card from the shuffled deck
- getCardNo returns the array position of the deck – this method is for error checking
- getScore returns the current score0
- getWin returns winning false or true
- playPontoon is used by a computer to decide to play another card or stick

The Classes Card, Deck and Hand will all need another interface Class to test the game. Unlike the other 3 Classes the interface Class will be in its own Class file. It is suggested that the interface be built up as the other Classes are created to allow the methods to be tested as coding progresses.



```
draw 3 Score 18                 draw 6 Lose 25

Diamonds 1
Diamonds 12
Hearts    5
                  Hearts    12
                  Clubs     7
                  Spades    6

Applet started
```

Here 6 cards have been drawn and the computer lost 18 to 25