# CBL Game Development

Chosen game: Minesweeper

Made by: Eliza Oborzyńska and Ece Özhan

## Advanced topics of choice:

- Version control: GitHub (code share, baseline)
- Program Configuration: Java *.properties* file
- GUI technology: Swing

## Product backlog:

1. Minefield grid definition
    a. Define main field grid requirements
    b. Define main data structures
    c. Define game properties (*.properties* configuration file)
    d. Define an example minefield grid

   How to demo:
   In VS code show the main data structure declarations.
   In text editor open and show *.properties* configuration file

   Notes:
   Learning objective is to represent minefield layout in array structure.

2. Minesweeper grid rendering

   How to demo:
   Start the game and show the grid. Compare VS structure with GUI display.

   Notes:
   Learning goal is to start learning how GUI works including layout management (e.g. *GridLayout*).

3. Minefield grid with content hidden

   How to demo:
   Show the grid with buttons on it that can be clicked on (yet no processing of interaction).

   Notes:
   Learning objective is to explore more about how GUI works.

4. Basic user interaction handling on the minefield

   How to demo:
   After the click on the button the content behind is shown.

   Notes:
   Learning objective includes writing a custom *MouseAdapter* to be able to click on the button.

5. Advanced user interaction processing with right mouse click

   How to demo:
   Demonstrate how right mouse click marks the location of a mine.

   Notes:
   Learning objective includes writing a custom *MouseAdapter* to handle user mouse events. Specifically, mouse clicked right.

6. Advanced user interaction processing with left mouse click

   How to demo:
   Demonstrate how left mouse click results in different actions such as opening up empty cells, hitting a mine or showing cells with numbers that indicate the number of mines it neighbors.

   Notes:
   Learning objective includes writing a custom *MouseAdapter* to handle user mouse events. Specifically, mouse clicked left.

7. Overall Game Control
   a. Starting the game
   b. Ending the game

   How to demo:
   Launching the program opens a ready-to-play MineSweeper window. The game starts when the first cell is clicked. The game ends when the user wins or loses. The user wins when there is no click on a mine, all mines are correctly marked, and no closed cells are left. The user loses when either the user misjudges a mine location or the game time outs. The user can start a new game by clicking on the emoji icon. The user can end the game anytime by clicking the ‘X’ button on the frame.

   Notes:
   Set up and configure a new minefield to play with. Learning objective is to discover different execution paths involving correct and wrong decisions e.g. opening up cells, marking a mine, hitting a mine, etc. Follow the conditions for ending the game in code.

8. Obtain and integrate icons to Swing components

   How to demo:
   Show GUI items with icons on them, these include the button for the minefield cells that are marked with a flag, mine/bomb, numbers 1, 2 or 3.

   Notes:
   Learning objectives include rendering Swing components with images on them.

9. Status bar rendering

   How to demo:
   At the top of the MineSweeper panel, there is a status bar. It displays how many mines are left in the grid and the time it took so far. In addition, there is also an emoji icon for restarting the game and showing the end game status.

   Notes:
   Learning objective includes rendering GUI components such as JPanel, JLabel and JButton, including layout management (e.g. FlowLayout).

10. Status bar changes during game

   How to demo:
   When a mine is marked on the grid the amount of mines should be lowered by one. Each second the time ticks. When the game is lost the emoji icon changes to a sad face.

   Notes:
   Learning objective includes controlling GUI components such as JLabel and JButton and updating their contents.