Overview

Chris Piech
CS106A, Stanford University
Final Exam Info

THE CS106A FINAL EXAM IS FRIDAY JUNE 8TH FROM 8:30AM TO 11:30AM PST.

Location

Last Name: A through H
William R. Hewlett 201

Last Name: I through Z
William R. Hewlett Teaching Center

Review Session

There will be an optional final review session this Wednesday at 7:30 pm in Educ 128. Hope to see you there!

What to bring

The exam is on computer. You should bring:
- A laptop (with bluebook installed) and charger
- The device you use for two-step authentication
- Paper notes (it's open book)
- A power strip/extension cord (optional, but recommended if you have access to one)

Practice

Solutions will be posted Wednesday. Note that the BlueBook practice exam includes a file called "Instructions".

Final Win 2017
Paper Practice
Extra Practice
Paper Practice
Final Win 2018
Bluebook Pract

BlueBook

Like the midterm, the final exam is administered on a digital tool called BlueBook. If you still have bluebook from the midterm, skip this section. If you have a new laptop, please make sure to download and install BlueBook on your laptop before the exam.

- Mac download: Mac
- PC download: PC
- Linux download: Linux

Note: If you're using a Mac and you get an error saying that the Disk Image is from an unidentified developer, don't panic! Simply open up the Mac-BlueBook-1.0.0.dmg file in your finder, and right click it and select 'open'. The same window will pop up, but this time you'll have a chance to open it anyway. On Windows, if you get a message that says, "Windows protected your PC," you can click on "More Info" and then "Run anyway".

A practice exam that can be run on BlueBook can be downloaded above. This exam will be run under timed conditions, and give you an idea of what to expect for the actual exam.

