Kent Stringer

Dylan Shoupe

Dylan Terry

**NIM Requirements Document**

Requirements:

* Language
  + C#
* OS
  + Windows
* UI/UX
  + Command Prompt

Functional:

* Player(s) must remove at least 1 tile and up to the total number of tiles in single heap each turn
* Instructions Page available at all times to the user
* Random starting player
* Basic AI
  + Random legal moves
* 2 Game Modes
  + Player vs Player
  + Player vs AI
* User is able to enter custom name(s) for each human player before each game
* 3 Difficulties
  + Easy
    - 2 heaps (3/3)
  + Medium
    - 3 heaps (2/5/7)
  + Hard
    - 4 heaps (2/3/8/9)
* Multiple Games per session possible
  + User is required to choose game mode
  + Enter human player name(s)
  + Select difficulty
  + Continues until user elects to exit program
* Player who draws the final tile loses the game
* Ascii art heap display
  + Will include text stating remaining tiles
* Player moves must be validated for legality

External Specifications:

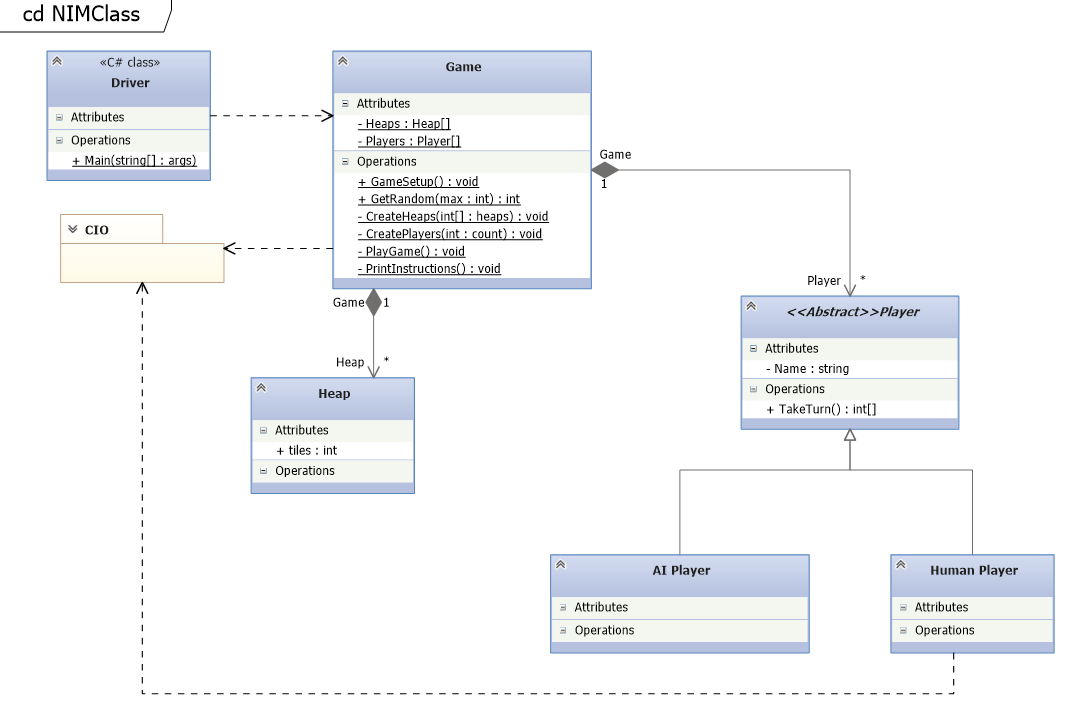
* User will be able to launch program from an exe file
* User will play game in the command prompt window
* User will see some color displayed in the console
  + Ascii Art
  + Winner screen
* User will input commands through the keyboard
* User can exit at anytime
* User can see the instructions at anytime

Dead line:

* Friday, November 10th 2017 by 3pm

**Design**

UML:



