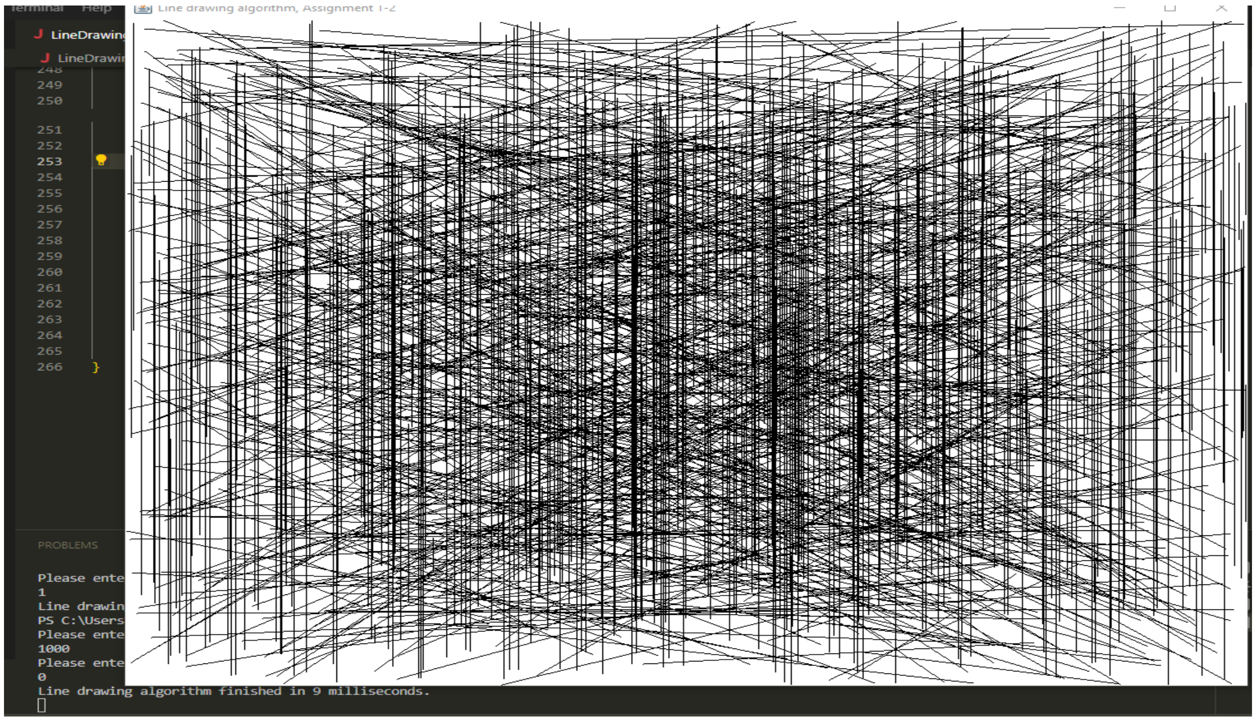
Brent Voyles CS 4810

2/7/2023 Tues/Thurs 11:10am

Assignment 1-2 Report

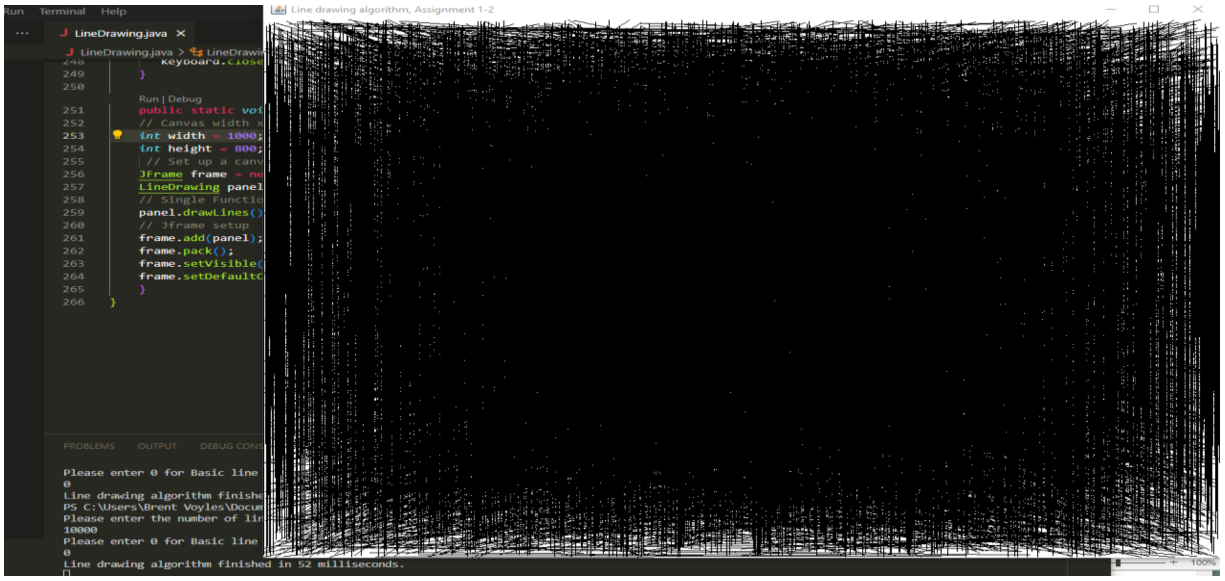
Basic Line Drawing Algorithm

1000 lines drawn in 9 milliseconds.



```
Please ente
1
Line drawin
PS C:\Users
Please ente
1000
Please ente
0
Line drawing algorithm finished in 9 milliseconds.
```

10,000 lines drawn in 52 milliseconds.

