Assignment 2: Supplemental Materials

1 Solution Checklist

The following list gives you some hints as to what you should make sure that your program covers. Note that this list may not be comprehensive. Make sure to closely read the assignment sheet.

- **File I/O mechanism related issues**
  - ☐ Make sure your program runs fine no matter whether the `players.dat` file exists.
  - ☐ Make sure every `exit` command triggers data to be written to the `players.dat` file.
- **Polymorphism related issues**
  - ☐ Make sure the AI player is implemented using inheritance mechanism.
  - ☐ Make sure you are leveraging the polymorphism to invoke the methods, i.e., using object declared in parent class to invoke methods overridden in the child class.
  - ☐ If attempting for bonus marks, make sure the advanced game is implemented using either inheritance or interface mechanism
- **Victory guaranteed strategy related issues**
  - ☐ Make sure your AI player can play without errors whether it plays as the first one or second one to move.
  - ☐ Try your best to win as many test cases as possible on the submission system with your AI player.
- **Submission issues**
  - ☐ Make sure there is only **one Scanner** object throughout your program.
- **Error handling issues**
  Only the following errors need to be handled (you may choose to handle more if you wish):
  - ☐ Adding a new player with an existing username.
    Error message: The player already exists.
    Follow-up operation: Print '$' and prompt for user input again.
  - ☐ Removing a player with a non-existing username.
    Error message: The player does not exist.
    Follow-up operation: Print '$' and prompt for user input again.
  - ☐ Resetting the statistics of a player with a non-existing username.
    Error message: The player does not exist.
    Follow-up operation: Print '$' and prompt for user input again.
  - ☐ Displaying a player with a non-existing username.
    Error message: The player does not exist.
    Follow-up operation: Print '$' and prompt for user input again.
  - ☐ Starting a game where at least one of the player's username does not exist.
    Error message: One of the players does not exist.
    Follow-up operation: Print '$' and prompt for user input again.

- ☐ Removing stone outside of the range [1, `stoneRemovalUpperBound`], where `stoneRemovalUpperBound` is the maximum number of stones allowed to be removed in one turn.
  Error message: Invalid move. You must remove between 1 and `stoneRemovalUpperBound` stones (you do not need to consider the singular or plural form of "stone").
  Follow-up operation: Prompt for the same player to remove stone again.

- **Blank line and whitespace related issues.**
  - ☐ Make sure that between the output of last command (including indication for nonexistent users, confirmation for all-user operations, display results, and game results) and the next command prompt, there is one blank line.
  - ☐ Make sure that there is a whitespace between `stones left :` sentence and `* * *`, and also between all `* * *`, but NO whitespace at the end of `* * *`.
  - ☐ Make sure that there is a whitespace between `(y/n)` and `Are you sure...` sentence.
  - ☐ Make sure that in game state, there is one blank line before the `Initial stone count...` and one blank line after the `Player 2:...`.
  - ☐ Make sure that in game state, if a user inputs an invalid move, there is one blank line between the user-input move and the indication sentence `Invalid move...`.

- **Ranking display related issues.**
  - ☐ Make sure that the ranking is in a descending order of winning ratio by default or when "desc" is provided as an argument.
  - ☐ Make sure that the ranking is in a ascending order of winning ratio when "asc" is provided as an argument.
  - ☐ Make sure that winning ratio is rounded to the nearest integer value when displaying the winning ratio. However, please note that exact values are still used when comparing and sorting two users' winning ratios.
  - ☐ Make sure that users with the same winning ratio are sorted according to the alphabetical order of the user name irrespective of the argument supplied to the rankings command, e.g., if username lskywalker and username hsolo have the same winning ratio, you should rank hsolo higher.
  - ☐ Make sure that the first and the second column strictly have 5 and 10 characters, respectively.

- **Display player related issues.**
  - ☐ Make sure that the displayed list is sorted in an alphabetical order of the username.

- **Game related issues.**
  - ☐ Make sure to correctly update the statistics for players when a game ends.
  - ☐ Make sure to check valid move by comparing the move to the smaller one between the current number of stones and the upper bound for one move.
  - ☐ Make sure that after an invalid move, it is still the turn of the player who made the invalid move.

- **Command line prompt related issues.**
  - ☐ Make sure that the command prompt appears again after each command is issued (except the `exit` command).
  - ☐ Make sure that only one command prompt is displayed after a game is over and the system returns to the idle state.

- **Other issues.**
  - ☐ The maximum number of players can be set as 100.

☐ The branching statement `switch` may not support `Strings` on the submission server.

☐ The boxes enclosing the above example executions are NOT to be printed out, they are just for better illustration.

☐ The winning ratio of players with 0 games is 0.

☐ The maximum number of stones in each game is assumed to be 11 for the Advanced Nim game.

☐ `Collections` such as `ArrayList` are acceptable.

☐ Players are by default sorted according to the lexicographical order of the usernames.

☐ Usernames can be assumed to be all lower-cased.

☐ "*What does it mean when lastMove is a blank String/null in the advanced Nim game?*"
It means that there is no last move and your `NimAIPlayer` object is to make the first move.

☐ "*My code works perfectly on my machine but it does not get any winning ratio in the last test.*"
First, check the FAQ items above and make sure you have handled each of them. Second, make sure that you do not change the Boolean array input parameter in `advancedMove()`. We just need your move returned as a String, and we will update the Boolean array.