CS102 – Algorithms and Programming II
Lab Programming Assignment 4
Fall 2022

ATTENTION:

- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:

  CS102_Sec1_Asgn4_YourSurname_YourName.zip
- Replace the variables "YourSurname" and "YourName" with your actual surname and name.
- You may ask questions on Moodle.
- Upload the above zip file to Moodle by the deadline (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TA during the lab. You will resubmit your code once you demo your work to the TA.

GRADING WARNING:

- Please read the grading criteria provided on Moodle. The work must be done individually. Code sharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The Honor Code specifies what you can and cannot do. Breaking the rules will result in disciplinary action.

Treasure Map

You are going to implement a Java program for following paths in 2D treasure maps. Each path is going to be stored using a set of instructions. Each instruction consists of the direction (North, East, South, West) and an integer indicating the number of units to move along that direction.

Map class should keep the treasure map as a 2D array. The maximum map size can be 200 by 200. Each map cell is in one of the four types: **walkable** (empty), **non-walkable** (block), **starting point**, or **treasure**. There can be multiple treasures on a map, but there is only one starting point. All the paths are relative to the given starting point. Include a **readMap(string filename)** method in the Map class to initialize the map based on the input text file. The letters in this txt file would indicate the following cells:

B: Block, non-walkable cell.
.: Empty, walkable cell.
S: Starting point, only one for each map, walkable cell.
X: Treasure, walkable cell.

For example, a 6 by 4 map file could contain the following lines:

```
BBB...
B...B.
BX..S.
..B.X.
```

We want to store a bunch of paths in the **Map** class. Include a **void createPath(string filename)** method to the **Map** class to create a path from the instructions included in a text file. Each line of this path text file would contain the letter of the cardinal direction (N, E, S, W) and the number of units to move along that direction, separated by a space character. The number of units to move is an integer that is in the range [0, 200].

For example, a path file could contain the following lines:

```
E 1
N 3
W 2
S 1
W 2
S 1
```

Using the map given above, starting from the cell S, this path ends on a treasure cell.

Include a **boolean isPathToTreasure(Path p)** method to decide whether a given path reaches a treasure or not. To this end, you need to follow the instructions of the path in order. Any instruction that cannot be performed is ignored. For example, if the instruction says N 5, but there is a non-walkable cell at 2 units north of the current cell, then you should skip this instruction and continue with the following instruction. (We either follow the instruction completely or ignore it completely, we cannot follow it partially if there is a blocking cell, in this case we completely ignore it.) When all the instructions of a path are processed and we reach a treasure cell, return true. Otherwise, if we did not reach a treasure cell at the end, return false.

Include a current position variable in the Map class to simulate movement on the map. Note that you need to keep both x and y coordinates to locate a cell in the 2D map. Include a **boolean isMovePossible(Instruction i)** method in the Map class to check if a given instruction is applicable based on the current position. This method should consider the non-walkable cells and the map limits while determining if the given instruction is possible to perform. Note that we can only move along the cardinal directions, diagonal movement is not possible. Make use of this method in your **isPathToTreasure** method to skip invalid instructions.

You will be given a folder of path files to test your program. Include a **void readFromFolder(string folderName)** method in your Map class to read all the path files included in the given folder to create Path objects out of them using your **createPath** method. You should also store the filename of each path, as we will use their filenames while reporting which paths end up in a treasure cell.

Include a void **processPaths()** method to test all the paths kept in the Map class. Report the filenames for the paths that end up in a treasure cell. You will use the **isPathToTreasure** method for this task.

Include a **boolean checkPathCombination(Path a, Path b)** method in the Map class. This method should follow the path that is formed by appending the instructions of path B to the instructions of path A. It will function similarly to the **isPathToTreasure** method, skipping any instruction that cannot be performed, and returning true if we end up on a treasure cell. Only difference is that we will test the combined path.

Finally, include a void **processPathCombinations()** method to test whether any combination of two paths stored in the Map class can reach a treasure or not. To this end, you will use the **checkPathCombination** method for each stored path pair. Note that this method is not symmetric, checkPathCombination(a,b) and checkPathCombination(b,a) may have different results as the map includes non-walkable cells. Print out the filenames of path pairs that can reach a treasure cell.

Example files for map and paths are included under the Moodle submission page for you to experiment with.

Preliminary Submission: You will submit an early version of your solution before the lab. This version should at least include the following:

- The following methods should be functional:
    - readMap
    - createPath
    - isMovePossible
    - isPathToTreasure.

You will have time to complete your solution in the lab. Do not forget to make your final submission at the end. Even if you finish the assignment in the preliminary submission, you should submit for the final submission on Moodle.

Not completing the preliminary submission on time results in 50% reduction of this assignment's final grade.