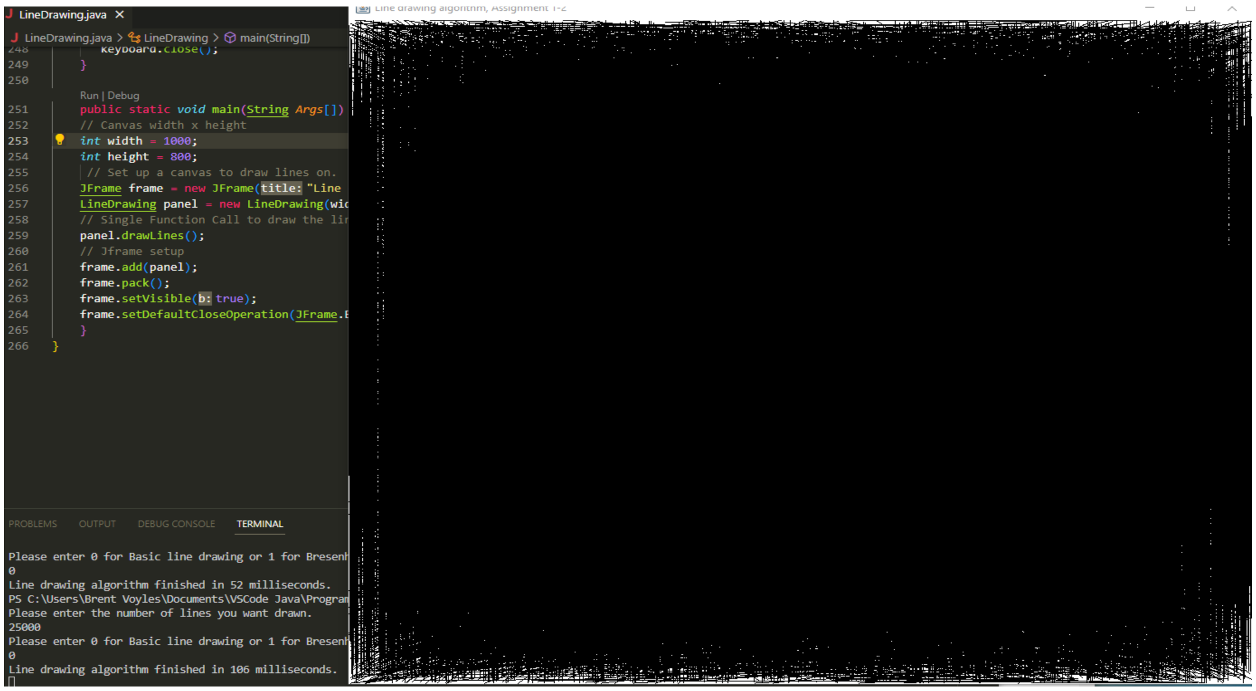

Run Terminal Help
LineDrawing.java
LineDrawing.java > LineDrawi
Line drawing algorithm, Assignment 1-2
Run | Debug
PROBLEMS OUTPUT DEBUG CONS
100%

```
        keyboard.close
    }

    public static voi
    // Canvas width x
    int width = 1000;
    int height = 800;
     // Set up a canv
    JFrame frame = ne
    LineDrawing panel
    // Single Functio
    panel.drawLines()
    // Jframe setup
    frame.add(panel);
    frame.pack();
    frame.setVisible(
    frame.setDefaultC
    }
}
```

```
Please enter 0 for Basic line
0
Line drawing algorithm finishe
PS C:\Users\Brent Voyles\Docum
Please enter the number of lir
10000
Please enter 0 for Basic line
0
Line drawing algorithm finished in 52 milliseconds.
```

25000 lines drawn in 106 milliseconds.


LineDrawing.java
LineDrawing.java > LineDrawing > main(String[])
Line drawing algorithm, Assignment 1-2
