Programming Techniques, 2022-2023

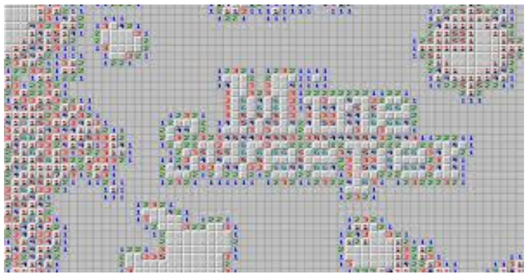
Ludo Buynseels, Stijn Langendries, Jeroen Wauters, Nianmei Zhou, Koen Pelsmaekers.

Lab session 4-6: Minesweeper

Goal

The goal of lab session 4-6 is to learn how to use **inheritance** and **dynamic method binding** in a more complex application. You will implement a minesweeper game using text-based user interface. This exercise has three parts: (1) UML diagram, (2) implementation, and (3) test code.

Rules of the game



The goal of the game is to locate all the mines on a board with tiles. The game is won when all the tiles that are not mines are opened. The game is lost when opening a tile that contains a mine. If you have never played minesweeper before, try it at https://minesweeper.online.

a. Game mechanics
A player can open a tile (left click on a GUI):

- First tile rule: The first click on a board *never hits a mine*. If there is a mine on that position, you should move it to another location without a mine and update the board.
- Hit a mine: If this is not the first click, when the opened tile contains a mine, *the game is over (lost) and all mines are shown*;
- Numbered tile: When the opened tile contains no mine but has at least one mine in the adjacent eight (or less) tiles, a number indicating *the number of mines in the valid adjacent tiles* is shown on the tile;
- Empty tile: If the opened tile is not a mine and is not adjacent to any mine, we call it an empty tile. When it is opened, *all its adjacent empty tiles are opened automatically until reaching a numbered tile or a mine*.

A player can also flag and unflag a tile (right click on a GUI). Flagged tiles cannot be opened; when a player tries to open a flagged tile, nothing happens.

- If the tile shows no flag, a flag appears over the tile (this is an intent to mark a tile as a mine)
- If the tile shows a flag, the flag will be removed.

b. Generate the board

You can customize the minesweeper game by giving the size of the board (i.e., row and column) and the number of mines. You can also generate a minesweeper game according to a selected difficulty according to the examples below:

- Easy: 10 mines on an 8-by-8 board
- Medium: 40 mines on a 16-by-16 board
- Hard: 99 mines on a 16-by-30 board

Assignment

The assignment of lab 4-6 is to develop a minesweeper game using a text-based interface. This means the game board is displayed by printing it to the terminal, and the player interacts with the game by giving text commands.

Exercise 1: UML class diagram

Before you start the implementation, please first according to the game mechanism of minesweeper, design an UML class diagram on Visual Paradigm (or some other UML modeling software of your choice). Make sure your UML diagram includes inheritance in a logical way.

Some pointers:

- Discuss! This assignment does not have a single correct solution, so compare your own ideas with those of your team mate, and analyze the pros and cons of each.
- Be detailed: include attributes (name and type), method parameters and return types, correct relationships (arrows) between classes, etc.
- Inheritance rarely makes sense if the children do not exhibit different behavior, so make sure this is included in your design! This means your child classes should have at least one significant method.
- One of the design principles of object orientation is that each class should be responsible for only one thing. Some practical examples for this exercise:
    - A game is not the same as a game board
    - Try to separate the visual and non-visual parts of your application
    - Try to separate the user input from the rest of your application

  A good rule of thumb to follow is: if any of the above elements were to change, you should need minimal changes to other parts of your code (= other classes). For example, if we decide to replace the visualization to be graphical instead of text-based, ideally this would mean swapping the class responsible for the visualization but not touching anything else. Keep the concept of interfaces in mind to achieve this.

Once your lab teacher has approved your design, you can start the implementation.

Exercise 2: Implementation of game

Exercise 2a: Basic minesweeper game

Implement your game based on your UML design. Think about how you want to represent your board, and which commands you will need to play the game.

Some pointers:

- Consider the data structures you will use, especially for your game board (a 2-dimensional grid of fixed size).
- Use recursion (a function calling itself) when opening an empty tile.
- At some point in your implementation you will have to check the neighbors of the current tile. There are nine different situations: top left, top right, bottom left, bottom right, top row, bottom row, left column, right column, and center tile. Do NOT implement this as nine different if-statements with very similar code. If you're smart about this, you can implement this with a single if-statement.

Exercise 2b: A new type of tile

First make sure you have a finished, fully functional, traditional minesweeper. When this works it's time to get creative. Invent at least one new type of tile and implement it in your game.

Some examples for inspiration:

- Mine-clearance tile: when opening this you find some explosive disarming equipment. The next time you click on a mine, instead of the game ending, this equipment protects you and is consumed in the process.
- Mini-mine tile: When opening this tile, instead of the game ending, your score is decreased.
- Radar tile: When opening this tile, a random mine is flagged automatically.

If your code was well designed, you should be able to implement this with minimal changes to your existing code. Discuss your solution and implementation with your lab teacher.

Exercise 3: Write your test code

You will write your own test code to check if your implementation is correct. You should at least test the functions below:

Generate the correct world according to customized parameters, namely, row, column, and number of mines
Generate the correct world according to difficulty
Allocate mines
Update numbered tiles
Flag, unflag an unopened tile

Output the board
Open a tile with explosive neighbors
Open a tile without explosive neighbors
Open a mine
Flag an opened tile
First tile rule
Game lost
Game win
Interaction based on text commands

Hint: a) taking edge cases into account as much as possible, e.g., invalid input; b) to make the test of opening a tile easier and in a controlled way, you can implement a method in Minesweeper class that enables generating a board according to a given 2D-array.