Other Resources

Exam Strategies
Plan for today

• Announcements/Exam logistics
• Overview
• Tracing
• 1D Arrays
• 2D Arrays
• ArrayList
• Montage
<table>
<thead>
<tr>
<th><strong>Model</strong></th>
<th><strong>String</strong></th>
<th><strong>Array</strong></th>
<th><strong>2D Array</strong></th>
<th><strong>ArrayList</strong></th>
<th><strong>HashMap</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>Sequence of letters or symbols</td>
<td>Fixed length elements in a list</td>
<td>Grid / Matrix of elements</td>
<td>Growable list of elements</td>
<td>Key/Value mapping</td>
</tr>
<tr>
<td><strong>Type of element</strong></td>
<td>chars</td>
<td>Objects &amp; Primitives</td>
<td>Objects &amp; Primitives</td>
<td>Objects</td>
<td>Object/Object</td>
</tr>
<tr>
<td><strong>Access Elements</strong></td>
<td><code>str.charAt(i);</code></td>
<td><code>arr[i];</code></td>
<td><code>arr[r][c];</code></td>
<td><code>list.get(i);</code></td>
<td><code>map.put(key, value)</code></td>
</tr>
<tr>
<td><strong>Special notes</strong></td>
<td>Immutable</td>
<td>Watch bounds!</td>
<td>Row, col structure</td>
<td>Just fantastic</td>
<td>Each key must be unique. Unordered</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>“Hello world”</td>
<td>Histogram</td>
<td>ImageShop pixels</td>
<td>Hangman words, entries in namesurfer</td>
<td>NSDatabase, FPDatabase</td>
</tr>
<tr>
<td>Model</td>
<td>String</td>
<td>Array</td>
<td>2D Array</td>
<td>ArrayList</td>
<td>HashMap</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------</td>
<td>--------------------------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>Sequence of letters or</td>
<td>Fixed length elements in a</td>
<td>Grid / Matrix of elements</td>
<td>Growable list of elements</td>
<td>Key/Value mapping</td>
</tr>
<tr>
<td></td>
<td>symbols</td>
<td>list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of element</td>
<td>chars</td>
<td>Objects &amp; Primitives</td>
<td>Objects &amp; Primitives</td>
<td>Objects</td>
<td>Objects/Primitives</td>
</tr>
<tr>
<td>Access Elements</td>
<td>str.charAt(i);</td>
<td>arr[i];</td>
<td>arr[r][c];</td>
<td>list.get(i);</td>
<td>map.put(key, value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>list.set(i, elem)</td>
<td>map.get(key);</td>
</tr>
<tr>
<td>Special notes</td>
<td>Immutable</td>
<td>Watch bounds!</td>
<td>Row, col structure</td>
<td>Just fantastic</td>
<td>Each key must be unique.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unordered</td>
</tr>
<tr>
<td>Examples</td>
<td>”Hello world”</td>
<td>Histogram</td>
<td>ImageShop pixels</td>
<td>Hangman words, entries</td>
<td>NSDatabase, FPDatabase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>in namesurfer</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>String</td>
<td>Array</td>
<td>2D Array</td>
<td>ArrayList</td>
<td>HashMap</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------</td>
<td>-------------------------------------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Model</td>
<td>Sequence of letters or symbols</td>
<td>Fixed length elements in a list</td>
<td>Grid / Matrix of elements</td>
<td>Growable list of elements</td>
<td>Key/Value mapping</td>
</tr>
<tr>
<td>Type of element</td>
<td>chars</td>
<td>Objects &amp; Primitives</td>
<td>Objects &amp; Primitives</td>
<td>Objects</td>
<td>Object/Object</td>
</tr>
<tr>
<td>Access Elements</td>
<td>str.charAt(i); arr[i];</td>
<td>arr[r][c];</td>
<td>list.get(i); list.set(i, elem);</td>
<td>list.add(elem)</td>
<td>map.put(key, value); map.get(key);</td>
</tr>
<tr>
<td>Special notes</td>
<td>Immutable</td>
<td>Watch bounds!</td>
<td>Row, col structure</td>
<td>Just fantastic</td>
<td>Each key must be unique. Unordered</td>
</tr>
<tr>
<td>Examples</td>
<td>“Hello world”</td>
<td>Histogram</td>
<td>ImageShop pixels</td>
<td>Hangman words, entries in namesurfer</td>
<td>NSDatabase, FPDatabase</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>String</td>
<td>Array</td>
<td>2D Array</td>
<td>ArrayList</td>
<td>HashMap</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-------</td>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Sequence of letters or symbols</td>
<td>Fixed length elements in a list</td>
<td>Grid / Matrix of elements</td>
<td>Growable list of elements</td>
<td>Key/Value mapping</td>
</tr>
<tr>
<td><strong>Type of element</strong></td>
<td>chars</td>
<td>Objects &amp; Primitives</td>
<td>Objects &amp; Primitives</td>
<td>Objects</td>
<td>Object/Object</td>
</tr>
<tr>
<td><strong>Access Elements</strong></td>
<td><code>str.charAt(i);</code></td>
<td><code>arr[i];</code></td>
<td><code>arr[r][c];</code></td>
<td><code>list.get(i);</code>&lt;br&gt;<code>list.set(i, elem)</code>&lt;br&gt;<code>list.add(elem)</code></td>
<td><code>map.put(key, value)</code>&lt;br&gt;<code>map.get(key);</code></td>
</tr>
<tr>
<td><strong>Special notes</strong></td>
<td>Immutable</td>
<td>Watch bounds!</td>
<td>Row, col structure</td>
<td>Just fantastic</td>
<td>Each key must be unique. Unordered</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>“Hello world”</td>
<td>Histogram</td>
<td>ImageShop pixels</td>
<td>Hangman words, entries in namesurfer</td>
<td>NSDatabase, FPDatabase</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>Array</td>
<td>2D Array</td>
<td>ArrayList</td>
<td>HashMap</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Sequence of letters or symbols</td>
<td>Fixed length elements in a list</td>
<td>Grid / Matrix of elements</td>
<td>Growable list of elements</td>
<td>Key/Value mapping</td>
</tr>
<tr>
<td><strong>Type of element</strong></td>
<td>chars</td>
<td>Objects &amp; Primitives</td>
<td>Objects &amp; Primitives</td>
<td>Objects</td>
<td>Object/Object</td>
</tr>
<tr>
<td><strong>Access Elements</strong></td>
<td><code>str.charAt(i)</code>;</td>
<td><code>arr[i]</code>;</td>
<td><code>arr[r][c]</code>;</td>
<td><code>list.get(i)</code>;</td>
<td><code>map.put(key, value)</code>;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><code>list.set(i, elem)</code></td>
<td><code>map.get(key)</code>;</td>
</tr>
<tr>
<td><strong>Special notes</strong></td>
<td>Immutable</td>
<td>Watch bounds!</td>
<td>Row, col structure</td>
<td>Just fantastic</td>
<td>Each key must be unique. Unordered</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>&quot;Hello world&quot;</td>
<td>Histogram</td>
<td>ImageShop pixels</td>
<td>Hangman words, entries in namesurfer</td>
<td>NSDatabase, FPDatabase</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>Array</td>
<td>2D Array</td>
<td>ArrayList</td>
<td>HashMap</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------</td>
<td>----------------------------</td>
<td>----------------------------</td>
<td>---------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Model</td>
<td>Sequence of letters or symbols</td>
<td>Fixed length elements in a list</td>
<td>Grid / Matrix of elements</td>
<td>Growable list of elements</td>
<td>Key/Value mapping</td>
</tr>
<tr>
<td>Type of element</td>
<td>chars</td>
<td>Objects &amp; Primitives</td>
<td>Objects &amp; Primitives</td>
<td>Objects</td>
<td>Object/Object</td>
</tr>
<tr>
<td>Access Elements</td>
<td><strong>str.charAt(i);</strong></td>
<td><strong>arr[i];</strong></td>
<td><strong>arr[r][c];</strong></td>
<td><strong>list.get(i);</strong></td>
<td><strong>map.put(key, value)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>list.set(i, elem)</strong></td>
<td><strong>map.get(key);</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>list.add(elem)</strong></td>
<td></td>
</tr>
<tr>
<td>Special notes</td>
<td>Immutable</td>
<td>Watch bounds!</td>
<td>Row, col structure</td>
<td>Just fantastic</td>
<td>Each key must be unique.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unordered</td>
</tr>
<tr>
<td>Examples</td>
<td>&quot;Hello world&quot;</td>
<td>Histogram</td>
<td>ImageShop pixels</td>
<td>Hangman words, entries in namesurfer</td>
<td>NSDatabase, SteamTunnel</td>
</tr>
</tbody>
</table>
Strings under the hood are 1D Array of chars

