

Team 5 Idea Submission

Minesweeper Game

# 

# Table of Contents:

[**Table of Contents:**](#_vovpmuepmly3) **2**

[**Team members**](#_uix1d0avz24k) **3**

[**Idea overview:**](#_xrvn7gflgqn4) **3**

[**Requirements and Constraints**](#_not4micmijl8) **3**

[Exception Handling:](#_77qxj9oljmtk) 3

[GUI:](#_s9lkv4ldq4wr) 4

[Proper class hierarchy:](#_feo1v7j4p0wc) 4

[Threading](#_wcqvgwt62gxg) 4

[**Design Mockup:**](#_c74mxxltzvd) **5**

[**Class Diagram:**](#_ywyx17igaj8m) **6**

[**Classes:**](#_8d3mub8s3h0i) **7**

[Panel Class:](#_dc18lj9e8rag) 7

[Number Panel](#_q622bdalgd4w) 7

[BlankPanel](#_a8lmfn3rte4g) 8

[MinePanel](#_obbk6xrc33u3) 8

[Playfield:](#_wylobx60xccw) 9

[GameController:](#_sr658o6xsuhy) 10

[**Copyright:**](#_3mqwhsj2pjaa) **10**

# Team members

|  |  |
| --- | --- |
| Name | Email |
| Zachary Kirchens | zacharyalankirchens@gmail.com |
| Bao Wong | [baohwong@yahoo.com](mailto:baohwong@yahoo.com)(gmail account) |
| Jacqueline Fernandez | jacquelinefernandez799@gmail.com |

# Idea overview:

We will be replicating the classic Windows application “Minesweeper”. We will be doing this using the following gameplay loop:

* The game itself has a 10\*10 matrix of “tile” objects. 10 mines will be placed around the board using a random number generator. Then, the game will loop through the other tiles and change them to either numbered or blank tiles, with the number tiles measuring the number of mines in their surrounding 8 tiles (like how the game detects them)
* The user will click/tap on squares throughout the board avoiding the mines. If the user clicks on all the squares that were not mines, then the user wins. Otherwise they lose.
* This program should fulfill the requirements set forth by the project requirements listed in the “Requirements and Constraints” header.

# Requirements and Constraints

For this project there are several requirements. This is how our project meets or exceeds these requirements as listed.

## Exception Handling:

Our project handles exceptions by ensuring that only certain buttons/ clicks are registered. Any characters outside of (left clicking), may not be registered.

For inter-class communication, basic type exception handling will be implemented.

To go further, this game doesn’t require text input of any kind in it’s initial state. Should we choose to expand the project to include customizable window sizes

## GUI:

We will be using Java Swing for the windows as well as interactive elements. For the interactive elements, we will mostly be using buttons, which will change states and content based on gameplay.

We have a window that will serve as the container for 100 buttons for the 10x10 grid minefield.

## Proper class hierarchy:

We will be using several classes that make full advantage of inheritance. These include a main inheritance abstract class. There are three child classes named “BlankPanel”, “MinePanel”, and “NumberPanel” which extends to the main “Panel”. “Windows.java” will server as the main class in which the program will run from. “Playfield.java” manages the creation of the panels and events.

## Threading

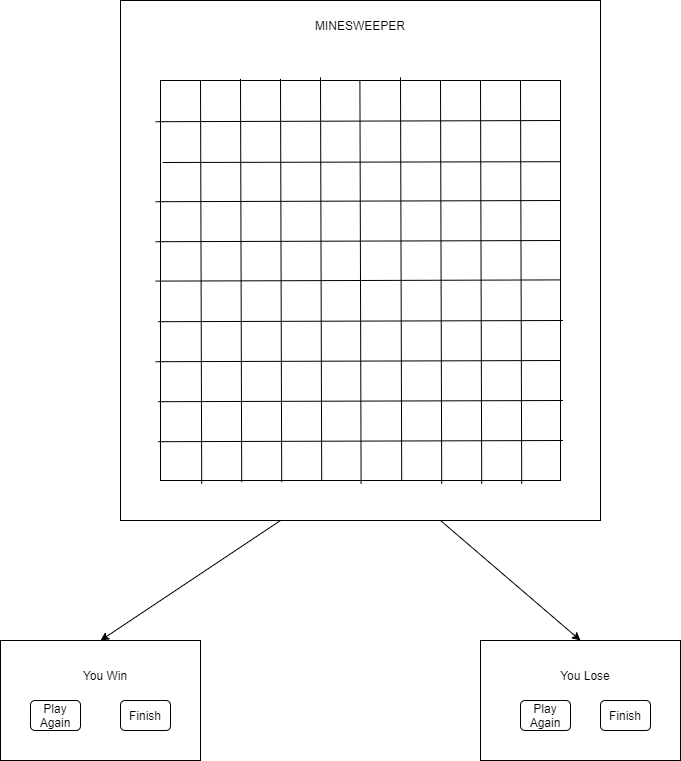
Our project makes use of multithreading using Java Swing. Whenever a window opens through the win/lose state, a new thread is opened. The main thread is reserved for the game itself.

