Problem 1 (20 points): Vowel or consonant?

Write a program that prompts the user to enter a single character from the keyboard and then checks if it is a letter and if so, determines if it is a consonant or a vowel. Here are some sample runs:

Enter a character: Z
Z is a consonant.

Enter a character: e
e is a vowel.

Enter a character: *
* is not a letter.

Call your file: Letters.java.

Note 1: You should read chapter 4 before attempting this problem.

Note 2: Think the problem through. The task of determining if the letter is a consonant or a vowel can be solved in many ways: some of them are much longer to implement than others (you should try to find a short way).

Problem 2 (30 points): Generating Cards

Write a program that generates a single card from a 52 card deck. The standard deck has 4 suits: spades, hearts, diamonds, clubs. In each suit there are 13 cards: ace, kind, queen, jack, and the number cards from 10 down to 2.

Your program should simply print the name of the suit and the card “number” of a randomly generated card.

Call your file: GetCard.java.

Note 1: You will be using random number generation in order to pick a random card.

Note 2: Your program should NOT have a huge if...else... or switch statement with 52 cases. There is a way of solving this problem with much fewer cases!

Problem 3 (50 points): Blackjack Game

Blackjack is a card game. The player with the highest score at the end of the game, as long as that score does not exceed 21 points. Write a program that gives the user a chance to play Blackjack. The human user is one player and the dealer is the computer (well, your program).

The simplified game goes along the following steps:

• Two cards are given and displayed to the user. (Your program needs to display the two cards. It also needs to store the value of these as score for the user.)
- Two cards are given to the dealer, but these are secret and not displayed to the user. (Your program still needs to store these values as the code for the dealer.)

- The user is prompted whether they would like an additional card. They can request as many additional cards as they would like, until they choose to "stop". If the value of the cards exceeds 21 at any point during this process, the user has "busted" and lost, and the game ends. If the user asks to get another card, the card should be displayed on the screen. (This should be done inside a loop and that continues as long as 1) user's score is not greater than 21, 2) user opts to get another card, 3-4) additional conditions from the next bullet.)

- The computer dealer decides whether to request as many additional cards as it wants. Your "silly" dealer always asks for two additional cards. If the computer dealer’s cards exceed 21 points during this process, the dealer has busted and lost, and the user wins. The card should be displayed on the screen. If the dealer asks for another card, the card should be displayed on the screen. (This should be done in the same loop as the previous bullet. It adds two more conditions to the loop continuation condition 3) the dealer opts to get another card, 4) the dealer’s score is not greater than 21).

- At the end of the game,
  - if either the user or the dealer "busted", then the other is the winner
  - if nobody "busted", then if the user's score is higher than the dealer’s score, the user wins, otherwise, the dealer wins.

Call your file: BlackjackGame.java.

Note 1: Do not worry about possibly generating duplicate cards. We will solve this problem in the future.

Note 2: You should be using the code that you develop for problem 2 in the solution of this program.

Note 3: You might want to use methods for this program (in fact, it would be desirable to include "getCard" as a method)

Grading

Does the program compile? If not, you will lose all the points for that problem.

Is the program properly documented? (worth ~30% of each problem)

Proper documentation includes:

- preamble with the name of the author, date of creation and brief description of the program;
- appropriately chosen variable names, i.e., descriptive names;
- comments inside the code describing steps needed to be taken to accomplish the goal of the program;
- appropriate formatting, indentation and use of white space to make the code readable.

Remember that the code is read by humans and it should be easy to read for people who were not involved in its development.

Is the program well developed? (worth ~40% of each problem) Make sure you create variables of appropriate types, use control statements (conditionals and loops) that are appropriate for the task, accomplish your task in a well designed and simple way (not a convoluted algorithm that happens to produce the correct output for some unknown reason). You should also design a friendly and informative user interface.

Is the program correct? (worth ~30% of each problem). Make sure that your program produces valid results that follow the specification of the problem every time it is run. At this point you can assume a "well behaved user" who enters the type of data that you request. If the program is not completely correct, you get credit proportional to how well it is developed and how close you got it to the completely correct code.

What and how to submit?

You should submit three source code files combined into a single zip file to NYU Classes. Do not submit all the files that Eclipse creates, just the source code files that have .java extensions.

Questions

Post any questions you have regarding this assignment to Piazza under the Homework2 tag.