Methods Used:

Cell:

Creates A cell object that contains the current state of the cell

Values:

cover: Boolean if it is covered

mark: Boolean if it is marked

mine: Boolean if it is a mine

value: Integer Mines surrounding

**getValue()**

Gives the value stored in Cell Object

**uncover()**

Sets the cell value cover to false

**setMark(Boolean mark)**

Sets the Mark status of the Cell Object to mark

**isCovered()**

Checks if the it is still covered and returns as true or false, default is true

**setAroundMines(int count)**

Sets the count of mines surrounding the Cell

**getAroundMines()**

Returns the count of mines surrounding the Cell

**setMine(Boolean b)**

Sets the mine Value to b

**isMine()**

gives the Mine value

**returnV()**

Returns the value of ‘B’ if a bomb “ “ if not and the count otherwise

Board:

Creates and maintains the Cell Array and gives values back to the GUI

Values:

state: String[][] contains values for the cells

cellArray: cell[][] contains Cell objects that contain all values info

**setup()**

Creates a 2d Array of Cell Objects then randomly selects 10 as mines

Uses: Cell.setAroundMines(I,j), Cell.getAroundMines(I,j), Cell.isMine(I,j)

**reveal(int i,j)**

Reveals the Cell at coordinates given and changes the state appropriately, will also recursively call itself if blank

Uses: Cell.returnV(I,j), Cell.uncover(I,j), Cell.is Covered(I,j)

**mark(int I,j)**

Sets the Cell as Marked and changes state to reflect change

Uses: Cell.getMark(I,j), Cell.setMark(I,j)

**unmask()**

reveals all the Cells

Uses: Board.reveal(I,j)

**checkWin()**

Checks if the all are covered or mines and changes win value to true if so

Uses: Cell.isCovered(), Cell.isMine()

GUI:

matrixTen.fxml:

AnchorPane has three labels: a title, a mine count, and directions.

**mainLayout**

Main.java:

links to the java fxml document

Gridpane is created and added as a child to the anchorpane