



CS 341

Software Design

Homework 5

Identifying Classes, UML Diagrams

Due: Oct. 22, 11:30 PM

Objectives

- To gain experience doing object-oriented design
- To gain experience developing UML diagrams

A Word about Enumerated Types

An enumerated type is a type in which the programmer designs the set of values that make up that type. In languages without a mechanism to create enumerated types, like old versions of C and Java prior to 1.5, this is typically done by creating constants to hold each value of the type. This assignment has several places where enumerated types are the right solution. For example, the suit of a card can be any of Diamond, Club, Spade, or Heart. Java 1.5 provides a simple way to create enumerated types. For example, to define an enumerated type for suit, you would say:

```
enum Suit {CLUB, DIAMOND, HEART, SPADE};
```

This is actually a special type of class that has exactly 4 values in it. Since it is a class, though, it means you can define methods to operate on `Suit` instances.

Since we are not writing code with this assignment, it is not so important to know how to create an enumerated type in Java (but if you want to learn more at this point, I recommend that you look at <http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html>). It is important, though, to recognize when an enumerated type would be useful as in the case of a card's suit here. When you are identifying classes, it is not necessary to display these enumerated types in class diagrams. You can just use them as the types of class attributes and method parameters, much as you would use `String` in that way.

Assignment

In this assignment, you will practice the initial stages of design, as we have been doing in class. In particular, I provide an English description of a problem (Texas Hold 'Em Poker) and you are expected to identify classes, attributes and methods of those classes, and relationships between classes that you would use to design a program that allows multiple users to play this game. As with the Monopoly example in class, you should focus your attention on identifying classes in the problem domain, not the solution domain. In particular, you can ignore things like the user interface, the file system, networking, etc.

For this assignment, you should turn in one or more class diagrams that provide an overview of the classes and their relationships to each other. Every class (except those being used as enumerated types) should appear on at least one diagram. If the diagram gets cluttered, break it up into multiple diagrams, keeping the most closely-related classes together. Also, keep in mind that it is legitimate for one class to appear on multiple diagrams.

You should also turn in two sequence diagrams. One sequence diagram should show the sequence of method calls required for a player to raise a bet. The second diagram should show the sequence of method calls used to determine which player wins a hand and how that player collects the money in the pot.

Finally, you should turn in an English description of what you consider to be the interesting design decisions that you made. As we have seen in class, it is important to consider trade-offs when doing design to leave us with a system that is maintainable and understandable. In particular, be sure to consider the following questions as they relate to specific design issues we have discussed so far in class:

- Which classes should be immutable, if any?
- When is it appropriate to use inheritance?
- Which of your classes are enumerated types?
- Are the design principles of high cohesion and low coupling followed?
- Does each class encapsulate appropriate information and provide a good abstraction?

Problem Description

The problem is to do an initial design of a program that allows multiple users to play Texas Hold 'Em Poker against other players. Remember that you should focus your efforts on the design of classes from the problem domain. This description is broken down into two parts: terminology shared by many poker games, and rules specific to Texas Hold 'Em.

Basic Poker Terminology

Poker is played with a standard deck of 52 cards consisting of 4 suits and 13 ranks. Ranks in order from high to low are Ace, King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, 2. The suits are Diamonds, Hearts, Spades, and Clubs and are unordered.

Poker games generally consist of the following activities:

- Dealing cards to players
- Betting or folding
- Determining a winner based on the contents of the players' hands

What makes one poker game different from another is the details of these rules, such as how many cards are dealt, when betting occurs, etc., but there is some common terminology across all games. First, when it is a player's turn to bet, there are generally four options available (but some games may restrict when each option is available):

Fold	A player who folds drops out of the game. He/she forfeits any bets made in earlier rounds, is not able to participate in any future betting, and is unable to win the pot.
------	--

Bet or Raise a Bet	The player increases the amount that is being bet in the current round. To stay in the game, all other players, in their turn, need to bet at least this amount.
Call	The player matches the current bet amount exactly.
Check	If no player has yet bet in this round, a player can essentially bet \$0.

When the game ends, the player with the highest hand wins all the money that has been bet. Hands are ranked as follows, from highest to lowest:

1. Royal flush: Ace, King, Queen, Jack and Ten all of one suit
2. Straight flush: any five cards in sequence all of one suit. If there are multiple straight flushes, the one with the highest card wins.
3. Four-of-a-kind: four cards of the same rank, like 4 Jacks. If there are multiple Four-of-a-kinds, the highest ranking one wins.
4. Full house: three cards of one rank and two of another. The hand with the highest three-of-a-kind wins.
5. Flush: five cards from the same suit. If there are multiple flushes, the hand with the highest ranking card wins. If these are the same, the second highest is compared, and so on, to break the tie.
6. Straight: five cards in a sequence of mixed suits. In case of multiple Straights, the hand with the highest card wins.
7. Three-of-a-kind: three cards of the same rank. If there are multiple Three-of-a-kinds, the highest ranking one wins.
8. Two pairs: two cards of one rank and two cards of another rank. Ties are broken by comparing the rank of the higher pair, then the rank of the lower pair, and then the rank of the fifth card.
9. One pair: two cards of one rank. Ties are broken by comparing the rank of the pair, then the rank of the highest card not in the pair, then the second highest, etc.
10. High card: bad hand! If multiple hands have the same high card, compare the next highest, etc.

