Title: Blackjack Simulation

Introduction (1 page):

Blackjack is a card game, also know as twenty one, where the players compete against the dealer. The objective of of Blackjack is to beat the dealer in one of the following ways:

- Get 21 points on the player's first two cards (called a "blackjack" or "natural"), without a dealer blackjack;
- Reach a final score higher than the dealer without exceeding 21; or
- Let the dealer draw additional cards until his or her hand exceeds 21.

Above is a brief introduction to the card game BlackJack. For more details, visit Wikipedia to know more.

Since BlackJack is a popular game at Casino and there is a huge cash flow for this game, many people want a way to simulate this game to learn more about it and to avoid spending money at casino to play it. As there are many more ways to play blackjack, with many different dependable variables, we would like to study the phenomenon on different chance of winning, and find the most optimal setting by comparing the win-loss ratio for some situations. Also by finding the win-loss ratio for each condition, we will study the correlation between the ratio and condition given.

Model Description (1-2 pages):

For our model, we will use the Monte Carlo model to randomly generate and assign the number for playing the blackjack. Using the Monte Carlo, the blackjack player will be given a random choice of cards, and based on the cards received they will make a decision on the next action. Also using the random model, all player are prohibited from cheating the game, and will not be under a outside interferences for playing. Each model will be under a different condition and constraints however, and thus will have a different results.
We will assume that all players will be playing by the rules and will not be given an ability to cheat. Each deck will have the proper number of cards when being dealt out to the player, and will not be swapped with a different cards. Every time a new game is played, all the cards will be shuffled to ensure no similar situation is presented.

We will have entities that represent a real Blackjack games plus entities that support the simulation like : course of actions, visualize,..:

- Dealer: This entity is the dealer of our simulation
- Player: This is the entity that is the player of our simulation

- Card: To represent our deck of cards
- Visualize: To animate the game
- Course of actions: Include different actions that the players can do
- ConstantClass: Represent the constants for the simulation. For example, number of deck of cards or number of players. By using this, we can change the simulation and make it more diverse

Dealer:

This class is the dealer of our simulation. It will be used to represent the actions the dealer will take in a real life situation. Here are the following actions that dealer can do:

- Hit: The dealer will draw card from the simulation
- Check: Check to see if other players busted or not
- ShuffleTheDeck: shuffle the deck

Player

This class is the player of our simulation. It will be used to represent the actions the player can do. Here are some of the actions the player can do:

- HIT: Request another card. The player can request a hit as many times as he/she likes, but if his/her total goes over 21, he/she will Bust and therefore lose the hand.
- STAND: the player request to receive no more cards. The current hand will be judged against the dealer's.
- SPLIT: If the player have two cards of the same denomination, he/she can split his/her cards into two hands and play each hand separately. His/her original bet will be duplicated for the new hand. Note: according to Atlantic City Blackjack rules, he/she may only draw one additional card on each ace when splitting a pair of aces. Also note that this option will normally only be available immediately after he/she receives his/her first two cards.
- DOUBLE: The player get one more card, his/her turn will end, and his/her bet will be doubled. Note: this option will normally only be available immediately after the player receive his/her first two cards, however, Doubling after Splitting is usually allowed.

Card

This card class will serve as our original 52 cards. And it will have the following option:

- SelectRandomly: select one of the 52 cards randomly, without replacement
- Shuffle: Shuffle the deck of card

- ReStockDeck: replaces the missing cards

Visualize:

This will be the visualization of winning ratio of each simulation. We will use plotlines of histogram to graph the percentage of the ratio, and after all simulation is over, we will compare the data with other simulation to find the correlation or difference. Each time a game is played, console also prints the cards each player had, and who had won the game.

ConstantClass

This will be the class that hold the constants of our simulation such as number of players, number of decks, the amount of chips, etc.

