Lab – Blackjack

# Summary

This lab costs of writing a Blackjack game. The rules of the game can be found at <https://en.wikipedia.org/wiki/Blackjack>

Please read the entire lab before starting any work.

# Team size

8 people

# Deliverables

1. A minimum viable product, consisting of the following:
   1. A working deck of cards, with normal deck operations (shuffle, draw, etc...)
      1. Think about what properties cards have
      2. For the MVP, Ace is always worth 11
   2. Two players, one of which is the "dealer"
   3. The ability to draw all player hands to the screen
   4. The ability to ask the player what they want to do
   5. A game manager, which drives the game logic:
      1. Each player gets two cards
         1. Player cards are both face up
         2. Dealer cards are one face up one face down
      2. Player gets to perform an action
         1. If they stay, their turn is over
         2. If they hit, they get another card, face up
         3. If the total of all their cards is over 21, they "bust" and lose
      3. Once player is done, dealer's cards both go face up
      4. Dealer must hit at 16 or below, must stay at 17 or above
      5. If dealer busts, player wins (as long as player is still in the game)
      6. If dealer stays, highest card total wins. Tie is called a "push" and nothing happens.
2. Once the MVP is done, a working blackjack game, playable by any number of players
   1. A “shoe” of cards, usually made up of six decks
   2. Betting
      1. Players should have money
      2. Players should be able to bet before they get cards. If they win, they double their money. If they lose, they lose their bet
   3. Ace can have a value of 1 or 11, based on the player’s choice
   4. BlackJack hand (Ace and 10, J, Q, or K)
      1. If the dealer has blackjack, everyone at the table loses, unless the player has blackjack also, in which case this is a "push" and nothing happens.
      2. If the player has blackjack and the dealer doesn't, they usually get more than double their money. See the rules.
   5. Special player actions (look these up):
      1. Double down
      2. Split
      3. Buying insurance
3. A sprint retrospective, which will be shared with the class

# Requirements

## Form a team

* Form a team based on the team size for the lab. Come up with a team name
  + Try to work with different people than you have before
* Send me your team name using Skype
  + I will create an Area Path and an Iteration Path for your team in VSTS.
  + The Area Path will be Cohort#\Labs\<LabName>\<YourTeamName>
  + The Iteration Paths will be
    - Cohort#\Backlog
    - Cohort#\Iteration 1

## Perform sprint planning

Don’t rush through this, it should probably take about 30 minutes.

1. Think about what user stories are required for the game to work. Use VSTS to create work items for each user story.
   1. The Area should be Cohort#\Labs\<LabName>\<YourTeamName>
   2. The Iteration should initially be Cohort#\Backlog.
   3. This Area/Iteration combination is your product backlog.
2. Break down each user story into tasks. Use VSTS to create work items for each task, using child tasks as appropriate.
   1. You should be discussing high level tasks, not implementation details.
   2. Fill out rough time estimates for each task as a team.
3. Write a query that gets a tree view of the user stories in your team’s area, along with all their child tasks, sorted by priority ascending.
   1. Save this query as a shared query so anyone on your team can run it.
4. Once the query works, prioritize the user stories, and tasks within the user stories.
   1. Remember, Agile is about delivering something that runs (even if it’s incomplete), then iterating on it.
5. Decide on your minimum viable product (MVP), and change the iteration for those tasks to Cohort#\Iteration1. This is your sprint backlog.
6. Distribute the tasks amongst your team.

## Create team branch and automated build

1. Browse to <https://microsoftleap.visualstudio.com/Cohort7/_git/Chess/branches>
2. One person should create a new remote branch from master that matches your team name. This will be your team branch
3. One person should create a build definition for your team branch. Set it to build for every commit into your team branch. Kick off the build manually to make sure it works before moving forward.

## Create personal branches

1. In VSTS, each team member should create their own remote personal branch from the team branch. Name it something like <yourAlias>-dev
2. Each team member should clone the remote repo, which will get you a local master branch.
3. Each team member should create a new local branch from their personal remote branch. Give it the same name as your personal remote branch.

## Execute the sprint

1. Each team member should commit work into their local branch and push it to their personal remote branch.
2. When ready, each team member should create a pull request from their personal remote branch into the team branch.
3. The team should perform code reviews to make sure the team branch stays healthy.
   1. The team branch should always be “shippable code”!

# Code requirements

1. All code should be contained in classes.
   1. The main method should only instantiate the game class and start the game with game.Start() or something similar.
2. All classes and functions should be small and reusable whenever possible.
   1. Each class should have a single responsibility
   2. Each function should do one thing
3. All code should be very well commented. At a bare minimum:
   1. Each class should have XML style comments describing what it does.
   2. Each function should have XML style comments describing what the function does, the inputs, and the outputs
4. All functions (except the game loop) should have accompanying unit tests.
   1. Test normal cases, edge cases, and exceptions (if you use them)!