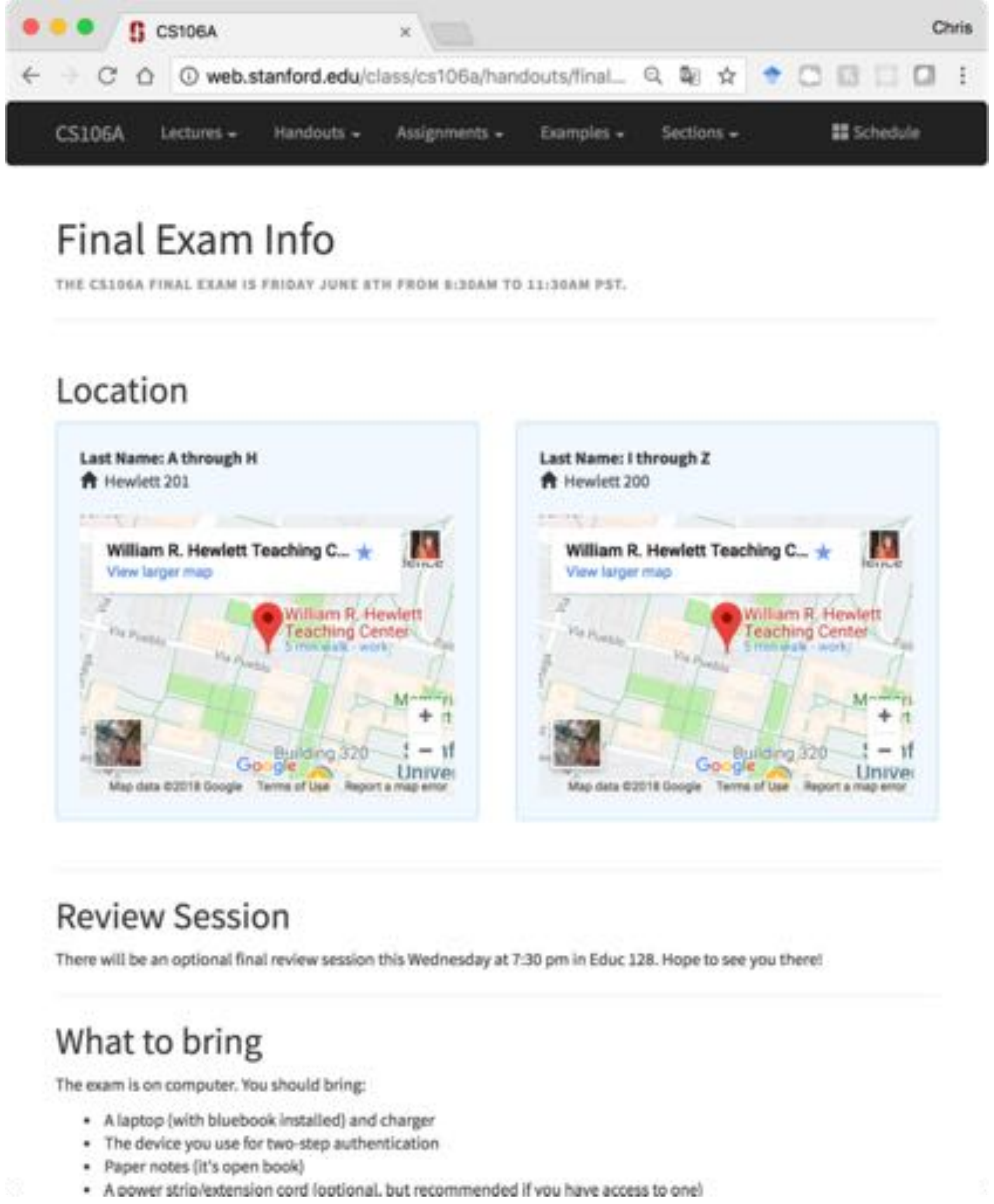
Overview

Chris Piech

CS106A, Stanford University

And if you scroll down….