In any case where a tie cannot be broken, the pot is split evenly among those who tied.

Texas Hold 'Em Rules

Here are the sequences of actions for a single hand:

1. Dealer shuffles the deck. With each hand, the next player to the left becomes the dealer.
2. The player to the left of the dealer is required to bet half the minimum bet (before receiving any cards), called "posting a blind".
3. The next player to the left is required to post a blind equal to the full minimum bet.
4. Each player is dealt two cards, face down, called the "hole cards".
5. First betting round starts with the player third to the left of the dealer. Players can call, raise, or fold on their turn, but they may not check.
6. The dealer deals three cards face up in the middle of the table. These are called the community cards.
7. The second betting round happens, starting with the player to the left of the dealer. Players may check, call, raise or fold on this and all future betting rounds.
8. The dealer adds a face up card to the community cards.

9. There is a third round of betting beginning with the player to the left of the dealer.
10. The dealer adds a fifth face up card to the community cards.
11. There is a final betting round beginning with the player to the left of the dealer.
12. All the players who have not folded turn their cards face up. A player's hand consists of the five cards that create the best hand using the five cards in the community cards and the two cards that were dealt to the player. The player with the best hand wins the pot.

Drawing UML Diagrams

You can use any software you like to draw your UML diagrams. You can even draw them by hand if you do it neatly. Using a UML editor has the distinct advantage, though, that what you create is valid UML!

There are 2 choices for drawing UML. There is an Eclipse plug-in available called EclipseUML Studio. It is installed on the Windows side in the lab or you can download it and install it yourself. You will need a site license to use it, which I can give to you. This license will only allow you to use it if you start Eclipse while on campus.

The second alternative is a stand-alone tool called Violet, which you can download and use for free. It claims to also be available as an Eclipse plug-in but I haven't tried that.

So, what's the difference? UMLStudio has more features in it but it is also quite slow and awkward to use. Violet is very stripped down, just the basics, but is simple to use. It provides sufficient functionality for this class.

Since Violet is pretty intuitive, I don't have any further advice on using it. The next section offers some tips on UMLStudio.

Using Eclipse UMLStudio

To create a UML diagram, open the File menu, select New, and then Other. In the dialogue box that comes up, scroll down to UML diagrams and click on the triangle to select which type of diagram you want to create.

One of the cool features of this tool is that it keeps your diagrams and code consistent with each other. That is, if you add an attribute in a class in your UML diagram, the tool will add an instance variable to the corresponding Java code. Similarly, if you modify the Java code, your class diagrams will also be updated. Keeping design documentation and code consistent is one of the big hurdles in doing good documentation, so this is great! (Violet will not do this for you.)

You can download EclipseUML Studio from <http://www.omondo.com/>. We have available a site license that will allow you to use this for free while on campus. Just ask me for the file. This is an Eclipse plug-in, so it does require that you have Eclipse installed. It is again available on Linux, Mac and Windows. There is on-line help available via the Help menu. Select Help Contents. This should open up a manual in a Web browser. The UML plug-in help is under the topic "EclipseUML User Guide".

When you create a new class, a new association between classes, add an attribute to a class, or add a method to a class, a dialog box will pop up where you can provide details. Creating classes is exactly the same dialog box as when you create a class using the Java perspective within Eclipse. What follows is some advice about what information to provide in the other dialog boxes. The general rule of thumb is that if you don't understand what some choice means, it's probably not something you should use.

Attributes

The important things to fill in are: name, type, and visibility. Never select transient or volatile for the modifiers. Ignore dimension and stereotypes. If you select "Use accessors", Eclipse will generate a getter and setter method for the attribute. If you also select read-only, it will generate only the getter. You can ignore the Tagged Values pane. You should fill in the Javadoc pane.

Methods

The important things to fill in are: name, return type, visibility, parameters and exceptions. Ignore dimension and stereotypes. Never select synchronized or native for the modifiers. Also fill in the javadoc pane but ignore the other two panes.

Associations

The label is optional. Provide one if the intent of the association is not obvious. The first and second class names should be already set based on how you drew the edge. Again, ignore stereotypes.

Fill in the 1st association end and 2nd association end panes. Check Navigable if you intend there to be an instance variable in the 2nd class that refers to the 1st class. If so, you should also provide a name for it. The multiplicity can be any number in addition to the common things that appear in the pulldown menu. Ordered is only meaningful if the multiplicity is greater than 1. In that case, checking Ordered indicates that the order of the things in the collection matter (that is, it's not a set). Leave Association type set at None. Set Visibility. Never select transient or volatile for modifiers. Use accessors will again generate code for you. If the multiplicity has an upper bound of 1, you will get a getter and setter as with attributes. If the multiplicity is more than one, it generates a lot of methods, like addTitleDeed, removeTitleDeed, numberOfTitleDeeds, etc. Ignore stereotypes, dimension and qualifier.

Please ask if you have questions about how to use this tool.

Grading

Identification of classes	15 points
Identifying methods of a class	15 points
Identifying relationships between classes	15 points
Identifying attributes of classes	15 points
Class diagrams	15 points
Sequence diagrams	10 points
Discussion of design tradeoffs	15 points

Turning in your work

To turn in your work, please email me your UML diagrams (as a jpg, gif, pdf, Violet file, or the entire Eclipse project) and your design discussion.