School of EPS.

EECE Department

B39HV2. Software engineering II

Assignment- An Object-Oriented Card Game

This assignment is intended as an exercise in classes, including association, aggregation and inheritance.

AIM: You are required to write a C++ program that implements one of two card games.

Note: Source Code Plagiarism software will be run on your solutions and even elaborate plagiarism will be detected. Work at your best of your abilities on your own and you will get the most of the exercise.

Background:

One of the biggest advantages of object-oriented programming is that, once you have created a class, it can be used in any application that involves objects of that class. This reflects the utility of objects in real life, where a deck of card, for instance, can be used to play a variety of card games. The rules of each game may vary but the same cards are used. We will take this example as the basis of our assignment, where the aim is to create a card game using object-oriented principles. Two suitable card games are suggested, namely " Play you cards right " and the memory game "Pelmanism". Each game is described below.

Play your card right:

This game requires a single deck of cards. The deck will be shuffled and the top card is displayed. The player will place a bet on whether they believe the next card is higher or lower than the displayed card (in this version Aces count low). Once the stake is made, the next card is displayed and if the bet is won, the player will get twice their stake money repaid, if it is lost, they will loose their stake. The player will always loose if the next card is of the same face value as the current card. The game will end when:

- The player runs out of money
- The deck has ran out of cards.

The value of cards are defined as below:

King > Queen > Jack > 10 > 9 > 8 > 7 > 6 > … > 2 > Ace.

Pelanism:

For the more ambitious. This memory game requires a single deck of cards. The deck should be shuffled (same as previous game) and dealt into 4 rows of 13 columns (52 cards). The player selects two cards and turns them over. If they have the same face value, they are removed as a pair. Other wise, they are turned face down again and the player plays again. The objective of the game is to collect all pairs in a minimum number of tries.

TIP: The characters used to depict card suits are part of the ASCII character set:

(Hearts = ASCII 3, Diamonds = ASCII 4, Spades = ASCII 5, Clubs = ASCII 6).

For instance, we can display a heart as:

```
 cout <<  char(3) << endl;
```

Note however that this is only possible on windows. If you are using a UNIX xterm, you will have to use a letter such as "2C" for 2 of clubs as fonts and characters are handle differently.

<u>Program:</u>

As you will be using object-oriented programming for this assignment, the correct starting point is to consider which classes and objects are required. Two obvious candidates are a class card to represent playing cards and a class deck representing a pack of cards. To determine what these classes will look like, you might want to consider the states and behviours that these classes will exibit. For instance a card can have a face value (Ace, 1, … Jack, Queen, King) and a suit (Heart, …, Clubs). They can also be face up or face down… What behaviours can a card have? They can be turned over, (which will affect their state). In programming terms, we might want to add other behaviours such as display and compare it with other cards (TIP: you might want to use operator overloading here). A deck is a collection of cards and you can shuffle it, display it, deal cards, etc…

Start writing a *Card* Class and test it using a suitable driver. Once you have managed to do that, implement the Deck Class. Again, test it using a suitable driver. Once you have done this, you can tackle one of the games described earlier. The game itself should ideally be implemented using a class or classes.

<u>SUBMISSION</u>

You should submit printouts of your program files with your name clearly marked on it. You should also submit a floppy or make the software available (indicate where it is on your linux account and make sure it is accessible). Deadline is Friday the 14th of March 2003. Hand in your work with your name clearly marked on it in the student Office, room 2.44.

Yvan Petillot. 10/2/02