

## MSArtifacts-Timing

In this example, we explore how the Minesweeper V0.0 artifacts (i.e., the minimal version) are affected when we add the following feature. Changes are highlighted.

Feature id: timing

Description: The game keeps track of the total time spent on a game. The counting starts from the moment the first cell is cleared or marked and stops when the game is won or lost. Time elapsed is shown to the player after every mark/clear operation.

---

### Minesweeper V0.0

#### Overview

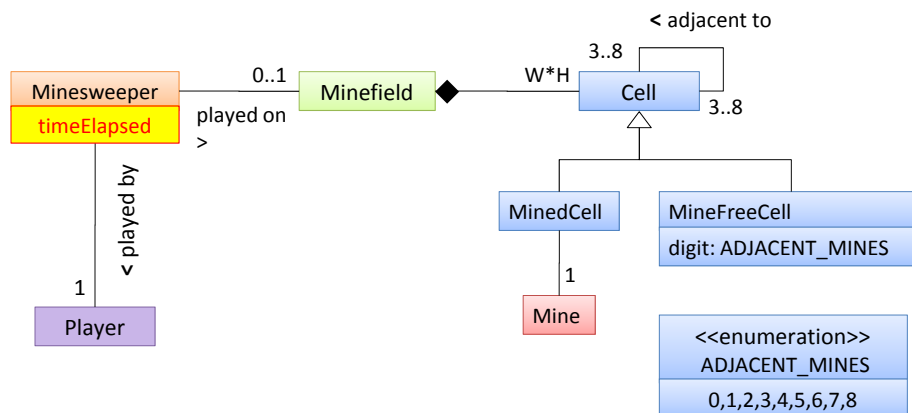
Minesweeper is a single-player computer game. The game starts by giving the player a minefield consisting of cells some of which are mined. Initially all cells are hidden. The player can win the game by deducing all cells correctly.

The player can clear a cell which he/she deduces to be mine-free or mark a cell which he/she deduces as mined.

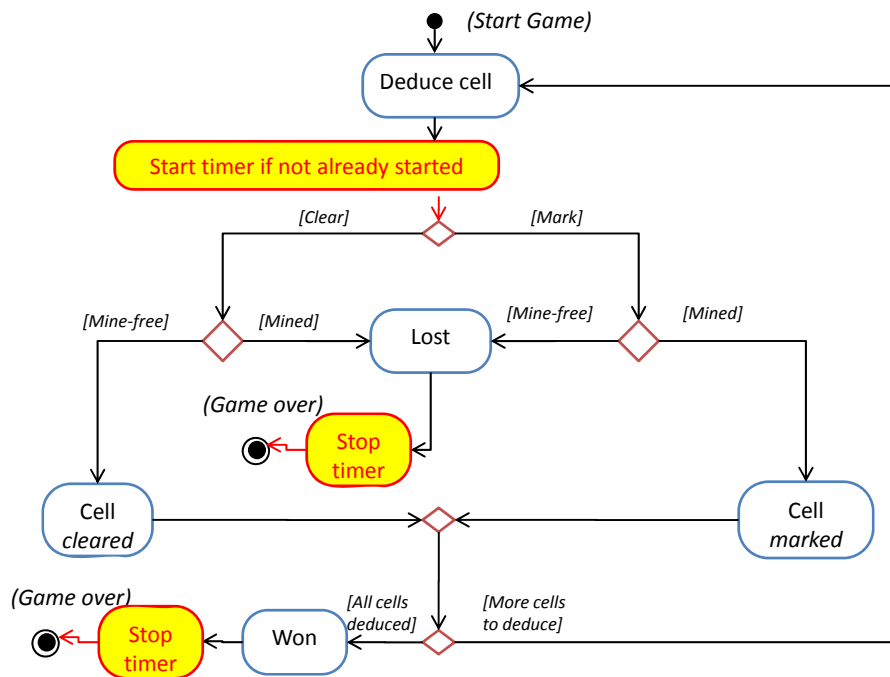
A cleared cell reveals a digit indicating how many adjacent cells are mined. The player can use this information to deduce the nature of adjacent cells. **Time elapsed is shown to the player after every mark/clear operation.**

The game is lost immediately if a mined cell is cleared or a mine-free cell is marked.

