Week 4: Bomb Sweeper

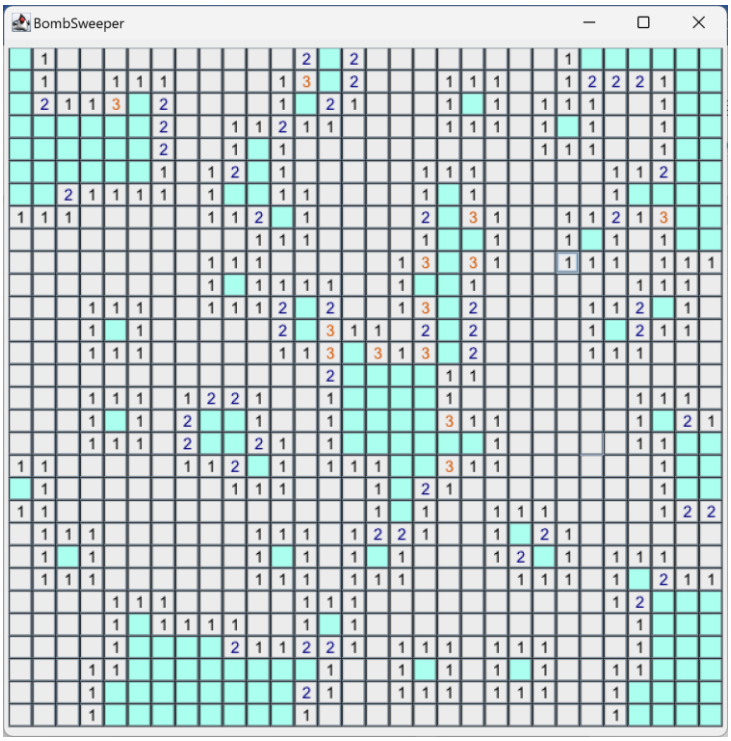
Aim of the Exercise

This exercise focusses upon testing and strengthening your understanding of inheritance, object references and implementing recursive algorithms. **The aim is for you to gain experience of writing more algorithmically complex code in Java while maintaining code elegance through using a combination of iteration and recursion.**

The Task

Your task this week is to implement the logic behind the well-known game of minesweeper. The graphical functionality has been provided for you, as has a template for a (reasonably!) elegant object model. For this task, you should only need to modify the **BombSquare** class, which provides a basic template for the exercise.

You are not permitted to modify the GameBoard or GameSquare classes. Restrict your work to the BombSquare class, and/or any other new classes you may wish to create.



Task 1: Detecting Bombs

Review the Javadoc documentation for the provided classes and study the **BombSquare** source code. Note that the **BombSquare** class is a subclass of **GameSquare**, which also holds application specific information. Note also that in addition to its inherited instance variables, the **BombSquare** class holds information on whether or not a given square contains a bomb. This is generated by a random process in the constructor, such that 1:10 squares contain a bomb.

Based on the resources provided, develop the **BombSquare** class such that when a square is clicked it, it obeys the following rules. **Try to implement this code in as elegant a fashion as you can.**

- Displays an image of a bomb if that square has a bomb on it.
- Displays the number of bombs in surrounding squares otherwise. Note that surrounding squares a classified as the eight squares that are horizontally, vertically or diagonally adjacent to the square.
- Your program must work for a board of any size or shape (you can modify this in the main method in the Driver class).

Hint: You'll need to study the methods and instance variables you have inherited from the **GameSquare** class…

Task 2: Expanding into Empty Space

Traditionally, minesweeper implementations enhance the playing experience by reducing any unnecessary work of the player.

Using recursion, update your program such that when a square is clicked that has zero bombs in its surrounding squares, it not only reveals that square, but also any adjacent squares (horizontally, vertically or diagonally adjacent).

Portfolio Contribution

As discussed in the introductory lecture, all practical work this term will contribute to your portfolio assessment.

This piece of work will carry marks for:

- core functionality related to the accurate implementation of the specification and constrains described above.
- maintaining object-oriented paradigm. Solutions following a procedural/functional paradigm will be significantly penalised.
- scalable and extensible solutions that do not make unnecessary assumptions.