Final Exam Info

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

Location

Last Name: A through H
Hewlett 201

Last Name: I through Z
Hewlett 200

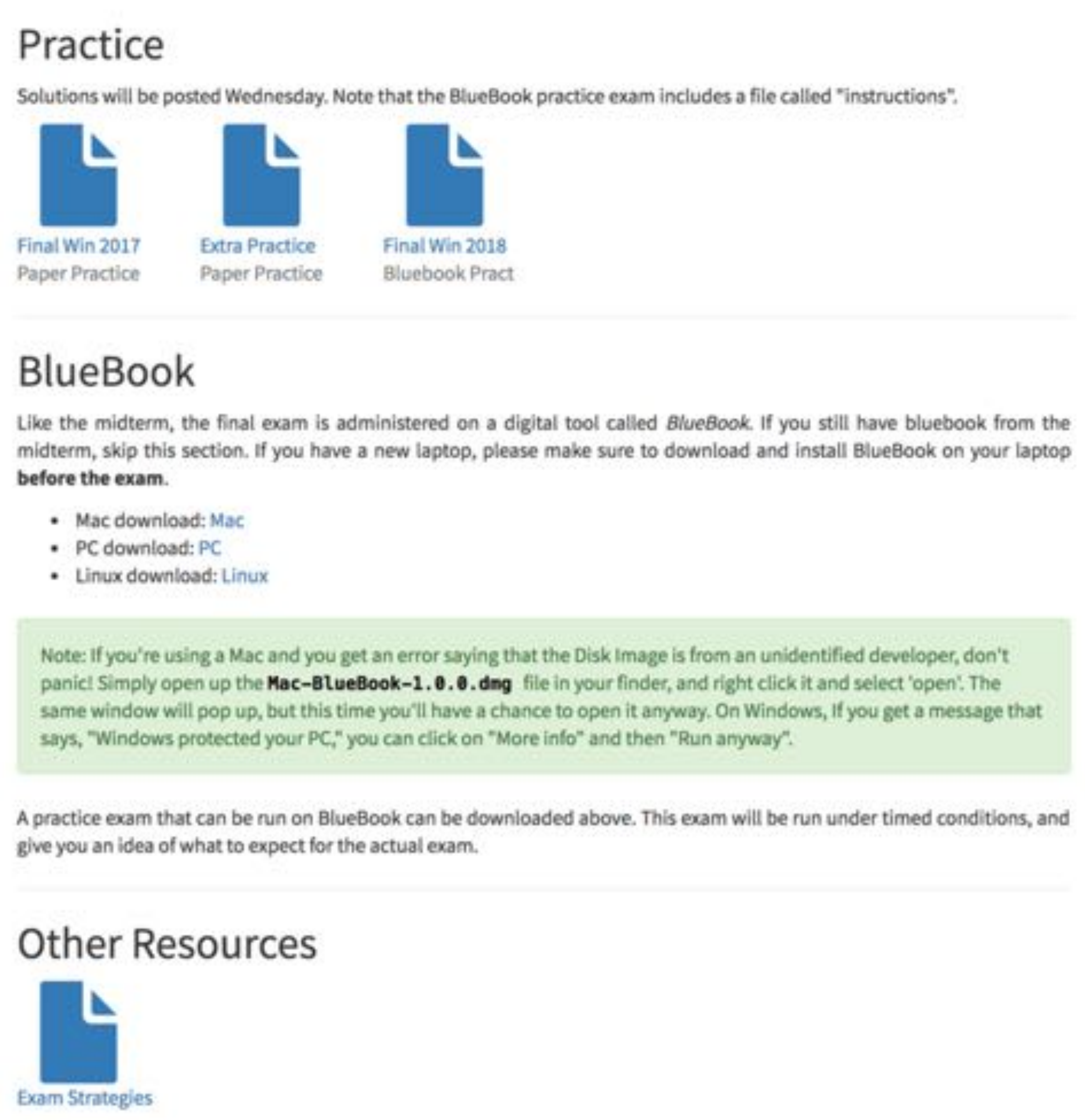
Review Session

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".

