Description of Game

**Name**

Mastermind

**Goal**

To guess the randomly-generated “code” within a certain number of tries.

**Number of Players**

One

**Controls**

The mouse is used to interact with the game.

**Description**

You are a code-breaker. The computer has generated a code, composed of colours in an unique arrangement, that you must guess with only a few tries. The computer is feeling a bit merciful today and will tell you a few things:

* If you’re completely wrong (there will be a red circle in the sidebar).
* If you have the right colour, but it’s in the wrong place (there will be a blue circle in the sidebar).
* If you have the right colour in the right place (there will be a green circle in the sidebar).

However, it isn’t nice enough to tell you which colour it’s talking about – that’s up to you to figure out.

Do you think you can crack the code? Are you the next mastermind?

The computer’s random “set” is revealed at the end of the game.

“Check” button to check row

Locked rows

Current row

**Hints:** Green indicates correct colour in correct location, blue indicates correct colour in wrong location, red indicates wrong colour.

Previously filled row, cannot be changed.

Colours to choose from.

Check

Classes

# Board

This class contains the BoardPanel and the buttons that the users will interact with.

**Instance Variables**

BoardPanel panel – the BoardPanel.

JPanel buttonPanel – the panel that contains the buttons.

JButton check – the button that says “check” that allows the user to check their current row.

JButton instructions – the button that says “instructions” that allows the users to open the instructions in a pop-up window.

# BoardPanel

The all-encompassing class. This will essentially be the window the user is interacting with.

**Instance Variables**

Indicator[] – An array of indicators – one for each row.

Palette – The palette will be at the bottom, for the user to choose colours from.

Row[] – An array of indicators.

activeColour – the current colour chosen by the user from the Palette.

randomSet[] – The computer’s randomly generated set of Circles (one Row).

# Circle

Each individual circle. Each has its own colour and status.

**Methods**

setColour(colour) – A method to set the colour of the circle.

Draw(g) – draws the circle.

# Palette

The palette contains all the colours that the user can choose from.

**Instance Variables**

Circle[] – An array of 8 Circles so that the player can choose them to put in the rows.

Rectangle – The rectangle that surrounds the Palette.

Color[] – Array of colours

**Methods**

getSelected() – Returns the selected colour.

Draw(g) – draws the palette.

getColor(x, y) – Gets the colour at a certain x and y coordinate.

# Row

Each row is made up of 4 spaces/circles, which can be unfilled, filled, or locked (if it was a previous row).

**Instance Variables**

Circle[] – An array of 4 circles.

Rectangle – The rectangle that surrounds the Row.

**Methods**

setColor(column, Color) – sets a circle to a certain colour.

Draw(g, active) – draws the row. Draws circles if active, rectangle only if not.

getFilled() – Returns true/false depending on if the row is filled.

containsColor(Color) – returns true/false depending on if the colour is in the row.

containsDuplicates() – returns true/false if there are duplicates in the row.

getColor(index) – Returns the colour at a certain index.

# Indicator – inherits from Row

This tells the user how close they are to cracking the code.

**Methods**

setStatus(status, wrongPlace, wrong) – Sets the indicator’s 4 circles to their modes.

Implementation

1. **Begin implementing the game with the Board class.**
2. **Implement the Circle class.**
3. **Next, add the Palette.**
4. **Then, add the Row class.**
5. **The Indicator class is next.**
6. **Get the Palette to work with a Row.**
7. **Test out Row and Circle statuses, ensure they can be unfilled/filled/locked.**
8. **Implement the logic for the Indicators.**
9. **Get Rows to work with their Indicators.**
10. **Begin implementing larger number of Rows for further testing.**
11. **Perform further testing on Indicator and general Board logic (if the end is reached, does the person lose?)**
12. When all is working, attempt to increase difficulty levels by increasing the number of colours or circles needed to be guessed.