Homework 4
(40 points)
This assignment will be programmed individually.

PRELIMINARIES
So that you don’t spend large amounts of time writing code, I have supplied the starter code in Java. I realize this isn’t convenient for everyone, but you should be able to use a C++ subset of Java for what you are asked to do for this assignment. I use JCreator to run the code. A separate document (posted with this one) tells you how to download and use JCreator.

Posted on the website is sample code which sets up the game of Liar’s dice and plays it with up to six random players. Study the code to see how it is implemented. Study the posted rules as well so you see how scoring occurs. In the GUI, you can see the values of both the visible and hidden dice, but the players do not see the hidden information. The code contains a director for dice images and a directory of sounds. If you make an illegal bid, you get the donkey sounds. When you extract the data from Home4.zip, do make sure you retain the directory structure, as the program is expected it.

The Action class records the moves of the game. Basically, your choices are either to increase the current bid or challenge the current bid. For security purposes, I do not let you keep your own dice, so you have to tell me what you want to reroll. For ease of programming, you just tell me what NOT to reroll. For example, if you say keep=6, I make all hidden ones and sixes visible, and reroll the rest. I realize I have limited your choices, but hopefully, this limitation is reasonable.

```java
class Action{
    boolean challenge; // True if action is to challenge current bid
    int count;          // How many of value you are bidding
    int value;          // Value of the dice you are bidding
    int keep; // which value to keep in rerolling, 0 means keep all

    Action(boolean c) {challenge = true; count = value = keep = 0;}
    Action(int c, int v){
        count = c; value = v; keep = 0; challenge = false;
    }
    Action(int c, int v, int r){
        count = c; value = v; keep = r; challenge = false;
    }

    //For those of you unfamiliar with Java, the toString function is the conventional way to allow a class
    //to print its contents. While sophisticated printing would want something fancier, toString is useful
    //for debugging.
    public String toString(){
        if (challenge) return "Challenge";
        if (reroll >0) return "Bid " + count + " of " + value + " **REROLL " + reroll;
        return "Bid " + count + " of " + value;
    }
}
```

You are to implement the Player interface for three types of players.
interface Player {
    static int DICE = 5;     // number of dice in hand
    static int MAXVALUE = 6; // maximum value of a die: 1,2,3,4,5,6
    int howMany(int which, Action currBid);
    public String toString();  // Prints player identification
    Action takeTurn(JTextArea talk, final Dice[] hand, final Action currBid, final int[] visible, final int[] visiblePP);
}

The howMany function should perform as follows. Given the fact that currBid is the last bid made and you want to bid a certain number of dice of value “which”, howMany tells you the minimal legal bid. For example, if the current bid is five 4’s, howMany(4,currBid) = 6, howMany(6,currBid) = 5 or howMany(1,currBid) = 3. This function will be the same for all players, but I wanted you to have the joy of writing it.

toString() prints whatever you choose as the name for the agent. I would suggest using a name that you recognize, but that doesn’t identify you to others. It needs to be more specific than just Alpha (as everyone will have an Alpha).

Note that the constructor must have no arguments.

The player’s responsibility is to decide what action to take given the state of the game. takeTurn has inputs which specify the state and a single Action output.

Action takeTurn(JTextArea talk, final Dice[] hand, final Action currBid, final int[] visible, final int[] visiblePP);

talk: an input parameter which gives the player access to the debug screen.
hand[]: the set of up to five dice that are yours
currBid: the bid of the previous player
visible[]: visible [0] is the number of hidden dice held by all players (including you).
             visible[i] is the number of visible dice with value i held by all players (including you).
visiblePP[]: is the visible information (as described above) for the previous player

I only give you information about you, the previous player, and yourself. I realize this isn’t complete information, but hopefully this will be enough.

You are allowed to keep ANY other information you choose.

I have coded one player (call him/her Omega). He/she makes random decisions on whether to challenge, up the current bid in the current value, in ones, or in a different value. Whether to reroll or not is also controlled by a random number generator.

The Program

Write the howMany function, to be used by all players.
Design three new players, Alpha, Beta, and Gamma (each implements Player). Implement various levels of sophistication (with Alpha being the simplest and Gamma being the most sophisticated).
Using Eagle, turn in ONLY the code for your three players. Be careful that you haven’t changed anything in the interface, or it will not work when I try to run it.

In order to facilitate the class tournament, name the players Alpha, Beta, and Gamma.

**The Documentation**
Create a separate document which is zipped with your program file(s) which contains the following.

For each player, discuss the following characteristics:
1. In words, briefly describe the motivation behind the player’s actions.
2. Would you categorize the player as risk averse, risk neutral, or risk seeking?
3. Does the player counter-speculate about what the other players will do?

Design a series of tests to answer the following questions using a table of results:
1. In a game of two players, for each player (Alpha, Beta, Gamma), does the player consistently beat Omega?
2. When you have a game consisting of Alpha, Beta, Gamma and Omega, who wins? Is the result reproducible or does it vary?
3. Does the most sophisticated of your players always win? Why or why not?

**Grading**
You will receive half the points for well documented/design code and half the points for the output. You will be graded on how well you compete against an “unknown” competitor. You will receive up to five bonus points if you perform well in the class tournament.

Some will spend hours and hours on this assignment, but realize that this is not expected nor desirable.