Code Swap Review

Team 105 needs to pick a code base to use for our project. There are 5 options to choose from. We used a 5 part metric to decide which code base to use that augmented Prof. Alphonce's instructions.

-But does it compile?
-Design of the code (modular)
-Easy to follow code
-How well documented
-How well unit tested it is

Candidate A: Code base compiles. Code is confusing to read due to poor design decisions. Programmers appeared to be ArrayList-happy when they wrote this code. Most things are ArrayLists despite not being the optimal choice. GUI is functional, and aesthetically pleasing. Due to almost non-existent Javadocs, code is not easy to follow. Many tests covering a variety of methods and situations.

Candidate B: This code base fails to compile when run. Due to the time constraints of the project, time to debug badly written strangers' code is non-existent. Therefore this code base was eliminated immediately.

Candidate C: Code compiles. Methods are Javadoc'd at the start of the method. This describes why the method was written, but there are no comments inside the methods describing what is actually happening. However, window color is hard to read. The way it initializes the board is wrong.(It fills the board based on the order of TwoDirection, Straight, and ThreeDirection, not randomly.) Code is also inefficient because of the design causing lags in game play as the board is shifted and the images redrawn on it. Code is mostly unit tested.

Candidate D: Code compiles. Code is not very modular and there are no Javadocs in GUI part. When the game is run there are a lot of errors. The player's name doesn't show up when the game is run. Multiple exceptions are thrown in the program. When a tile is inserted there are errors that occur.

Candidate E: Code base compiles. Code is not optimally modular, but some of it can be reused. GUI is functional, and is aesthetically pleasing. Appears to have all the functionality that was required at the end of stage 2. Appears to have good JavaDocs and explains code well. Code appears to be designed and written well and looks to be modular. There are some less than optimal choices that were made regarding rotation of the tile. Code has many tests that check to see if code works for many situations for each method. Has the most unit tests of any of the methods.