CMSI 186

PROGRAMMING LABORATORY

Spring 2008

Program 1: Video Poker

For our first problem, we'll revisit some code that you wrote in CMSI 185 — the poker hand classifier. You will build upon this code to write a text-based video poker program.

Program to Write

Write a set of Java classes that includes a *poker.VideoPoker* class. When invoked via *java poker.VideoPoker*, this program creates a deck of cards, shuffles it, then displays the top five cards of the deck to the user. Each card should be numbered from card 1 to card 5.

The program then asks the user to enter the cards that he or she would like to hold. The user is expected to enter these cards as a comma-separated list of numbers, such as "2, 3, 5." The program should be robust enough, however, to reject any other user input without terminating in an error. Incorrectly formatted input should be met with an error message and an offer to try again. If the user enters nothing, then this is interpreted as "hold *none* of the cards."

The program then replaces the cards that were *not* held with the next cards on the deck. The final set of five cards is displayed, and the program states the poker hand that was formed, if any.

Design Notes

You will need at least three other classes in addition to *poker.VideoPoker*:

- *poker.Card* represents a single playing card, with methods for determining that card's suit and rank, as well as representing that card as a human-readable string
- *poker.Deck* represents a standard set of 52 playing cards, with methods for accessing individual cards in that deck as well as shuffling the deck and checking it for validity
- *poker.PokerHandClassifier* holds routines for determining whether a set of cards contains a known poker hand

And, because *PokerHandClassifier* can be somewhat involved, you should have a *PokerHandClassifierTest* class as well.

Gotchas

- Remember that poker hands aren't disjoint — a full house contains a pair and a trio — but of course the full house "wins" because it's ranked higher. Make sure to consider that.
- Processing the user's input — especially in a robust manner, as specified here — can be a task in its own right. You should design your code so that these routines are easy to test, and easy to modify (this is actually true of all the code you write, but particularly compelling in this case).

Possible Enhancements

After you have fulfilled the core specifications of the video poker program, you can explore the following enhancements:

- Add a command line argument which specifies number of hands that the user wishes to play. For example, invoking:

  java poker.VideoPoker 5

  …should play 5 hands of video poker.
- Implement a scoring system; attaining the better (and rarer) poker hands results in more points and thus higher scores.
- Improve the user interface; in a text-based environment, can the program interact with the user in a manner that is easier to learn, more entertaining, and/or less prone to errors?
- The Java *enum* construct is a (relatively) new addition to the language. Look it up and see if you can use it to improve your code in some way.