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

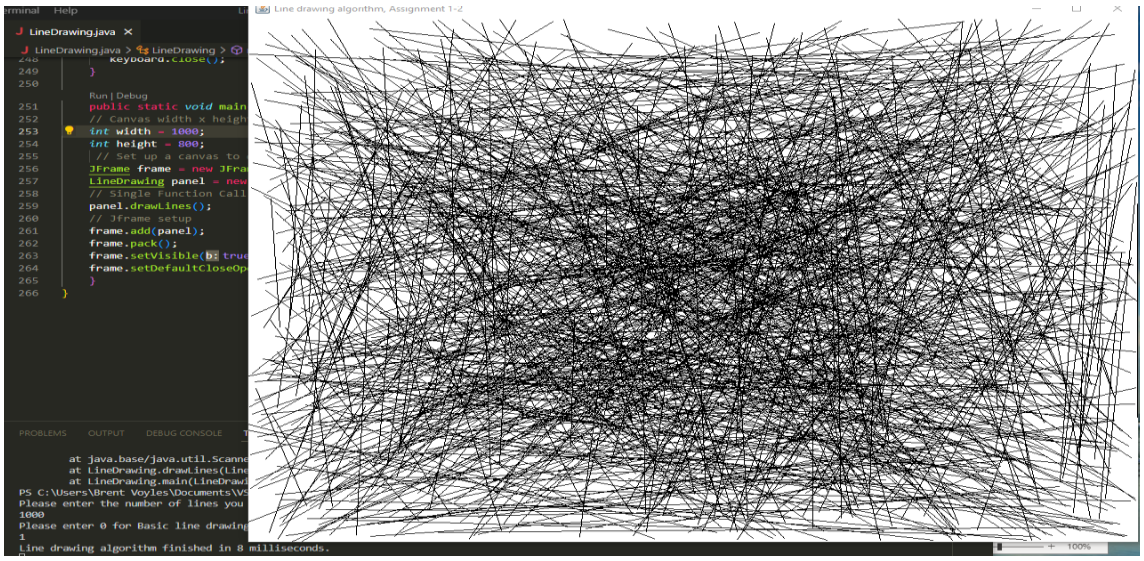
```
        keyboard.close();
    }

    Run | Debug
    public static void main(String Args[])
    // Canvas width x height
    int width = 1000;
    int height = 800;
     // Set up a canvas to draw lines on.
    JFrame frame = new JFrame(title: "Line
    LineDrawing panel = new LineDrawing(wid
    // Single Function Call to draw the lin
    panel.drawLines();
    // Jframe setup
    frame.add(panel);
    frame.pack();
    frame.setVisible(b: true);
    frame.setDefaultCloseOperation(JFrame.E
    }
}
```

```
Please enter 0 for Basic line drawing or 1 for Bresenh
0
Line drawing algorithm finished in 52 milliseconds.
PS C:\Users\Brent Voyles\Documents\VSCode Java\Program
Please enter the number of lines you want drawn.
25000
Please enter 0 for Basic line drawing or 1 for Bresenh
0
Line drawing algorithm finished in 106 milliseconds.
```

