Melissa Welsh
mwelsh2@u.rochester.edu

NOTES:

- For the 8x8 game:
    - h-minimax without pruning (method 4) works reasonably fast (~2.7 secs for first move) at a depth cutoff of 8
    - h-minimax with pruning (method 3) works reasonably fast (~1.5 secs for first move) at a depth cutoff of 12
- don't believe that the game is playing perfectly (seems too easy to beat)
- multicapture feature is not implemented
- Project must be run from the Run class in the main package.
- Project is built in Eclipse, all source files included

PROJECT STRUCTURE:

PACKAGE: board

-- Board
  -- initializes a Checkers board given a certain size
  -- includes important methods like **`getTileBetween()`**
-- Tile
  -- represents a tile on the board, stores diagonal neighbors

PACKAGE: checkers

-- CAction
  -- definition of an action **`a`** in Checkers is 4-tuple
  1. Type (***`move`***, ***`capture`***, ***`multicapture`***)
  2. Tile **`i`** : represents the starting tile of the action
  3. Tile **`j`** : represents the ending tile of the action
  4. LinkedList<Tile> **`captures`** : represents the list of captures if applicable
-- Checkers
  -- implements the interface Game in the game package
  -- includes the **`ACTION(s)`** , **`RESULT(s, a)`** , **`TERMINAL(s)`** , **`UTILITY(s)`** , and **`heuristic(s)`** functions
  -- **`heuristic(s)`** subtracts the opponent pieces from the player's pieces
-- CPlayer
  -- a Checkers player is 2-tuple
  1. Color (***`WHITE`***, ***`BLACK`***)
  2. Minimax Value (***`MAX`***, ***`MIN`***)
-- CState
  -- a Checkers state s is 4-tuple
  1. Set<Tile> **`Lb`**: Location of all black pieces on the board
  2. Set<Tile> **`Lw`**: Location of all white tiles on the board
  3. Set<Tile> **`K`**: Location of all kings on the board
  4. CPlayer **`turn`**: Player whose turn it is to move
  -- includes many important methods such as **`createInitial()`**, which creates the initial state of the game, **`checkCapture()`**, which checks if a capture is possible and

`getTurnPieces()`, which returns all pieces of player whose turn it is

PACKAGE: game
-- AlphaBeta
    -- contains a method for heuristic minimax with a fixed depth cutoff and alpha beta pruning
    -- when playing 4x4, the depth cutoff is automatically set to infinity
    -- implements Search interface
-- Game
    -- general interface for a two player, zero sum, perfect information game
-- hMinimax
    -- contains a method for heuristic minimax with a fixed depth cutoff
    -- recommend using a depth cutoff of maximum 8 if playing 8x8
    -- implements Search interface
-- Minimax
    -- contains a method for plain minimax, will not work (in reasonable amount of time) if playing 8x8
    -- implements Search interface
-- Random
    -- contains a method for randomly choosing the next move
    -- implements Search interface
-- Search
    -- general interface for adversarial search
    -- method **`chooseMove()`**, takes an input State **`s`** and returns an Action **`a`**

PACKAGE: main
-- Run
    -- PROJECT MUST BE RUN FROM THIS CLASS
    -- contains **`quit()`**, **`restart()`**, **`loseGame()`**, **`tieGame()`**, and **`winGame()`** methods
    -- contains **`gameInit()`** method to get user input for game initialization
    -- contains **`playGame()`**, method that keeps the game running on a while loop such that no terminal state is reached
    -- contains **`main()`** method