A screenshot of a computer

Description automatically generated

GUI instructions

Box 1 is a border layout

Box 2 is a JComponent

Invisible box 3 containing bottom elements is a JPanel

Discovery

Nouns:

|  |  |
| --- | --- |
| * Color | * Route |
| * Depth * Location | * Random number * Escape |
|  |  |

## CaveCell class:

* Get and set color based on depth number. Depth 1 should be lightest, depth 10 should be darkest and numbers in between are on a spectrum
* Get and set depth. This is an int that can be obtained by a cave class
* Randomize depth

## Cave class:

* Create cave cells. Create 2d array of cave cell objects
* Get route. Call route algorithm. If there is a route return yes.
* Show route path. If there is a route set cave cell color.
* Reset route path.

Implementation

Step 1a: create Basic GUI Layout

Create frame class that extends JFrame

Create border layout inside frame class

Create a text frame inside north border

Create box to hold cave

Create dummy cave cell class and test

Test: Simple run

Step1b: Create cave cells class fill GUI interface

Constructor creates cell object with 50 by 50-pixel border

Get and set color

Get and set depth

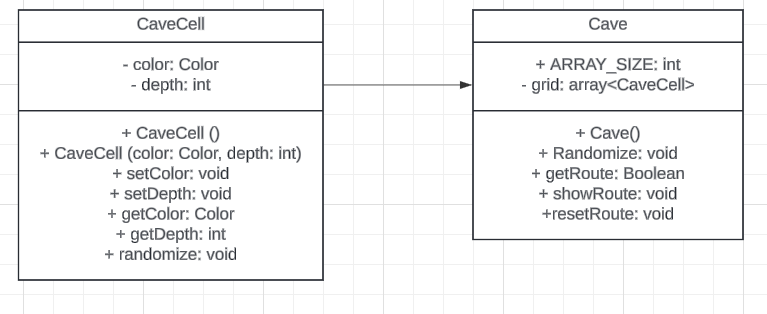
Paint method to display color and depth

Random generator

Connect color and depth

Test: Create cell object in GUI program and test methods

# **UML**



Step2: Create route algorithm

Test: with number array

Create route methods for Cave class

# **Routing Algorithm (BackTracking)**

Problem: Find whether a line can be drawn from a starting point to an end point based on these criteria:

|  |  |
| --- | --- |
| * Each square has a number * No more than 20 squares can be visited * The line can only go to the right or down | * The user chooses a starting number * The line cannot intersect a square with a higher number than the user number |

Partial solution: the final square (furthest right) is met

Examine:

If square is <= user number

If squares visited >= 0

If accepted

If we solved intro square(top left)

Return success

Else if there is a left square

Examine left square

Else if there is a top square

Examine top square

Else return fail

Base cases: oxygen == 0

arrayPOS [0][0] is solved

there is no left square

there is no top square