Game theory with Java

Craps

Dice games have been around since the dawn of man. Craps is a derivative of an old English game called ‘hazard’. What is today’s game was brought to New Orleans in the 1800s and gained popularity quickly.

NOTE: Our labs will delve into games of chance. Typically, when games of chance are played, wagering is involved. The labs will not focus on this aspect of the game, just the mechanics of the game itself. I’m not going to teach you how to wager, odds, payout, etc.

The game itself is very easy to learn. One shooter (person throwing the dice) is picked from a group of willing players. The shooter will continue to shoot until the shooter shoots a ‘seven-out’. The dice are passed to the left, and another shooter takes their turn.

The shooter rolls the initial roll, or ‘come out’ roll. The sum of the dice will make one of the following combinations:

|  |  |
| --- | --- |
| **Sum of Dice** | **Result** |
| 2, 3, 12 | Craps |
| 7, 11 | Natural |
| 4, 5, 6, 8, 9, 10 | Point |

If a craps or natural is thrown, the round is over. If a point is thrown, the shooter is then trying on successive throws to match the same point. If the shooter throws a seven, the round is over. This is called a ‘seven-out’. If the shooter throws the point, the round is over- the player wins the round. Any other roll (not point, not seven), the player continues to throw. In theory, a single round of craps can last forever.

For example, the trials below:

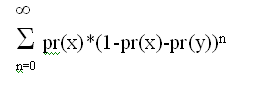
|  |  |  |
| --- | --- | --- |
| **Trial** | **Throws** | **Outcome** |
| 7 | 1 | Natural |
| 11 | 1 | Natural |
| 2 | 1 | Craps |
| 3 | 1 | Craps |
| 12 | 1 | Craps |
| 6-8-6 | 3 | Point / Win |
| 6-8-7 | 3 | Seven-out / Loss |
| 6-2-11-6 | 3 | Point / Win |
| 6-11-6 | 3 | Point / Win |
| 4-2-2-2-2-2-2-2-2-4 | 10 | Point / Win |
| 4-2-2-2-2-2-2-2-2-2-7 | 11 | Seven-out / Loss |

The probability for one trial is pretty easy to calculate. Consider that the summation of probability is 1, every combination of the dice (there are 36) must sum to 1.

|  |  |  |
| --- | --- | --- |
| **Outcome** | **# of possibilities** | **Probability** |
| Natural (7 or 11) | 8 | 0.22222 |
| Craps (2, 3, 12) | 4 | 0.11111 |
| Point (4, 5, 6, 8, 9, 10) | 24 | 0.66667 |

Assuming the each die is weighted so the throws are at random (1 to 6 are randomly thrown), the odds of calculating hitting a point can be tricky. As I’ve said before, in theory, the shooter can continue forever (if they don’t hit the point or seven-out).

Statistically, the probability of ‘x’ happening before ‘y’ can be represented with the following equation:



X = Probability of ‘seven-out’

Y = Probability of rolling your point

n = Number of trials

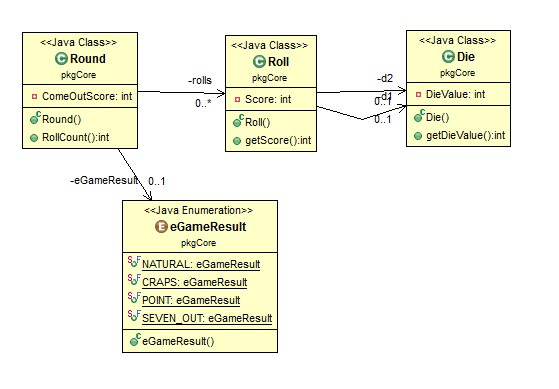
If you learn nothing else from statistics, learn this… ‘The Law of Large Numbers’. In simple terms, this means that as the number of trials approaches infinity, the result of the trials will closely math theoretical probabilities. If you toss a quarter in the air one million times, the results will be very close to 500,000 heads, 500,000 tails. At infinity, it’s exactly half.

How to complete the lab:

I’m giving you a running start… I gave you the scaffold of a working Java project. The project has the proper classes and contains zero errors.

Clone the following lab: <https://github.com/CISC181/Lab1>

1. You’ll have to change Die, Roll and Round class. Finish the code.
   1. Start with ‘Die’. Search for the //TODO: comments
   2. After Die, fix the ‘Roll’ class
   3. After ‘Roll’, fix the ‘Round’ class.
2. Finish the unit tests for Die, Roll, and Round.



Deliverables:

* Simple Java Project built on JDK version 1.8.
* Completed code for Die, Roll, Round and JUnit classes.

Grading – general guidelines/rubric

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Exemplary** | **Developing** | **Oh, come on!** |
| **Timeliness**  (25%) | Completed on time.  25 points | More than 1 day, less than one week late.  10 points. | More than 1 week late, less than 2 weeks late.  More than 2 weeks late – no submission possible.  0 points |
| **Knowledge of Content**  (40%) | Lab completed the bulleted deliverables, all functionality implemented, program(s) works as it should  40 points | Missed one deliverable  Example:   * Used JDK 1.7 instead of JDK 1.8 * Missed JUnit test case(s)   20-35 points | Missed more than one deliverable  0 – 20 points |
| **Coding- Design or Runtime errors**  (35%) | No errors, program compiles and executes as expected  35 points | No more than two errors  20-35 points | More than two errors  0-20 points |