String str = “cs106a”;

<table>
<thead>
<tr>
<th>'c'</th>
<th>'s'</th>
<th>'1'</th>
<th>'0'</th>
<th>'6'</th>
<th>'a'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
int[][] a = new int[3][4];

Outer array

<table>
<thead>
<tr>
<th>a[0][0]</th>
<th>a[0][1]</th>
<th>a[0][2]</th>
<th>a[0][3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[1][0]</td>
<td>a[1][1]</td>
<td>a[1][2]</td>
<td>a[1][3]</td>
</tr>
</tbody>
</table>
Tracing
• **Primitives**: int, double, boolean, char, ...
• **Objects**: GRect, GOval, GLine, int[], ... (anything with new, and that you call methods on)
• When passing parameters, make a copy of whatever is on the stack.

• **Primitives:** the *actual value* is on the stack (pass by value)

• **Objects:** a *memory address* where the information lives is on the stack. (pass by reference)
public void run() {
    int x = 2;
    addTwo(x);
    println(x);  // x is still 2!
}

private void addTwo(int y) {
    y += 2;
}
public void run() {
    GRect rect = new GRect(0,0,50,50);
    fillBlue(rect);
    add(rect);    // rect is blue!
}

private void fillBlue(GRect rect) {
    rect.setFilled(true);
    rect.setColor(Color.BLUE);
}
private void mystery(int[][] arr) {
    bloom(arr);
    frolic(arr[1][1]);
}

private void bloom(int[][] field) {
    for(int i = 0; i < field[0].length; i++) {
        field[0][i] += field[0][i + 1];
    }
}

private void frolic(int num) {
    int birds = num * 2;
    int bees = num % 2;
    num = birds + bees;
}
```java
private void mystery(int[][] arr) {
    bloom(arr);
    arr[1][1] = frolic(arr[1][1]);
}

private void bloom(int[][] field) {
    for(int i = 0; i < field[0].length; i++) {
        field[0][i] += field[0][i + 1];
    }
}

private int frolic(int num) {
    int birds = num * 2;
    int bees = num % 2;
    return birds + bees;
}
```

Input to **mystery()**
What is `arr` after?