Bresenham's Line Drawing Algorithm

1000 lines drawn in 8 milliseconds.

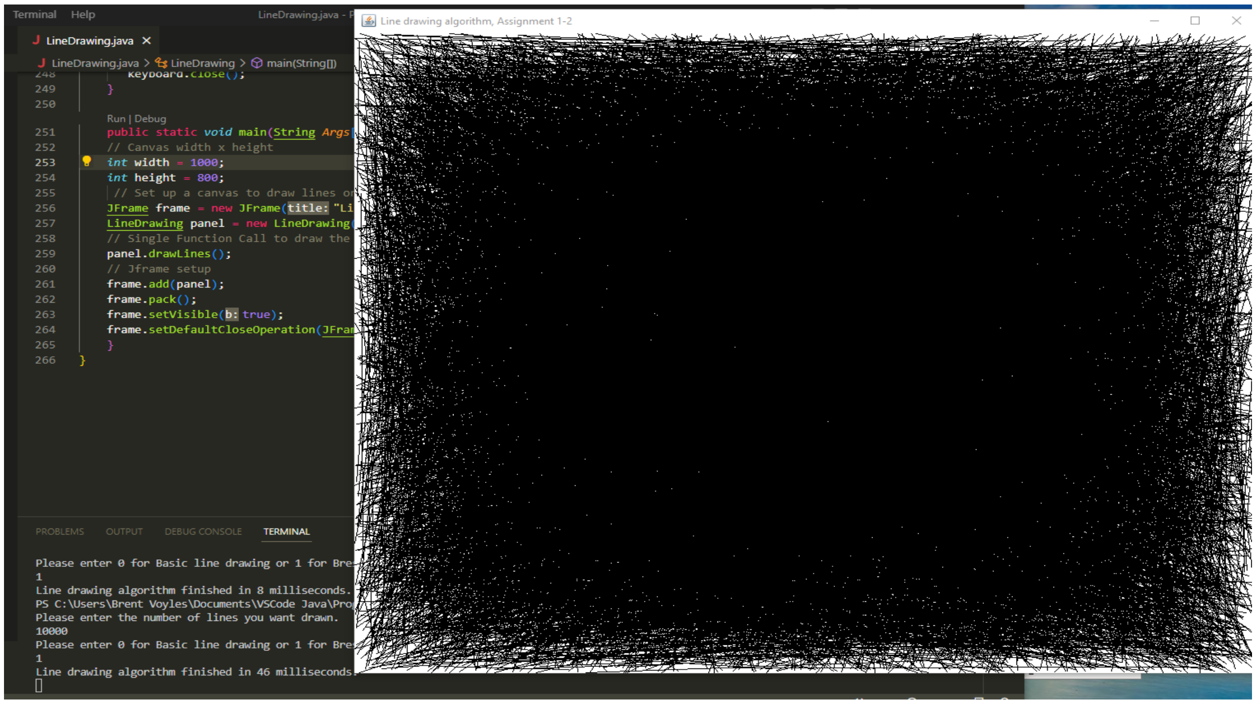

erminal Help
LineDrawing.java
LineDrawing.java > LineDrawing >
Line drawing algorithm, Assignment 1-2
PROBLEMS
OUTPUT
DEBUG CONSOLE
100%

```
        keyboard.close();
    }

    Run | Debug
    public static void main
    // Canvas width x heigh
    int width = 1000;
    int height = 800;
     // Set up a canvas to
    JFrame frame = new JFra
    LineDrawing panel = new
    // Single Function Call
    panel.drawLines();
    // Jframe setup
    frame.add(panel);
    frame.pack();
    frame.setVisible(b: true
    frame.setDefaultCloseOp
    }
}
```

```
        at java.base/java.util.Scanne
        at LineDrawing.drawLines(Line
        at LineDrawing.main(LineDrawi
PS C:\Users\Brent Voyles\Documents\VS
Please enter the number of lines you
1000
Please enter 0 for Basic line drawing
1
