EECS 448

<u>Group Meetings</u>

- August 31st: 448
  - Eaton 2 (Classroom)
  - Bennett, Brad,Tsung Yu, Matt
  - Discussed everyone's schedules and good times to meet the week. Waiting on a work schedule to be sure for next week. Also started to discuss what programming language we would like to use and if we are going to build a GUI or have it run in terminal.
- September 4th: 1005D Eaton
  - Bennett, Brad,Tsung Yu, Matt
  - Decided to use python for Project, began some outlining on Project. Made gitHub Repo.
- September 5th: Eaton 2
  - Bennett, Brad,Tsung Yu, Matt
  - SCRUM meeting, tested some code, set up meeting on Friday
- September 7th: Eaton 2
  - Bennett, Brad,Tsung Yu, Matt
  - SCRUM meeting, discussed what we are going to talk about at meeting tonight
- September 7th: Spahr 1326
  - Bennett, Brad,Tsung Yu, Matt
  - Work on detailed outline of project, assigned work and discussed what to do over weekend.
    - Brad- working on board creating and initialization
    - Bennett- working on executive class / game logic. Also making tile class
    - John- working on revealing tiles
    - Matt- working on GUI and user input
- September 10th: Eaton 2
  - Bennett, Brad, Tsung Yu, Matt
  - Discussed progress made over the weekend
- September 10th: Lab
  - Bennet, Brad,Tsung Yu, Matt
  - Merging work, connecting GUI
- September 13th : Spahr 2322
  - Bennet, Brad,Tsung Yu, Matt
  - Merging work, implement the GUI, testing the code
- September 15th: Google Hangout
  - Bennet, Brad,Tsung Yu, Matt
  - Review the code and debug

- September 16th: Google Hangout
  - Bennet, Brad,Tsung Yu, Matt
  - Final Review and debug

GUI / Main class

```
  While user want to continue
     Game start GUI (method)
        Ask width, height, numBomb
        instructions?
     Main game GUI (method)
        Main Game loop
     End game GUI (method)
        Game over or You win
        Ask if want to play again
  End while
Define other methods as need be
```

Description on how work was split between teammates:

We have the team meeting at Spahr Library 1326 and discuss how to split the work, we use the board to draw the block diagram and talk about the game logic. After the diagram finished, we share our personal coding experience with each other and roughly split the work to 4 parts. We split to board setup, reveal recursion function, game logic for executive and GUI.

Some of the challenges we faced during this project were learning git and github as it is integrate into the IDE we were using, Pycharm. Another obstacle was learning Python itself, none of us had known Python before this project. The third largest obstacle was integrating what had been done in terminal with game logic with the GUI.

Features that did not make the final version include: being able to restart the game from an interface, sounds, clock, and a better recursive algorithm. This was mostly due to a time constraint from learning a new language and GUI programming.

A few things we would have done differently are pick a language we already knew, modularize the GUI code better from the start, and expand platform to different OSes.

EECS 448

Minesweeper game Class Hierarchy

- Tile : A single tile object
    - Methods
        - GetAdjacentBomb : return int
        - getIsBomb : return bool
        - getIsVisible : return bool
        - getIsFlagged : return bool
        - countAdjacentBomb : return void
            - Counts all the adjacent bombs next to it
    - Variables
        - AdjacentBomb : int
        - isBomb : bool
        - isVisible : bool
        - isFlagged : bool
- Board: The main reference board
    - Constructor(length, width, num bombs) : Create an x by y board
    - Methods
        - RevealTile(x,y) : bool
            - If RevelTile.getAdjacentBomb==0
                - reveal (x+1,y), (x-1,y), (x,y+1), (x,y-1)
        - makeBoard
    - Variables
        - gameBoard[length][width] : Tile
        - Length
        - Width
        - Bomb count
        - totalFlags
- Executive : Handles the main game loop and interfaces the whole program together
    - Game Loop
        - Wait for user input
        - Reveals space user ‘presses’
            - If bomb game over
            - Else reveal Tile
        - Check for win/lose condition
    - On startup
        - Prompt user for board size and mine count
        - Initializes the board
- User Interface
    - Grid of ‘tiles’
    - On user click on tile reveal tile
    - Handle flag logic