MEMORIAL UNIVERSITY
Faculty of Engineering and Applied Science

ENGI 9818 Computer Software Foundations

Fall 2023

Assignment 0

Warning

Zero-tolerance on academic dishonesty. Academic dishonesty will, as a minimum, result in a mark of zero for the offending work. Academic dishonesty will be reported to the Associate Dean. Academic dishonesty includes copying from another student's work or allowing another student to copy from one's own work, consulting with any unauthorised person during an examination or test, using unauthorised aids (including AI, e.g., GPT-derivatives), and presenting the ideas or works of another as one's own. If you have any doubt about whether something constitutes academic honesty, consult with the instructor.

Question

We have a maze:

		W		
	E	W		
	W	X		

E represents the entry; W represents impassible walls and X represents the exit of the maze. Your goal is to write a function “escape”, to escape the maze.

Coordinate has the format of (row, column). In this example, the top-left corner has the coordinate of (0, 0), while the bottom-right corner has the coordinate of (3, 4).

Question 1

Write the interface contract as Javadoc in your code. The “return” is given.

```
...

 * @return A list of coordinates, from the exit point to the entrance point. The
first item of the list is the coordinate of exit point while the last item is the
coordinate of the entrance point.

...
```

Question 2

Follow the given template to implement the “escape” function and corresponding Junit 5 tests.

Submission

It is extremely important that you are following the naming conventions for your class, tests, files, and submissions. If the automatic system fails to recognise your submissions or failed to execute them, your assignment will be treated as NOT SUBMITTED.

Late submissions are NOT allowed and will NOT be accepted.

Requirements

1. Your submission must be a zip file and the file name must be your student number. Example: 202312345.zip
2. Your submitted zip file must contain two files, “EscapeImpl.java” and “EscapeTest<student number>.java”. The structure shall like:
   ```
   202312345.zip
         |- EscapeImpl.java
         |- EscapeTest201312345.java
   ```
3. The “EscapeImpl.java” file must include a public class, “EscapeImpl”, which implements the “Escape” interface.
4. The “EscapeTest<student number>.java” file must include a public class “EscapeTest<student number>”, which implements your tests. Minimal 3 tests are required.
5. You can change anything in “EscapeImpl.java” and “EscapeTest201312345.java” in the published example package.
6. No third-party libraries or dependencies.
7. Your code must be working with JDK 17 and up.
8. The “App.java” file include a basic example. Your program shall be, at minimal, working with the example.
9. You must use recursion to implement your algorithm.
10. Code style is important. You need to follow Java naming conventions and ensure there are no unnecessary extra new lines, spaces (e.g., additional spaces at the end of lines, a line with only spaces), etc.
11. Do not declare packages. You must use the default package.

Hints (you don’t have to read or follow this section)

1. We just need a path to get to the exist, it doesn’t matter the path is the shortest or not.
2. By using recursion, it means we are constructing the path to go from the exit to the entrance.
3. With the example maze given in the question, one of the valid output the output could be:
   `[(2, 2), (3, 2), (3, 1), (3, 0), (2, 0), (1, 0), (1, 1)]`
4. You can use the “testPath” function in the template to validate if a path is valid or not for your test. But this function is not perfect and has known bugs. You are free to change it.
5. The “escapeImpl” function given in the template can also be changed.
6. Think step by step:
   a. check the type of current cell, is it inaccessible? is it an Exit? is it an Entrance?
   b. If possible, try to go up, left, down and right.
   c. Beware of circling in the maze
7. When in doubt, debug your code step by step to understand how it runs.