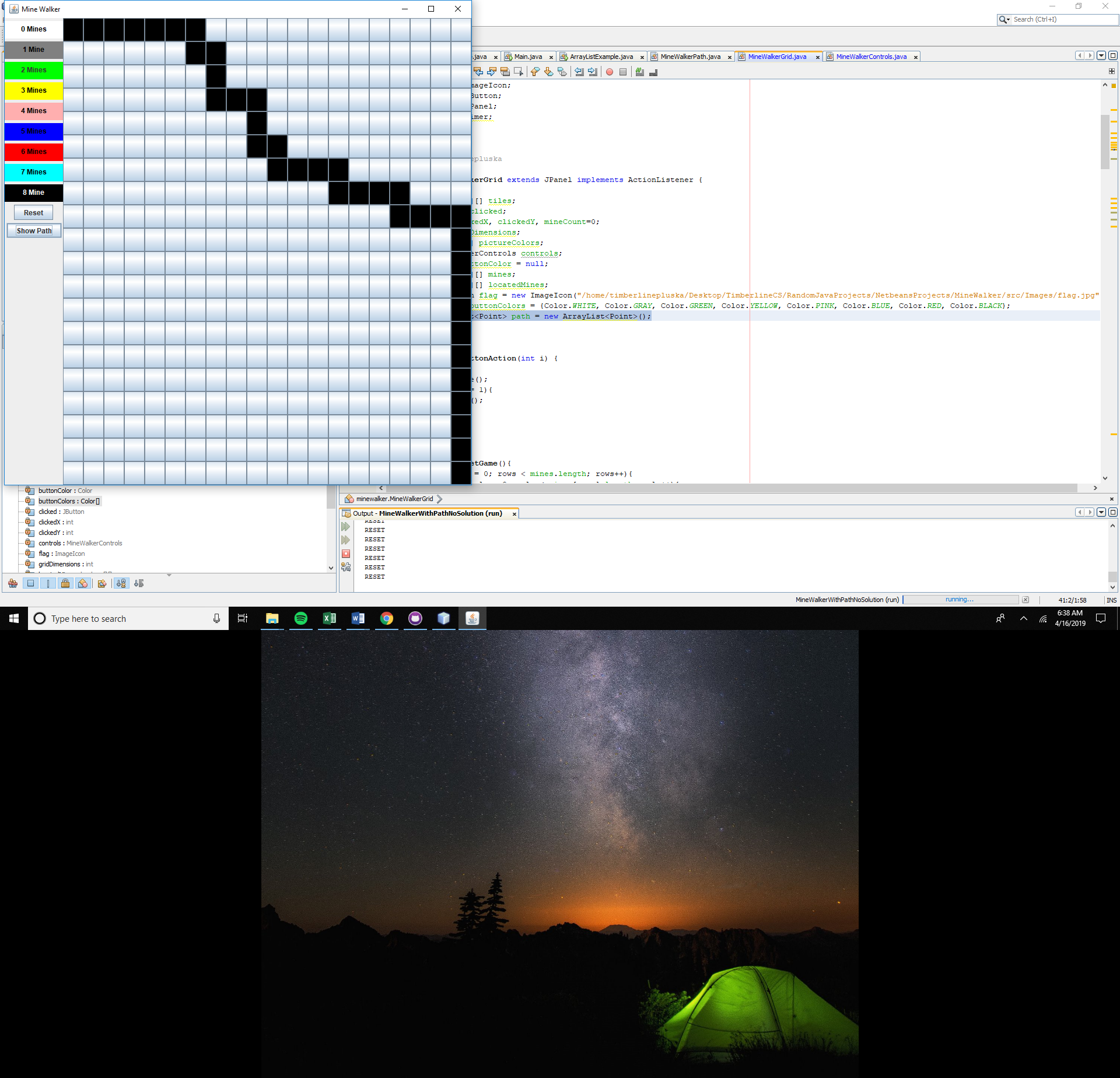
|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Mine Hunter Path** |  |

|  |
| --- |
| **Your Tasks (Mark these off as you go)** |
| * Initialize an ArrayList of Points * Write the showPath method * Have Ms. Pluska check off your showPath method * Write the MineHunterPath constructor * Have Ms. Pluska check off your mineHunterPath constructor * Complete challenges 1 thru 4 * Have Ms. Pluska check off your challenges 1 thru 4 before you continue * Receive credit for the group portion of this lab * Receive credit for the individual portion of this lab |

* **Initialize an Array List of Points**

In this lab you will build upon the Mine Hunter program you wrote previously. The goal of this lab is to write a class that when implemented will find a path through the mines on the grid like the picture shown below.



To create the path, you can assume that the path begins at coordinates x = 0 and y = 0 and ends at x = gridDimensions -1 and y = gridDimensions – 1. The path your program creates will be random that is, you will generate a random number and depending on the number the path will proceed forward or down. But, the path can NEVER hit a mine. Because we do not know how long the path will be we will store the coordinates of the path in an ArrayList. The ArrayList you create will hold Point values. Point values are convenient because we can use the points to represent x, y coordinates on the grid.

