Unit:
Designing and Developing Object-Oriented Computer Programs

Assignment title:
Card Stacks

December 2016

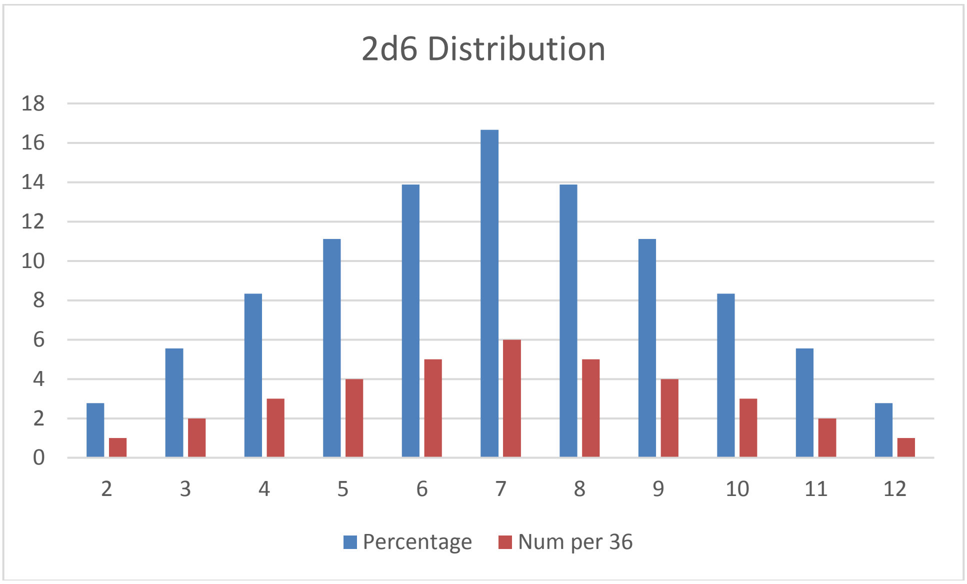
Important notes

- Please refer to the Assignment Presentation Requirements for advice on how to set out your assignment. These can be found on the NCC Education *Campus*. Click on Policies and Advice in the left-hand menu and look under the Advice section.
- You must read the NCC Education documents 'What is Academic Misconduct? Guidance for Candidates' and 'Avoiding Plagiarism and Collusion: Guidance for Candidates' and ensure that you acknowledge all the sources that you use in your work. These documents are available on *Campus*. Click on Policies and Advice in the left-hand menu and look under the Policies section.
- You **must** complete the **'Statement and Confirmation of Own Work'**. The form is available on *Campus*. Click on Policies and Advice in the left-hand menu and look under the Policies section.
- Please make a note of the recommended word count. You could lose marks if you write 10% more or less than this.
- You must submit a paper copy and digital copy (on disk or similarly acceptable medium). Media containing viruses, or media that cannot be run directly, will result in a fail grade being awarded for this assessment.
- All electronic media will be checked for plagiarism.

Introduction

Many games make use of dice to add an element of randomness to game rules. However, dice are often an unsatisfying system for adding variation because of just how dependant they are on the players' luck. Over a long enough period of time, the distribution of any correctly balanced dice will converge on the ideal. However, that period of time is often longer than that of a single play of a game. Some games have sought to resolve this issue through the use of a 'card stack'.

A card stack is a set of cards which contain the set of numbers for specific dice roles, in a statistically correct distribution. Consider the card stack for two six sided dice (known as 2d6 for short). This would contain the numbers 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. However, a 2d6 dice roll does not have an even distribution – you are more likely to get a seven than a two, and more likely to get a six than a twelve. This is a consequence of how many different ways the numbers can be made up by the pips on the face of the dice. The actual distribution of a 2d6 looks like this:



You have been asked by a local board-game shop to create a flexible card-stack application that can calculate the distribution of any given combination of dice, and keep a card-stack of that combination stored in memory. Whenever the user asks for a number of dice from a previously generated combination, a result should be dealt off of the stack and that number should be placed on a list of numbers which have previously been 'drawn'. When all numbers in the stack have been exhausted, the stack is reshuffled. When creating a card-stack, the user should also be able to specify how many cards should be 'removed' from the stack to eliminate the possibility of card-counting. These cards should simply be dealt off of the top of a randomly shuffled stack before numbers are generated.

A user will request a dice roll by providing dice notation in the form of <num >d<faces>-<cards to deal> Asking for one eight sided dice will be represented by the string 1d8. Four twelve sided dice minus two cards would be 4d12-2. Eight twenty sided dice minus six would be 8d20-6. Four four sided dice would be 4d4. Upon requesting a dice-roll, the application will check to see if it has previously generated a card-stack for that dice combination. If it has, it will provide the next card on that stack. If it has not, it will go through a four step process:

1. Calculate the correct distribution of results from that dice combination by 'rolling' them TEN THOUSAND (10, 000) times, and storing the results of how many times each combination was encountered.
2. Reducing those values down to the percentage chance of encountering each individual number.
3. Generating a card stack large enough to give the correct number of each result, to the accuracy of a tenth of a percentage point.
4. Shuffling that card stack and then removing the correct number of cards as per the user request.

A stack then is defined as its dice combination plus the number of cards initially removed from the stack. 3d6-2 should be a different stack to 3d6-4, and should be generated and dealt independently.

The application should also allow the player to 'peek' at a particular stack at any time to see what numbers will be generated in the future, and in which order, to ensure that the stack is being correctly generated and retained. Peeking should also show which cards have been removed, and which cards have already been 'dealt'. If desired, the player should also be able to 'shuffle' the stack at any time. The application too should maintain a graph for each stack, showing how often each number has been drawn from the stack and how many are left to be generated from each possible number. This should be regenerated every time a number is dealt, and should be displayed when the user peeks at a stack.

You will need to be sure that your distribution of numbers, and your testing should emphasise determining the correctness of the card stack you generate.

Your program then will need to perform the following operations:

- Set up the GUI
- Parse dice notation into the correct number of dice and faces.
- Generate card-stacks when requested, according to the process above.
- Reshuffle stacks when each of the 'cards' have run out
- Permitting the user to peek at stacks.

Task 1 – 50 Marks
50 Marks are available for a program which fully meets the requirements of the brief as outlined above.

Task 2 – 25 Marks
25 Marks are available for the selection of appropriate testing data.

Task 3 – 25 Marks
25 Marks are available for the provision of a fully detailed class diagram.

Submission requirements

- Your program must be submitted as a zip file of the full project.
    - Whatever IDE you use, it should be possible to open and run the project directly from the extracted archive.
- Your testing data must be accompanied with a short, 100 word discussion of how the data was selected and executed.

Candidate checklist

Please use the following checklist to ensure that your work is ready for submission.

Have you read the NCC Education documents 'What is Academic Misconduct? Guidance for Candidates' and 'Avoiding Plagiarism and Collusion: Guidance for Candidates' and ensured that you have acknowledged all the sources that you have used in your work?	❑
Have you completed the 'Statement and Confirmation of Own Work' form and attached it to your assignment? **You must do this.**	❑
Have you ensured that your work has not gone over or under the recommended word count by more than 10%?	❑
Have you ensured that your work does not contain viruses and can be run directly?	❑