Take 2
2D Arrays
// return the maximum value in the matrix
private double getMax(double[][][] matrix) {

// return the maximum value in the matrix
private double getMax(double[][] matrix) {
    double maxValue = matrix[0][0];
    for(int r = 0; r < matrix.length; r++) {
        for(int c = 0; c < matrix[0].length; c++) {
            if(matrix[r][c] > maxValue) {
                maxValue = matrix[r][c];
            }
        }
    }
    return maxValue;
}
Get Max

// return the maximum value in the matrix
private double getMax(double[][] matrix) {
    double maxValue = matrix[0][0];
    for(int r = 0; r < matrix.length; r++) {
        for(int c = 0; c < matrix[0].length; c++) {
            if(matrix[r][c] > maxValue) {
                maxValue = matrix[r][c];
            }
        }
    }
    return maxValue;
}
private double getMax(double[][] matrix) {
    double maxValue = matrix[0][0];
    for(int r = 0; r < matrix.length; r++) {
        for(int c = 0; c < matrix[0].length; c++) {
            if(matrix[r][c] > maxValue) {
                maxValue = matrix[r][c];
            }
        }
    }
    return maxValue;
}
For each pixel chose one character to add to the corresponding row String.

- If brightness greater than 0.66, use ' '.
- If the brightness is greater than 0.33, use '1'.
- Else, you should use the character '0'.
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if(v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if(v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if (v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if(v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if(v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if(v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
Array List
• An **ArrayList** is a flexible-length list of a single type of thing.

• An ArrayList can only store **objects**.
  • For primitives use e.g. **ArrayList<Integer>** instead of **ArrayList<int>**. (**Integer** is a wrapper class for int)
  • Other wrapper classes: **Double** instead of double, **Character** instead of char, **Boolean** instead of boolean.

• An ArrayList has a variety of methods you can use like `.contains`, `.get`, `.add`, `.remove`, `.size`, etc.
Array vs ArrayList

• Array
  • Fixed size
  • Efficient (not a concern in this class)
  • No methods, can only use myArray.length (no parentheses!)
  • Can store any object or primitive

• ArrayList
  • Expandable
  • Less efficient than Array (not a concern in this class)
  • Convenient methods like .add(), .remove(), .contains()
  • Cannot store primitives, so use their wrapper classes instead
private void deleteDuplicates(ArrayList<String> list)

• Guaranteed that list is in sorted order
• {"be", "be", "is", "not", "or", "or", "or", "question", "that", "the", "to"} becomes {"be", "is", "not", "or", "question", "that", "the", "to"}

• Solution strategy:
  • Loop through ArrayList
  • Compare pairs of elements
  • If element.equals(nextElement), remove element from the list
deleteDuplicates()

List: 
{"be", "be", "is", "not", "or", "or", "or", "question", "that", "the", "to"}

Current Index (i): 0

curr  next
deleteDuplicates()

Current Index (i): 0

List: 
{"be", "be", "is", "not", "or", "or", "or", "question", "that", "the", "to"}
```python
def deleteDuplicates():
    List = ['be', 'be', 'is', 'not', 'or', 'or', 'or', 'question', 'that', 'the', 'to']

    Current Index (i): 0
```
def deleteDuplicates():

    List = ['be', 'is', 'not', 'or', 'or', 'or', 'question', 'that', 'the', 'to']

    Current Index (i): 0
```python
List = {"be", "is", "not", "or", "or", "or", "question", "that", "the", "to"}
```

Current Index (i): 1
deleteDuplicates()

List
{"be", "is", "not", "or", "or", "or", "question", "that", "the", "to"}

curr next

Current Index (i):  1
Sometime later...
deleteDuplicates()

List = \{"be", "is", "not", "or", "or", "or", "question", "that", "the", "to"\}

Current Index \((i)\): 3
deleteDuplicates()

List: 
{"be", "is", "not", "or", "or", "or", "question", "that", "the", "to"}

Current Index (i): 3
deleteDuplicates()

List

{"be", "is", "not", "or", "or", "or", "question", "that", "the", "to"}

Current Index (i): 3
deleteDuplicates()

0 1 2 3 4 5 6 7 8
List {"be", "is", "not", "or", "or", "question", "that", "the", "to"}

Current Index (i): 3
```python
def deleteDuplicates():
    return {'be', 'is', 'not', 'or', 'or', 'question', 'that', 'the', 'to'}
```