Line drawing algorithm finished in 8 milliseconds.
```

10,000 lines drawn in 46 milliseconds.

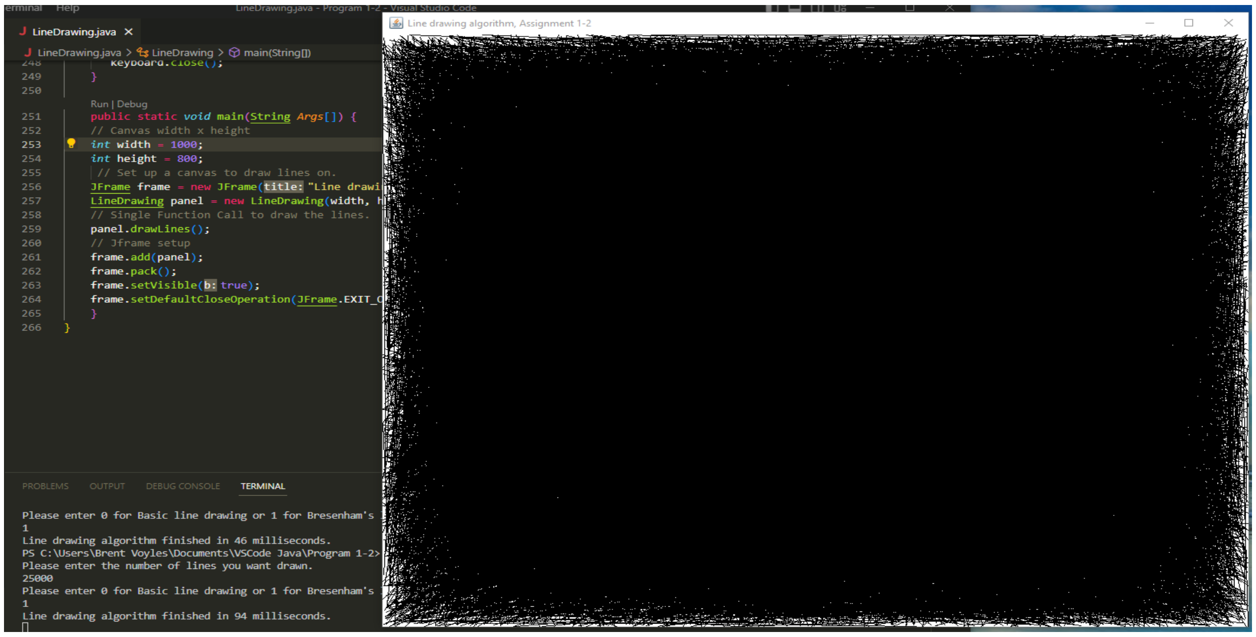


```
        keyboard.close();
    }

    public static void main(String Args
    // Canvas width x height
    int width = 1000;
    int height = 800;
     // Set up a canvas to draw lines or
    JFrame frame = new JFrame(title: "Li
    LineDrawing panel = new LineDrawing(
    // Single Function Call to draw the
    panel.drawLines();
    // Jframe setup
    frame.add(panel);
    frame.pack();
    frame.setVisible(b: true);
    frame.setDefaultCloseOperation(JFram
    }
}
```

```
Please enter 0 for Basic line drawing or 1 for Bre
1
Line drawing algorithm finished in 8 milliseconds.
PS C:\Users\Brent Voyles\Documents\VSCode Java\Prog
Please enter the number of lines you want drawn.
10000
Please enter 0 for Basic line drawing or 1 for Bre
1
Line drawing algorithm finished in 46 milliseconds.
▯
```

