CBL Game Development

Chosen game: Minesweeper

Developed by: Eliza Oborzyńska and Ece Özhan

Advanced topics of choice:

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

Product backlog:

1. Minefield grid definition
    a. Define minefield grid requirements
    b. Define minefield 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 concept in array structure.

2. Minesweeper grid rendering
    a. Develop top control panel
    b. Develop minefield panel

    How to demo:
    Start the game and show the game board with GUI components and without any action listeners working. Compare data structure with GUI display.

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

3. Obtain and integrate icons to Swing components

    How to demo:
    Search for emoji and mine icons from the internet. Develop tile icons (empty, numbered and cross-flag) in PowerPoint and Paint 3D. Show GUI items with icons on them, which include control buttons and minefield cells that are marked with a flag, cross-flag, mine/bomb, empty tile, and numbered tiles from 1 to 8.

    Notes:
    Learning objectives include rendering Swing components with images/icons on them. Also learn how to organize and develop icons with specific sizes (32x32)

4. Minefield grid with contents hidden
    a. Set icons to control buttons
    b. Draw ImageIcons on grid for the cells
    c. Make trials with action listeners & mouse adapters

How to demo:
Show the grid with buttons and image icons that can be clicked on (yet no full processing of user action).

Notes:
Learning objective is to explore more about how GUI components are rendered and action listeners are defined.

5. 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.

6. Advanced user interaction processing with right mouse clicks

   How to demo:
   Demonstrate how right mouse click marks the location of a mine with a flag. Subsequent clicks toggle on and off the flag. Mine count display decreases with the addition of a flag, and increases with the removal of it.

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

7. Advanced user interaction processing with left mouse click

   How to demo:
   Demonstrate how left mouse click results in different actions such as discovering and 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.

8. Overall Game Control
   a. Starting the game (from launch, restart button or level buttons)
   b. Ending the game (by mine hitting, timeout or closing the frame)
   c. Dynamically changing difficulty levels

   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. The user can also dynamically change the difficulty level of the game among 3 options: beginner, intermediate and expert. Each level has different grid size and timeout value.

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. Identify and implement the possible use cases of the game ((re)start, change level, win, lose (hit mine or timeout), close).

9. User feedback management and display
    a. Top control panel information elements
    b. Bottom status bar information elements

How to demo:
At the top of the minefield, there is a control panel. In addition to the control buttons, the panel displays information on mine count and the time lapse every second. The emoji icon in the middle of the panel visually reflects the game status, being ongoing (smiley), won (cool), or lost (sad).

At the bottom of the minefield panel, there is a status bar. By default, it displays how many mines are left in the grid. It also textually displays the game status, in case the game is won or lost. Losing the game can be due to mine hitting or timeout.

Notes:
Learning objective includes rendering GUI components such as JPanel, JLabel and JButton, including layout management and controlling (preferred) sizes. Explore how refresh of the GUI is achieved after changing text or icon on the components.

10. Working with Git and GitHub
    a. Install Git on laptop and create GitHub account.
    b. Create remote GitHub repository and upload code in local repository
    c. Enable Git version control (install extensions) on VSCode and connect to remote repository
    d. Continue development using git:
        a. Branch
        b. Commit
        c. Pull (request, merge)
        d. Clone

How to demo:
Show the GitHub accounts and repositories on GitHub web interface. In VSCode open and demonstrate the use of Git source control through Source Control, Extensions, Remote Explorer and GitHub views. Show the history of commits and track changes in the repository.

Notes:
Learning objective is to learn and understand the basics of source code version control using Git and GitHub tools. Experience with developing code in local and shared, remote repositories.