# 

# Design Mockup:

The design mockup goes here (a visual representation of the main screen)

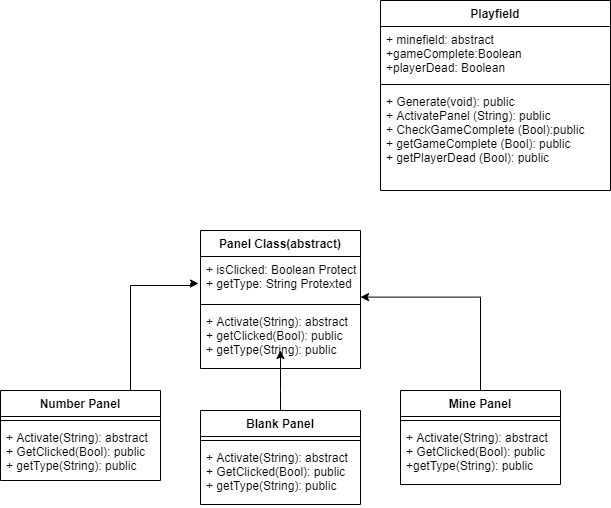
“Design Mockup.drawio”



# Class Diagram:

“Flowchart Diagram.drawio”

# 



# Classes:

Description of classes in detail. Class Diagram listed above. Uses Panel as abstract class with each panel type extending to it.

## Panel Class:

Abstract class.

Contains the following variables:

* Protected Boolean isClicked
  + If the panel has already been activated.
  + Used for blank panels
  + Is set to false by default;
* Protected String type;
  + Based on what kind of panel it is
  + Used primarily for world generation
  + Will be one of the following:
    - “ ”=Blank panel
    - 1-8= number panel
    - \*=mine panel.

Contains the following methods:

* String Activate
  + This method is created per panel, basically tells what happens if the panel is clicked.
  + Returns a String (this is used by encapsulating classes)
* Bool getClicked
  + Returns the value of isClicked
* String getType
  + Returns the value of type.

## Number Panel

Extends the Panel Class

Is a panel if there are mines surrounding it.

Contains the following variables:

(No variables, uses parent classes variables)

Contains the following methods:

* Constructor(int value)
  + Creates a new NumberPanel with a value of the passed parameter.
  + The passed parameter is set to Type
* String Activate:
  + Inherited from Panel Class
  + This method changes the contents of the button that was clicked to the number listed in the variables
  + Changes isClicked to true;
  + Returns the value of Type using getType
* Bool getClicked
  + Inherited from Panel Class
  + Returns the value of isClicked
* String getType
  + Returns the value of type.

## BlankPanel

Extends the Panel Class

The blank panel is used when there is no mines in the vicinity.

Contains the following variables:

Contains the following methods:

* Constructor:
  + Creates a new BlankPanel with isClicked set to False
* String Activate:
  + Inherited from Panel Class
  + This one changes the value of isClicked to true, then calls activate on all the other panels around it if isClicked is false (calls isClicked on them) and the type is not a mine.
  + Returns the value of type using getType
* Bool getClicked
  + Inherited from Panel Class
  + Returns the value of isClicked
* String getType
  + Returns the value of type.

## MinePanel

Extends the Panel Class

Basically is the panel if it is a mine

Contains the following variables:

Contains the following methods:

* Constructor:
  + Creates a new MinePanel with isClicked set to false.
* String Activate:
  + Inherited from Panel Class
  + Tells the Game Controller to end the game.
  + Returns an “\*”
* Bool getClicked
  + Inherited from Panel Class
  + Returns the value of isClicked
* String getType
  + Returns the value of type.

## Playfield:

Contains a 10\*10 matrix of panels for the game.

Contains the algorithm for board generation.

Contains the following variables:

* Panel [10][10] minefield
  + Contains all the linked numbers for the panels in the game.
* Boolean gameComplete
  + Tells the GameController if the game is complete
  + False by default
* Boolean playerDead
  + Tells the GameController if the player is dead
  + False by default

Contains the following methods:

* Void Generate
  + Sets up the playfield for the game and resets the player status.
  + Does this by:
    - Loops through and creates 10\*10 new instances of a BlankPanel object
    - Placing 10 mines randomly throughout the map.
      * Does this by creating a new MinePanel to replace the BlankPanel
      * It is statistically unlikely, but if a mine is unable to be placed after more than 3 times, it will be skipped.
    - Loops through the panels, if there is a mine in any of the 8 adjacent panels the BlankPanel object is replaced with a NumberPanel with the number of mines.
    - Sets gameComplete and playerDead to false
* String ActivatePanel(int x, int y)
  + Activates the panel at position x,y
    - If panel doesn’t exist, ignores requires, and returns null
    - Checks to see if there are any more Number panels that aren’t activated using the the CheckGameComplete method. If it’s true, it changes gameComplete to true
* Bool CheckGameComplete
  + Loops through the minefield to see if there are any remaining number panels that aren’t activated.
    - Loops through the minefield, and if it comes across a NumberPanel or Blank Panel that isn’t activated returns false
    - If the loop is able to complete, it returns true
* Bool getGameComplete
  + Returns the value of gameComplete
* Bool getPlayerDead
  + Returns the value of playerDead.

# Copyright:

Images:

* Cover image for documentation is considered CC0, but was retrieved from freesvg.org
  + <https://freesvg.org/vector-image-of-cartoon-sea-mine>