Locate the MineHunterGrid class and at the top declare an ArrayList

private ArrayList<Point> path = new ArrayList<Point>();

* **Write the showPath method**

Locate the MineHunterGrid class. Here you will write the showPath method. The show path method is a void type method and does not accept any parameters. The job of showPath is to show the random path through the mines when the “show mines” button is clicked.

The showPath method will create a new instance of MineHunterPath. To create the path, the MineHunterPath needs two pieces of information (1) the gridDimensions and (2) the mines. You will pass this information to the MineHunterPath constructor as parameters.

In the showPath method add the following,

MineWalkerPath newPath = new MineWalkerPath(gridDimensions, mines);

Once MineWalkerPath has done its job of creating the path, we need to get the path back and assign it to the path ArrayList we created previously,

this.path = newPath.getPath();

Recall that the path stores the point values of the path. To access these we need to iterate over the path we retrieved and access the x and y values for each point. You can color the path any color you choose, but in my example, I chose black.

for(Point p : path){

tiles[p.x][p.y].setBackground(Color.BLACK);

}

* **Have Ms. Pluska check off your showPath method**



Before you continue have Ms. Pluska check off your showPath method

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature \_\_\_\_\_\_\_\_\_\_\_\_

* **Declare the necessary global variables in the MineHunterPath class**

To create our path we need some information. Before we get started on the MineHunterPath class we need to declare the following global variables. Feel free to add or modify these as you go along.

private int gridDimensions;

private Point start;

private Point end;

private Point current;

private ArrayList<Point> path = new ArrayList<Point>();

private boolean done;

private boolean[][] mines;

private int x, y;

* **Write the MineHunterPath constructor**

Locate the MineHunterPath class. The constructor in this class must accept two parameters, (1) the gridDimensions and (2) the mines. In the constructor we also need to initialize the global variables we declared above.

An important feature in the MineHunterPath constructor is that we create two Point objects – one for the starting point and one for the ending point. These points represent tiles on the grid. The example below illustrates how to access the x and y values of any point on the grid,

|  |
| --- |
| **Access x and y values of a Point object** |
| Point start = new Point(0,0);  int starting x = start.x;  int starting y = start.y; |

Another important feature is that we add the starting point to our ArrayList that we declared above.

public MineHunterPath(int gridDimensions, boolean[][] mines){

this.mines = mines;

this.gridDimensions = gridDimensions;

this.start = new Point(0,0);

this.end = new Point(gridDimensions-1, gridDimensions-1);

this.current = start;

this.done = false;

path.add(start);

}

* **Have Ms. Pluska check off your mineHunterPath constructor**



Before you continue have Ms. Pluska check off your mineHunterPath constructor

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature \_\_\_\_\_\_\_\_\_\_\_\_

* **Complete Challenges 1 thru 3**

Challenge 1

Write a method called showMines()

showMines() is a void type method. The purpose of this method is to display the mines on the grid. To accomplish this task, you will iterate over the mines array you created. At each “true” location, you will set the background color of the button at that location to red (.setBackground(Color.RED). You will also set the icon on the button to a flag (.setIcon(flag);

Challenge 2

Write a method called checkForMine.

checkForMine is a void type method, which accepts two parameters (int mineX, int mineY) which represent where the user has clicked. The purose of this method is to check whether a mine is set at the mineX, mineY location. If a mine is found at the location, call showMines() to indicate that the player has lost.

Challenge 3

Write the paintTiles method.

paintTiles is a void type method, which accepts two parameters(clickedX, clickedY) which represent where the user has clicked. The purose of this method is to count all mines touching this location. And depending on the number of mines will change the background color of the button clicked to a different color in the array,

private Color[] buttonColors = {

Color.WHITE, Color.GRAY, Color.GREEN, Color.YELLOW, Color.PINK, Color.BLUE, Color.RED, Color.BLACK

};

For example, if the user clicked on the button shown below, the background color should change to yellow, because it is touching three mines.

|  |  |  |
| --- | --- | --- |
| mine |  | mine |
|  | User  clicked |  |
|  |  | mine |

Your method should account for boundary conditions. For example, if the user clicked on the button below, your program should not go out of bounds. The color of the clicked button should change to green.

|  |  |
| --- | --- |
|  | mine |
| User  clicked |  |
|  | mine |

Challenge 4

Write the isDone() method

isDone() is a boolean type method. Each time isDone() is called, it checks to see if the user has identified all the mines. If all the mines have been the identified, the method will return true, otherwise it will return false.

**Have Ms. Pluska check off challenges 1 thru 4**



Before you continue have Ms. Pluska check off challenges 1 thru 4.

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature \_\_\_\_\_\_\_\_\_\_\_\_

* **Receive Credit for the group portion of this lab**

Make sure to indicate the names of all group members, then submit this lab to the needs to be graded folder to receive credit for the group portion of this lab.

* **Receive Credit for the individual portion of this lab**

Implement challenges 1 thru 4 on your computer. You should have fully functioning game, once these methods are implemeted. Show Ms. Pluska your completed game to receive credit for this lab.