25,000 lines drawn in 94 milliseconds.



erminal
Help
LineDrawing.java - Program 1-2 - Visual Studio Code
LineDrawing.java
LineDrawing.java > LineDrawing > main(String[])
Run | Debug
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Line drawing algorithm, Assignment 1-2

```
        keyboard.close();
    }

    public static void main(String Args[]) {
    // Canvas width x height
    int width = 1000;
    int height = 800;
     // Set up a canvas to draw lines on.
    JFrame frame = new JFrame(title: "Line drawi
    LineDrawing panel = new LineDrawing(width, h
    // Single Function Call to draw the lines.
    panel.drawLines();
    // Jframe setup
    frame.add(panel);
    frame.pack();
    frame.setVisible(b: true);
    frame.setDefaultCloseOperation(JFrame.EXIT_C
    }
}
```

```
Please enter 0 for Basic line drawing or 1 for Bresenham's
1
Line drawing algorithm finished in 46 milliseconds.
PS C:\Users\Brent Voyles\Documents\VSCode Java\Program 1-2>
Please enter the number of lines you want drawn.
25000
Please enter 0 for Basic line drawing or 1 for Bresenham's
1
Line drawing algorithm finished in 94 milliseconds.
▯
```

Time taken for each algorithm to finish execution on a 1000x800 canvas.

Algorithm	**1,000 Lines**	**10,000 Lines**	**25,000 Lines**
Basic Line Algorithm	**9** Milliseconds	**52** Milliseconds	**106** Milliseconds
Bresenham's algorithm	**8** Milliseconds	**46** Milliseconds	**94** Milliseconds

Time taken for each algorithm to finish execution on a 1400x1000 canvas.

Algorithm	**1,000 Lines**	**10,000 Lines**	**25,000 Lines**
Basic Line Algorithm	**13** Milliseconds	**72** Milliseconds	**151** Milliseconds
Bresenham's algorithm	**11** Milliseconds	**60** Milliseconds	**125** Milliseconds

Based on the results, it is evident that the Bresenham line drawing algorithm executes faster than the basic line drawing algorithm in almost every scenario. When drawing many lines (10,000/25,000 lines), the Bresenham algorithm executes even faster than the basic line drawing algorithm. At 1000 lines, the Bresenham line drawing algorithm executes 1/9 of a millisecond faster or about 11% faster. When drawing 10,000 lines the Bresenham algorithm executes 6 milliseconds faster or about 11.5% faster. At 25,000 lines, the Bresenham algorithm finished execution 12 seconds before the Basic line drawing algorithm or around 12.22% faster. After noticing that the larger gap between execution times occurred at higher line numbers, I decided to do a couple more tests. The basic line-drawing algorithm finished 15,000 lines in 83 milliseconds whereas the Bresenham line drawing algorithm finished the same number of lines in 73 milliseconds. Next, I allowed both algorithms to draw 30,000 lines. The Bresenham algorithm was 10% faster and drew 30,000 lines in 125 milliseconds compared to the basic line drawing algorithm taking 139 milliseconds. The next thing I decided to test was if the length of lines made a difference in execution times. When I made the canvas larger, the result was that longer lines could be generated. This increased execution time dramatically! When drawing 1000 lines on the bigger canvas (longer lines) the basic line drawing algorithm executed in 13 milliseconds and the Bresenham algorithm executed in 12 milliseconds. When increasing the number of lines to 25,000, the basic line drawing took 152 milliseconds compared to 106 milliseconds with shorter lines and the Bresenham algorithm took 125 milliseconds compared to 94 milliseconds with shorter lines. Overall, the Bresenham algorithm took less time for all of the tests I ran. The most dramatic increase in times came when I tested the algorithms with longer lines.