Current Index \(i\): 4
deleteDuplicates()

List
{"be", "is", "not", "or", "or", "question", "that", "the", "to"}

Current Index (i): 4
deleteDuplicates()

1. Loop through ArrayList
2. Compare pairs of elements
3. If `element.equals(nextElement)`, remove element from the list

```java
private void deleteDuplicates(ArrayList<String> list) {
    for (int i = 0; i < list.size() - 1; i++) {
        String elem = list.get(i);
        // If two adjacent elements are equal
        if (list.get(i + 1).equals(elem)) {
            list.remove(i);
            i--;  
        }
    }
}
```

Strategy #1
deleteDuplicates()

• Loop through ArrayList in reverse
• Compare pairs of elements
• If element.equals(previousElement), remove element from the list

```java
private void deleteDuplicatesReverse(ArrayList<String> list) {
    for (int i = list.size() - 1; i > 0; i--) {
        String elem = list.get(i);
        // If two adjacent elements are equal
        if (list.get(i - 1).equals(elem)) {
            list.remove(i);
        }
    }
}
```
```
private void deleteDuplicates(ArrayList<String> list) {
    // Make a new list with only the ones to keep
    ArrayList<String> newList = new ArrayList<String>();
    String last = null;
    for(String curr : newList) {
        if(!curr.equals(last)) {
            last = curr;
            newList.add(curr);
        }
    }
    // Repopulate the old list
    list.clear();
    for(String v : newList) {
        list.add(v);
    }
}
```
public void displayQuery(String query) {

Use a helper method:

ArrayList<GImage> results =
    getSearchResults(query);

display your images in three rows of fixed height ROW_HEIGHT. You can scale images, but should maintain the ratio of their width to height. You can change the size of a GImage using it’s

    setSize(width, height) method

There is a spacing of GAP pixels between each picture. You can optionally include the GAP between the pictures and the border of the window.

No image should go off the screen. You should not display all 100 returned images – only display the ones that fit into the three rows.
You have come a long way
Enter a number: 17
17 is odd, so I make 3n + 1: 52
52 is even so I take half: 26
26 is even so I take half: 13
13 is odd, so I make 3n + 1: 40
40 is even so I take half: 20
20 is even so I take half: 10
10 is even so I take half: 5
5 is odd, so I make 3n + 1: 16
16 is even so I take half: 8
8 is even so I take half: 4
4 is even so I take half: 2
2 is even so I take half: 1
The process took 12 to reach 1.
Welcome to Hangman
Your word looks like this: ---
You have 7 guesses left
Your guess: a
There are no a's in the word.
Your word looks like this: ---
You have 6 guesses left
Your guess: e
There are no e's in the word.
Your word looks like this: ---
You have 5 guesses left
Your guess: d
There are no d's in the word.
Your word looks like this: ---
You have 4 guesses left
Your guess: m
There are no m's in the word.
Your word looks like this: ---
You have 3 guesses left
Your guess: u
That guess is correct.
Your word looks like this: u--
You have 3 guesses left
Your guess: g
That guess is correct.
Your word looks like this: u-zz-
You have 3 guesses left
Your guess:
Start with an intro

Each question asks the user if they agree or disagree

Users must answer with a number in the range 1 and 6

There are 7 questions. To get

New dribbles are created at the top of the screen at the same x location as the mouse

If moving the active dribble down makes it collide with an old piece, or go below the screen, don’t move it. Instead create a new dribble.
By the numbers
7 hard assignments
14,000 person hours programming
350 pieces of fruit
1 class 😊
You have my respect.
Why Study CS?
Interdisciplinary
Now is the Time
Everyone is Welcome
The End
public void displayQuery(String query) {
    ArrayList<GImage> results = getSearchResults(query);
    int index = 0;
    int row = 0;
    int currX = GAP;
    int currY = GAP;
    while (row < 3) {
        GImage img = results.get(index);
        double ratio = img.getWidth() / img.getHeight();
        double width = ROW_HEIGHT * ratio;
        if (currX + width < getWidth()) {
            add(img, currX, currY);
            currX += width + GAP;
            index++;
        } else {
            row++;
            currX = GAP;
            currY += ROW_HEIGHT + GAP;
        }
    }
}