Flow Diagrams:

Developer Docs:

* Classes
  + Driver
    - Will call GameSetup()
  + Game
    - Players : Player[2]
      * Class level variable that will hold two instances of Player
    - Heaps : Heap[]
      * Class level variable that will be of variable size based on the difficulty chosen by the player
      * Easy = 2, Medium = 3, Hard = 4
    - GameSetup() : void
      * Will have a do, while loop that loops until the user decides to exit the game
        + Will use C# library “CIO” to prompt the user for a menu selection

First menu will ask the user if they want to Start the game, see the instructions, or quit

Second menu will ask the user if they wish to play against a computer or another human player, see the instructions, go back, or quit

The Third menu will ask the user to choose a difficulty, see instruction, go back, or quit

Once the user has chosen a difficulty the createHeaps() method will be called and will be passed an int array based on the choice of the user. Easy {3,3}, Medium {2,5,7}, and Hard {2,3,8,9}

* + - * + Will call CreatePlayers() then will call PlayGame()
        + After PlayGame() has completed the user will be asked if they wish to play again. If the answer is no, the loop will end and the program will close. If yes, the loop will continue from the beginning.
    - PlayGame() : void
      * Int playerTurn will be determined by calling GetRandom(10)
      * Players array will use playerTurn mod 2 as the index value to determine turn order and the active player.
      * Will have a do, while loop that will continue until the end of the game or until the user has chosen to exit the game in mid game.
        + Inside the loop the takeTurn() will be called from the Players[] utilizing playerTurn % 2.
        + After takeTurn() completes, If all heaps added together equal 0 the loop will end with the opposite player being declared the victor. If it does not equal 0 the playTurn variable will be incremented by 1.
        + This loop will continue until all heaps added together equal 0 or the user opts to return to the main menu
    - CreatePlayers(int count) : void
      * Parameter “count” will determine the number of human players that will be created.
      * Given 1 human player the method will generate 1 AI player.
      * Each human player will be asked to provide their name
        + A default name will be given in the case of a NULL value
      * No more than 2 players can be created in a single game
    - CreateHeaps(int[] heaps) : void
      * Parameter “heaps” will be an array of int that will be utilized for to fill the class level variable Heap[] with the appropriate number of heaps and the correct number of tiles per heap.
        + Using initialization syntax, Heap[] will be created using the integers provided by the parameter “heaps” to create a new Heap instance passing in the int value for the tile amount of that Heap.
    - PrintInstructions() : void
      * Method will use console.writeline() to print out the instructions of the game
      * This method must be available during any player turn, during the game, and before starting any new game
    - GetRandom(int max): int
      * Will generate a number between 1 and the max value given
      * This method will be used to determine the turn order
      * This method will be used by the AI class to randomly choose a heap and a value to be taken.
* Models
  + Abstract Player Class
    - PlayerName : string
      * Property Value
    - TakeTurn() : int[]
      * Will be implemented by Human and AI class
  + Human Model (inherits parent player class)
    - TakeTurn(): int[] – Implemented from the Player class
      * Use C# Library “CIO” to get two int values from the user.
        + First value will the heap number
        + Second will be the number of tiles they wish to remove
      * This prompt will loop until the user has given valid input for both ints
      * Method will return an int array containing their heap choice and the number of tiles to be removed (choices{1,7})
  + AI Model (inherits parent player class)
    - TakeTurn(): int[] - Implemented from the Player class
      * Will call GetRandom() passing in the number of heaps
        + The number returned from GetRandom() will be validated to ensure the heap has any remaining tiles
        + This method will be called until a valid heap has been chosen
      * Will call GetRandom() passing in the number of tiles left in the chosen heap
      * Method will return an int array containing their heap choice and the number of tiles to be removed (choices{1,7})
  + Heap Model
    - Int tiles
      * Property value will initially be set by the constructor

UI/UX Templating:

* See attached UIUXWriteup.txt file