Game theory with Java

Blackjack

Blackjack is a popular card game. One or more players are pitted against a ‘dealer’. The object of the game is to have a hand that:

* Sums to a score higher than the dealer
* Is as close a sum of 21 without going over.

Each number card is scored as the value of the card. For example 2 of clubs is scored as ‘2’. 5 of diamonds is scored as ‘5’.

Each face card is scored as ’10’. Jack of Hearts is a ‘10’, King of Diamonds is a ‘10’.

Ace can be scored as ‘11’ or ‘1’- depending on how it helps/hurts the hand. Example:

J – A = this can be considered as 11 or 21. You would want to consider this as 21, because 21 is a winning hand

5 – A = this is considered as 6 or 16. You might want to consider this as a 6 and take your chances drawing another card to get to a better total than 16.

5 – A – A = this can be considered at 7, 17. Technically it could be considered 27, but it doesn’t make any sense to score it as 27 because it’s a losing hand. This can be considered at 7 or 17.

Lab #2 has the following classes:

* Card – Changes required
* Deck – Changes required
* Die – No changes required
* Hand – Changes required
* Player – No changes required
* Roll – No changes required
* Round – No changes required

**Card**

Add a private attribute for eRank and eSuit

Add a constructor to build a card based on given eRank and eSuit

I’ve given you a ‘compareTo’ method- uncomment the //return statement

**Deck**

Add a private attribute called ‘cards’ that’s an ArrayList of Card object

Add a constructor to build a deck of cards

Add a method to ‘draw’ a card from the deck.

**Hand**

Complete the implementation of ScoreHand

**HandTest**

We need to test to make sure known hands evaluate correctly. We can’t leave it up to chance. We need to set the value for the cards… but this violates the concept that the cards are randomly assigned. I need to create a private method to set the card value and then add it to the hand using a special ‘AddCard’ method.

One way to run private methods is to use Java Reflections. Take a look at HandTest.HandHelper() method. I’m using reflections to dynamically invoke ‘AddCard’ and ‘ScoreHand’. This method is called by the ‘test’ methods.

Please implement the test methods I’ve called out. I’ve implemented ‘test1’ as an example.

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/Lab2Starter>

1. Make the changes in Card, Deck, Hand
2. Finish the unit tests in HandTest

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 1.8 instead of JDK 1.9 * 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 |