Homework 1 – Due Mar. 14th 23:59, KST

Instructions: Complete the program and turn it in before the due date. Any deviations from the instructed deliverable format will result in a deduction of grade. No late submission for this assignment will be accepted. DO NOT COPY OTHER'S WORKS!

As your first programming assignment, you will create a very simple game engine. There's nothing fancy about this game engine, as you will be responsible for implementing the simplest operations only. The purpose of this assignment is to help you understand what it means to manage data in an OOP fashion without the help of existing data structures. As such, you must not use any external packages, and doing so will result in a zero credit.

Your task is to manage a game played on a grid. The number of rows and columns of this grid will be given as input to the constructor. On this grid, many players will be moving around through a non-diagonal single-step movement. That is, each player can move to one of four locations: up, down, left, and right. Wrap-around movements are also possible, meaning that moving right from a right-most grid will allow the player to appear on the far left. Example moves on a 5x4 grid are shown below.

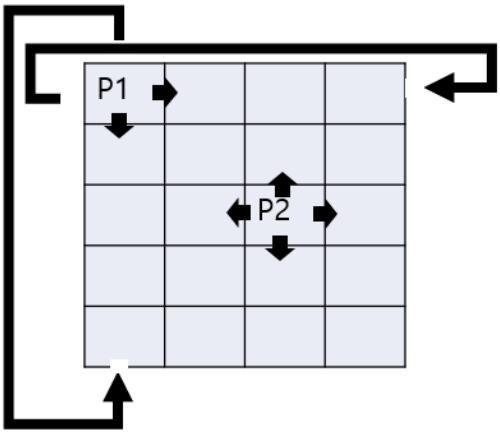

Figure 1 Player P1 can 'wrap around' the grid by moving up or left.

It's not shown in the example above, but players can 'capture' each other by moving into an already occupied grid. In that case, the original occupant is captured by the player moving in, and the former will automatically be removed from the game. Just think of how a game of chess is played.

The grid coordinates are specified as (r, c), where 'r' and 'c' are zero-based indexes of the row and columns, respectively. So P1's coordinate is (0,0) and P2's is (2,2).

In addition to these movement related tasks, you should also implement several housekeeping

operations, including insertion, removal, and retrieval of players and locations. Please refer to the GridGame.java file for full list of methods you are required to implement. In particular, carefully read all the 'TODO' comments in the code as they are also part of the official instructions.

You will be graded on correctness of your code (60%) and documentation (40%). The former states that all of your implementation must exhibit correct behavior as stated in the comments, and the latter requires that you clearly describe your implementation logic. Write a block of comment on top of each required method to describe your approach in detail. Obvious comments that merely translate the code will not be counted (e.g., "I declared a variable and set it to 0. Then I enter a for loop running until 'i' becomes a.length…. etc.").

Deductions

- Every unhandled exception and error will be worth -10 points. 3 or more such errors will give you a score of 0 for correctness.
- Importing and declaring a package: -10 points each, even if you don't use an imported package. You should habitually double-check your code before submission.

Deliverables

- A single GridGame.java file containing your name and SBU ID in the top-most comment. Do NOT use any package structures (i.e., no package declaration in the top).