Lab 3

CS321L Winter 2023, Professor Christopher Diggins

## Overview

This lab involves modifying a Blackjack game created using classes. See: <https://en.wikipedia.org/wiki/Blackjack> for the rules of Blackjack.

The object of the game is to win money by creating card totals higher than those of the dealer's hand but not exceeding 21, or by stopping at a total in the hope that the dealer will bust. On their turn, players choose to "hit" (take a card) or "stand" (end their turn and stop without taking a card).

Face cards are worth 10, and Aces are worth either 1 or 11, depending on what is more advantageous for the hand evaluation. A hand evaluated with an Ace worth 11 is called a “soft” hand.

In our version of Blackjack:

* We will use a single deck of cards.
* Players cannot split.
* Dealer stands on a soft 17 or higher, or hits automatically.
* After the bet the player receives two initial cards
* Next the dealer receives one card face up.
* Then the user can choose what to do

Payout:

* If the player busts, they lose their bet.
* If the dealer gets blackjack, the player loses their bet.
* If the player gets blackjack, the player gets 2.5 \* bet.
* If the player hand has a higher value than the dealer, they get 2 \* bet
* If the player hand is the same as the dealer, they keep their bet
* If the player hand is less than the dealer, they lose their bet

The code provided has a few problems:

* All fields are public (they should all be private)
* The player can bet more money than they have.

Tasks:

* Make all fields private.
* Add new functions, constructors, and/or properties to allow the functionality to work as before.
* Prevent the user from betting more money than they have.
* Write a unit test that tests at least five different hands, and checks they have the expected value

Submission:

* The project file and source code of the game.
* Play three rounds of the game. Put the output in a text-file: output.txt.

Grading

* **2 points**: all fields are private, and the code still works.
* **1 point**: user can’t bet more money than they have.
* **1 point**: code follows the coding guidelines and is easy to read and understand.
* **1 point:** you have a unit test to test the value of five different hands.