## CS 110 Java Assignment Part 2 : Card Game of Boneace31

## NOTES:

- Collaboration policy for this program: This is individual work, but students may consult each other and TAs for this semester for discussing the approach. You must NAME significant collaborators at the head of the files.

- See the rules for the game. Anywhere in the document where there is a choice in the way the game is played, you decide what the rule will be. (Eg. whether an Ace is worth 1 or 11, or just 1, or just 11….)

- Additional specifications for the game:
  a) Your name must appear on the screen throughout the game play.
  b) When running, we should be able to see each play clearly and understand what is going on, and see changes to the game status such as the player's scores, what cards are in the player's hands, etc.
  c) In this game, the user is one of the players. This means that the user must be given opportunity to determine whether to "stick" or "have it", and generally interact with the game. For the purposes of debugging and viewing what is going on in the game, the hands are shown "face up" so we can see all the cards in all hands. (Deck is hidden).

- File Organization and what to turn in:
  Arrange files in **one** folder called **Boneace31.**
  Files to turn in: ONE zip file-- **Boneace31.zip**

  Within the Boneace31 folder, create 2 folders:
  **Console** folder:

  **Card.java**
  **Deck.java**
  **Hand.java**
  **Boneace31Console.java (main function to start the program here)**
  **cards (this is the folder with all the card images in it)**
  **AND any additional files you created that are required for compilation.**

  **GUI** folder:

  **Card.java**
  **Deck.java**
  **Hand.java**
  **Boneace31GUI.java (main function to start the program here)**
  **cards (this is the folder with all the card images in it)**
  **AND any additional files you created that are required for compilation.**

We will compile your code so make sure *all* source files are there so it will compile. This is very important and you will lose points if we need to get additional or different files from you later.

Ideally, the data model is completely separate from the view which means that the Card/Deck/Hand/Boneace31 classes should not need to change when switching from the console to the GUI. BUT just in case, it is advised that once the console version works, that you copy everything into the GUI folder and proceed to make the GUI version. This will eliminate problems with mixing files between the two versions.

---

Console View: (This version is a stepping stone and allows less than full points--see grading form)

You may use the card symbols for output: eg.

(char) 3 for the ♥ symbol
(char) 4 for the ♦ symbol
(char) 5 for the ♣ symbol
(char) 6 for the ♠ symbol

By the way, JGrasp handles characters differently and will not display these symbols. So use H, D, C, S instead. You can modify your toString method in the card and see how the view looks for the string.

GUI View: (This allows full points --see grading form)

Put **image files** in a **folder named "cards**" for your program to find them using the path. eg. "**cards**/imagename**.gif**".
These are provided in a .zip file : cards.zip
DO NOT change the names of these .gif files!!
MAKE SURE you have them in the correct path, and that your file name is called **cards**.

**********

The **game Boneace31** ADT: "**what**" needs to be accomplished?
This is where we actually play the game. These are some of the things you need to implement. You need to decide exactly how to "grow" this code in pieces, testing as you go. You are responsible for the final design as this is a very high level description. The data to be kept is really the same whether playing the console version or the GUI version. So we begin by implementing the game class: **Boneace31.java**

values: 2 players (each with a name, a hand, a score…)
should this be a separate Player class?

a deck (which you already have)

game status information: whose turn it is,
players names and scores…(perhaps 2 instances of Player?)

operations: initialize the game ( each player, the deck, the game status…)
showGameStatus
player1Play
player2Play
updateScore
boolean: gameOver

"**How**" to accomplish in concrete code:
class **Boneace31.java**
Decide how to store the game status, etc…and how to code each function.
Test the functionality first with an internal test program main as an intermediate test (which may not play the entire game), then build class **Boneace31Console.java**.

**********

**Boneace31Console.java**
Does the game class work? This class creates an instance of the Boneace31 class and plays the entire game, displaying the game execution to the console, and getting user player input when needed. In this version, you may use JOptionPane to get input from the user when needed.

**Boneace31GUI.java**
This version uses GUI interface with the card images provided.
(You may add the card .gif name to each card as a parameter in the Card constructor.) Try *not* to use the JOptionPane for user input.

**********