Data Structures (810:052) Lab 3 Name:_____

Objective: To learn how to design software design with UML diagrams: class diagrams, use-case diagrams, and collaboration diagrams. Plus, how to documentation programs by writing APIs, using preconditions and postconditions, enforcing preconditions with exceptions, and using pydoc web-based documentation.

As your programs become more complex (larger in size and scope) and longer lasting (i.e., used for years in a production environment instead of just-run-once "toy" programs), you'll need to pay closer attention to program:

- design - to aid in project development, future maintenance and code reuse
- documentation - to aid fellow developers and future maintainers in understanding and using your software components correctly and effectively
- testing - to aid in software reliability and robustness

Part A: Read section 12.1 from the Lambert text. You're goal is to develop:

- use-cases and use-case diagrams (similar to Figures 12.1-12.4)
- class diagrams (similar to Figures 12.7 and 12.8)
- collaboration diagrams (similar to Figures 12.9)

for the following Video Poker programming problem.

You are to design a program that allows a user to play a video poker game that uses 5 dice. The program should be object-oriented (i.e., uses classes like a Die class, etc.) and have a GUI-based front-end (i.e., using Python's Tkinter module). The base set of rules is as follows:

- A player logs into the Video Poker game using an account name and password. The system looks up the about of money a player has on account. A new user can create an account. Any users can deposit more money into the game via a credit card.
- Each round costs $10 to play. This amount is subtracted from the user's account at the start of the round. Any user that has less that $10 on account cannot play unless they deposit more money.
- The player starts the round with five randomly rolled dice.
- The player gets two chances to enhance the hand by rerolling some or all of the five dice.
- At the end of the hand, the player's money is updated according to the following payout schedule:

Hand	Example	Payout
Two Pairs	4, 4, 3, 6, 6	$ 5
Three of a Kind	2, 4, 1, 1, 1	$ 8
Full House (a pair and a three of a kind)	3, 5, 5, 5, 3	$ 12
Four of a Kind	3, 3, 3, 3, 6	$ 15
Straight (1-5 or 2-6)	5, 2, 3, 1, 4	$ 20
Five of a Kind	3, 3, 3, 3, 3	$ 30

Part B. Read sections 2.3.6, 2.6.4, 8.1.2, and 12.2 from the Lambert text. Download and unzip the file at: www.cs.uni.edu/~fienup/cs052sum09/labs/lab3.zip . This contains a simple Die class in the die.py module.
a) Document the Die class at the module, class, method, and function levels. Include appropriate preconditions and postconditions in the documentation of each method as discussed in subsection 12.2.3.

b) For testing certain dice games, suppose we want to add a new Die method `setRoll` which takes as a parameter a roll value that is used to set the Die's roll. We might call this fuction as:

```
>>> from die import Die
>>> myDie = Die(6)
>>> myDie.setRoll(3)   # sets myDie to a roll of 3
```

Add any additional methods to the Die class that are needed to check the precondition(s) necessary for the `setRoll` method.

c) Enforce the preconditions by raising appropriate exceptions.

d) View the programmer-authored documentation for the Die class by typing help(Die) at the Idle shell prompt after importing the Die class.

e) Generate a web-page documentation for the Die class using pydoc as follows:

- Open a command prompt terminal window
- Change directories until you get to the directory containing you Die class
- Run the command (something like)

```
c:\python25\lib\pydoc.py -w die > die
```

  (Yours might have pydoc.py in a different location depending on the version and location of your Python install)
- Double-click on the die.html to view it in a web browser.

Part C. Read section 12.3 on testing. You are to develop a pyunit unit test for all the methods of the Die class developed in Part B. See section 12.3.5 and model your unit test after the TestStudent class on page 498.