#### Domain model



## Game flow



[Note how adding 'if not already started' avoids the use of another branch. The objective is to keep the diagram simple. We need not follow the UML notation to the letter if it doesn't add value. Using a branch is ok too.]

## Supplementary requirements

- One-mine-per-cell rule: There can be no more than one mine behind a cell.
- We do not time the game.
- We do not score the game.
- We do not show mine count.
- H,W,M is predetermined.
- Mine locations are randomly generated.
- We do not allow undo.
- Time elapsed is calculated from the first clear/mark action of the game. It is updated at every subsequent clear/mark action until the game is lost or won.

## Use cases

Use case: 01 - play game

Actors: Player

MSS:

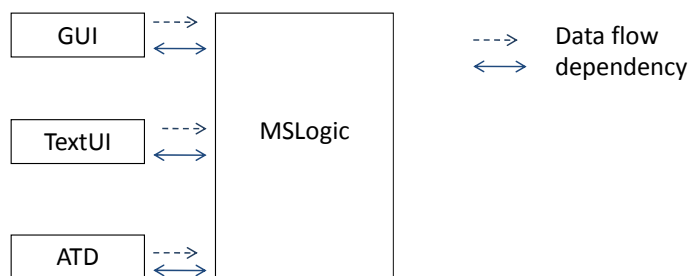
1. Player starts a new game.
  2. Minesweeper shows the mine field with all cells initially hidden.
  3. Player deduce a cell by doing one of the following actions: Clears a hidden cell, Marks a hidden cell.
  4. Minesweeper shows the updated mine field **and the time elapsed**.
- Step 3-4 are repeated until the game is either won or lost.
5. Minesweeper shows the result.

