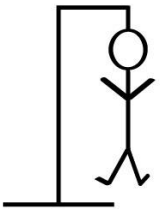
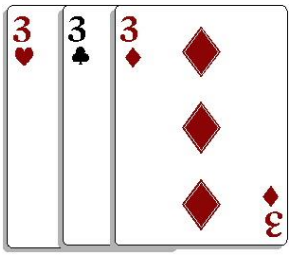


Department of Computer Science and CINSAM

NKU Summer Programming Workshop 2015



Project 10: We will walk through both of these projects and then you can implement the game of your choice. The card game is much more challenging, it is recommended that you do Hangman unless you are really understanding how to do methods, arrays and logic.

Hangman: Implement the code where there are comments. Specifically, create the words list, select a random word, get the word's size and fill in the correctGuesses boolean array. Next, fill in the if-else statement in the loop to respond correctly based on whether the user's guess is in the word or not. Next, implement these three methods:

- getGuess – first, create a String to that shows the user what they've guessed so for. For instance, if the word is "banana" and they have guessed 'a', they should see "-a-a-a". You will do this by taking the booleans in correctGuesses and creating a String where the letter at position i of the word is inserted into the String if correctGuesses[i] is true, or a '-'if correctGuesses[i] is false. Get the user's guess as a character and return it.
- convert – given the correctGuesses array and word, form the String as in "-a-a-a". This will be explained in more detail.
- fillIn – given the user's guess, c, the word, w, and the correctGuess array, alter correctGuess[i] to true for every character in word that matches c at position i. For instance, if the user has already guessed 'a' in "banana", then correctGuess is false, true, false, true, false, true. If they guess 'n', then correctGuess should be modified to false, true, true, true, true, true. This method should also return the number of times c appears in w. 2 in this example.

Enhancements are listed in comments at the bottom of the code.

Card game: a 3-card solitaire game, after getting their hand, the user can choose to replace 1 card and then the user can then choose to replace 1 more card (including the new card). The player then gets points based on the 3 cards:

A straight: 3 cards in sequence like 3 Clubs, 4 Spades, 5 Hearts (Aces are high only) – pays off $150
A flush: 3 cards of the same suit – pays off $100
3 of a kind – pays off $60 plus 3 times the sum of the 3 cards
A pair – pays off $20 plus 2 times the sum of the 3 cards
Nothing – pays off the sum of the 3 cards

The initial deal costs the user $30 ($10 per card). Each of the two discards costs $20 (the user can spend up to $70 in any one hand if the user discards twice). Cards are represented as integers where the face value is the int / 4 + 2 (e.g., 15 is 15 / 4 + 2 = 6) and the suit is int % 4 where where 0 is club, 1 is diamond, 2 is heart and 3 is spade. For instance, 15 / 4 = 2 is 6 and 15 % 4 = 3, so this is the 6 of spades. 50 / 4 + 2 = 14 (an ace, 13 is king, 12 is queen, 11 is jack) and 50 % 4 = 2, so 50 is the ace of hearts. All cards are worth their int value / 4 + 2 (6 is worth 6, aces are worth 14, kings 13, queens 12 and jacks 11). The player starts with $200 and continues to play until the user runs out of money or quits. Implement the following:

1. a method to generate a new card which has not been dealt yet in this hand
2. a method to generate a full hand of 3 cards
3. code for the user to replace a card and then replace another card
4. a method to convert an int value to a String that describes the card, for instance 15 is the 6 of spades (15 / 4 + 2 = 6, 15 % 4 = 3 which is a spade)
5. a method to determine the hand's value (straight, flush, 3 of a kind, pair, nothing), each has its own method where the method returns the dollar amount for the hand if the hand falls into that category (e.g., hand is a flush), or 0, for instance 3 of a kind returns 60 + 3 * sum of cards if the hand is 3 of a kind, or 0
6. a method to sort the cards (given to you)

A skeleton of the program is on the website. Read the comments carefully to see what needs to be filled in. NOTE: This is a text-only game.