CourseOfAction

A player can either play in a random way, playing dealers rule and playing the odds

- playRandom: At each round, there will be a random choice of what the player will do next (based on the cards being held. I.e. if the user has a total of 5, the user will have a 90% chance of hitting)
- playOdds: At each round, the player will analyze what cards have been draw and determine the remaining cards, hence selecting the action that maximize the benefits (there is more than one way of playing the odds we may be implementing more than one)
- playDealer: Like playing with odds, but this time, with the dealer’s rule like 3 decks instead of 1

Analysis (1 page):

Based on the output of the model, the analysis will be, seeing what is the optimal amount of games one should play using a certain strategy with x number of people at the table, to see how many games with each method produce the highest win loss ratio. Using the basic probability analysis, by estimating the average win/ratio for given number of players and number of decks, we will also be able to calculate the confidence intervals for the ratio.
Using the regression model would also help us answer the relations between the number of games played and the win/loss ratio for each simulation.

The metric we will primary use for our analysis are win/loss ratio presented in percentage, a number of chips player will have at given point, and the money that will be directly related to how many chips are in their possession.

Some of the questions we could ask of our model are:

- Which of these methods gives one the best chance of not going bust, when the dealer has a chance of going bust?
- What are the correlation between win/loss ratio and the condition for given simulation?
- At what condition, give the most consistent play of playjack with 50/50 win-loss ratio.
- At what point has the player earned or lost their money?
- Which simulation has the smallest variance in the win/loss ratio for most dependable scenario of playing?

Testing (1/2 page):

We will use formal Python unit testing framework, from coding and handling analysis as it can help build regression model and give us the correlations and hypothesis tests.
Before compiling all the simulation together, we will test each method and class individually and independently to ensure all of the method calls for the correct required interfaces and return correct provided interface.
All simulation will be simulated 100 times per given condition, before combining all datasets for making a comparison and analyzing. If during the simulation, we find 100 runs does not give fine details of the data, we may increase the number of runs for more apparent data.

Personnel (1/2 page):

Thuan Tran: I will be responsible for hosting the group’s source code on GitHub. I will also review the source code as well before accepting them into the repo

Jin Kim: I will be coordinating the group project, and follow it through to meet the benchmarks we have listed. Also I will be building the code to present the correct simulation for accurate analysis.

Himakar Kadiri: I will be playing the role of motivator and helping anyone in my group to complete a task, so that we are up to date on the project. I will also be responsible for analyzing our simulation model and checking to see if it answers our questions. I will also make sure, everyone attends the team meeting that are set up and keep everyone engaged from the beginning of the project till the end.

Trystan MacInnes: I will be responsible for implementation of unique systems and explaining how the systems should work together. I’m also responsible for adding additional ideas in the case of simplicity in the project.

Technologies (1/2 page):

For this project, here are the list of the technologies each member will use:

Thuan Tran

I will use GitHub to host the group's source code. In addition, I will use PyCharm Education as my IDE, GitKraken as my GUI Git Client, Slack for communication and Google Drive to store our documents plus real-time interaction.

Himakar Kadiri

I will be using Github to save and update my source code. Furthermore, I will be using Enthought Canopy to write the simulation model in python. As for as communicating with the group, I will use Slack and google drive to store files.

Jin Kim

Use the Github to insert or remove the file to use for this simulation, to ensure all files are always up to date. I will use the Enthought Canopy to write the simulation in python, as required for this project. As everyone else, my primary choice of communication will be through Slack, whilst keeping the documentation on the google drive.

Trystan MacInnes

I will be using Github as well to add to the system. I will be using Enthought Canopy for coding. For group conversation I will be using slack and for group documentation I will be using Google Drive. The operating system used for testing on my end is Windows 10.

Benchmarks (1/2-1 page):

Milestone 1

Discuss the entities, and relations between entities. After all members understand the simulation, complete the class diagram through Lucid Chart, then once diagram is complete, finish the technical documentation for our future reference. All members should fork the github repo

Milestone 2

Complete Pseudocode, or all codes ready to run the simulation. All members should have created a pull requests to the repo and test the codes together

Milestone 3

Complete analysis and our finding.