## Feature list

1. UI
  - 1.a Text UI
  - 1.b GUI
2. Minimal game play (new, mark, clear)

### 3. Timing

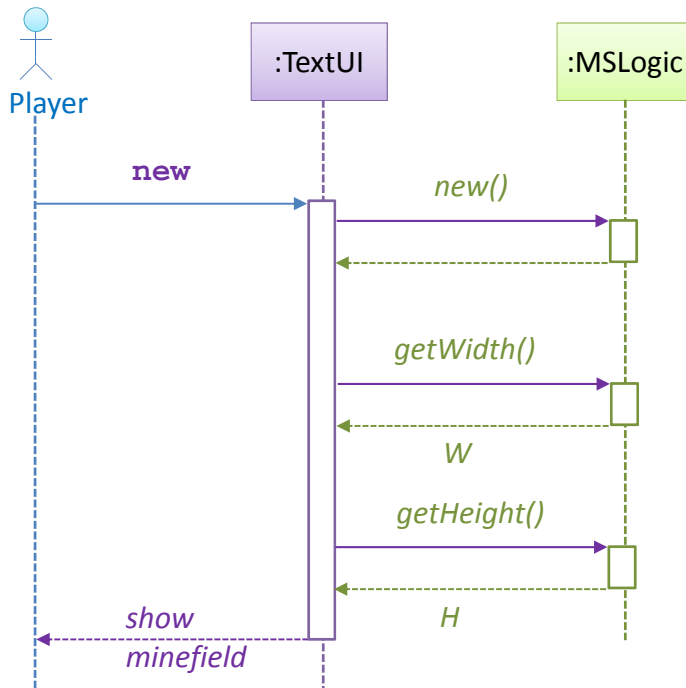
## Architecture



**No changes**

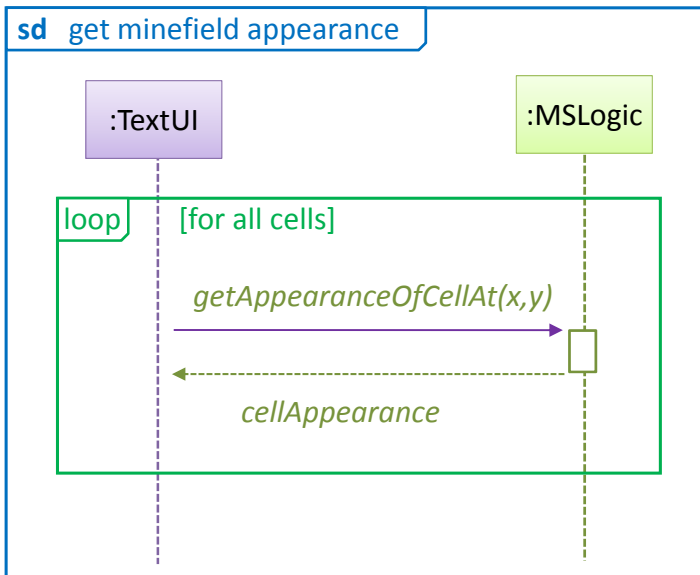
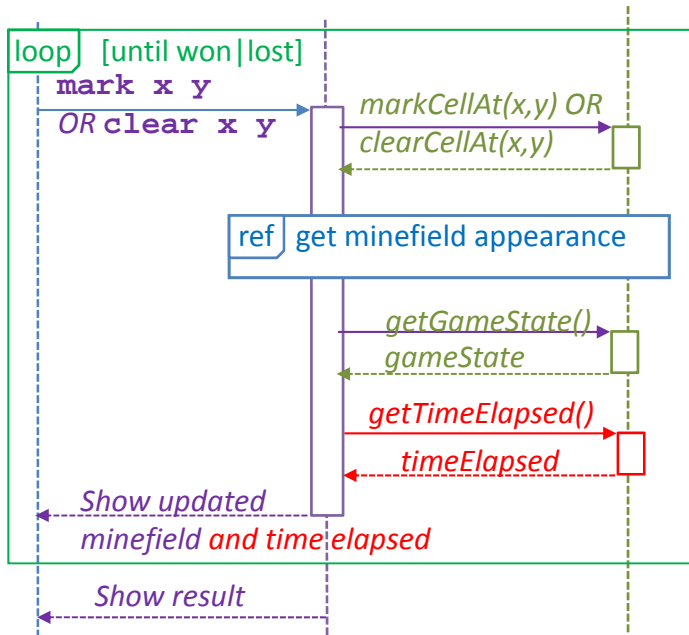
## Sequence diagrams

sd play MS (part A)



No changes

sd play MS (part B)



No changes

### MSLogic API

- newGame(): void
- getWidth():int
- getHeight():int

- clearCellAt(int x, int y)
- markCellAt(int x, int y)
- getGameState() :GAME\_STATE
- getAppearanceOfCellAt(int x, int y): CELL\_APPEARANCE
- getTimeElapsed():int  
(you may also write an operation contract for this)

## Glossary

- minefield: A rectangle grid of W x H cells hiding M mines. Initially, all cells are hidden.
- hidden : a hidden cell does not show whether a mine is hidden behind it.
- W : width of the minefield (in cells)
- H : Height of the minefield (in cells)
- M : Number of mines in the minefield
- MS : minesweeper
- clear : deduce a cell as mine-free
- mark: deduce a cell as mined
- adjacent cells: cells that share a boundary or a corner with a given cell
- mine-free cell:
- mined cell:
- incorrectly-cleared: a mined cell that has been cleared
- incorrectly-marked: a mine-free cell that has been marked
- GAME\_STATE: READY, IN\_PLAY, WON, LOST, PRE\_GAME
- CELL\_STATE: HIDDEN, CLEARED, MARKED
- CELL\_APPEARANCE: HIDDEN, ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, MARKED, INCORRECTLY\_MARKED, INCORRECTLY\_CLEARED
- Time elapsed: time since the first cell was cleared/marked.