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

Plan for today

- Announcements/Exam logistics
- Overview
- Tracing
- 1D Arrays
- 2D Arrays
- ArrayList
- Montage

	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object
Access Elements	`str.charAt(i);`	`arr[i];`	`arr[r][c];`	`list.get(i);` `list.set(i, elem)` `list.add(elem)`	`map.put(key, value)` `map.get(key);`
Special notes	Immutable	Watch bounds!	Row, col structure	Just fantastic	Each key must be unique. Unordered
Examples	”Hello world”	Histogram	ImageShop pixels	Hangman words, entries in namesurfer	NSDatabase, FPDatabase

	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object
Access Elements	`str.charAt(i);`	`arr[i];`	`arr[r][c];`	`list.get(i);` `list.set(i, elem)` `list.add(elem)`	`map.put(key, value)` `map.get(key);`
Special notes	Immutable	Watch bounds!	Row, col structure	Just fantastic	Each key must be unique. Unordered
Examples	”Hello world”	Histogram	ImageShop pixels	Hangman words, entries in namesurfer	NSDatabase, FPDatabase

	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object
Access Elements	`str.charAt(i);`	`arr[i];`	`arr[r][c];`	`list.get(i);` `list.set(i, elem)` `list.add(elem)`	`map.put(key, value)` `map.get(key);`
Special notes	Immutable	Watch bounds!	Row, col structure	Just fantastic	Each key must be unique. Unordered
Examples	"Hello world"	Histogram	ImageShop pixels	Hangman words, entries in namesurfer	NSDatabase, FPDatabase

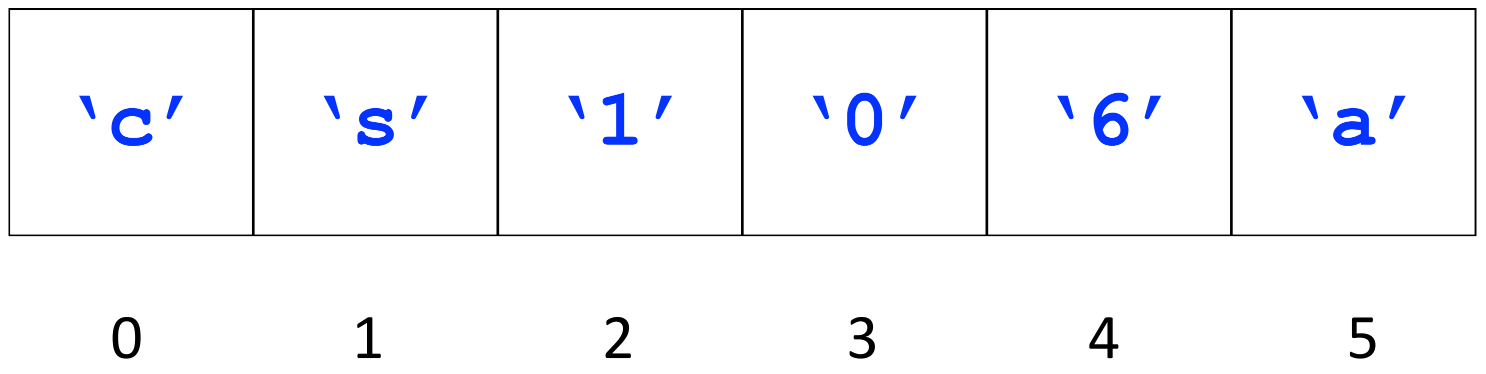
	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object
Access Elements	`str.charAt(i);`	`arr[i];`	`arr[r][c];`	`list.get(i);` `list.set(i, elem)` `list.add(elem)`	`map.put(key, value)` `map.get(key);`
Special notes	Immutable	Watch bounds!	Row, col structure	Just fantastic	Each key must be unique. Unordered
Examples	"Hello world"	Histogram	ImageShop pixels	Hangman words, entries in namesurfer	NSDatabase, FPDatabase

	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object
Access Elements	`str.charAt(i);`	`arr[i];`	`arr[r][c];`	`list.get(i);` `list.set(i, elem)` `list.add(elem)`	`map.put(key, value)` `map.get(key);`
Special notes	Immutable	Watch bounds!	Row, col structure	Just fantastic	Each key must be unique. Unordered
Examples	”Hello world”	Histogram	ImageShop pixels	Hangman words, entries in namesurfer	NSDatabase, FPDatabase

	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object
Access Elements	`str.charAt(i);`	`arr[i];`	`arr[r][c];`	`list.get(i);` `list.set(i, elem)` `list.add(elem)`	`map.put(key, value)` `map.get(key);`
Special notes	Immutable	Watch bounds!	Row, col structure	Just fantastic	Each key must be unique. Unordered
Examples	”Hello world”	Histogram	ImageShop pixels	Hangman words, entries in namesurfer	NSDatabase, SteamTunnel

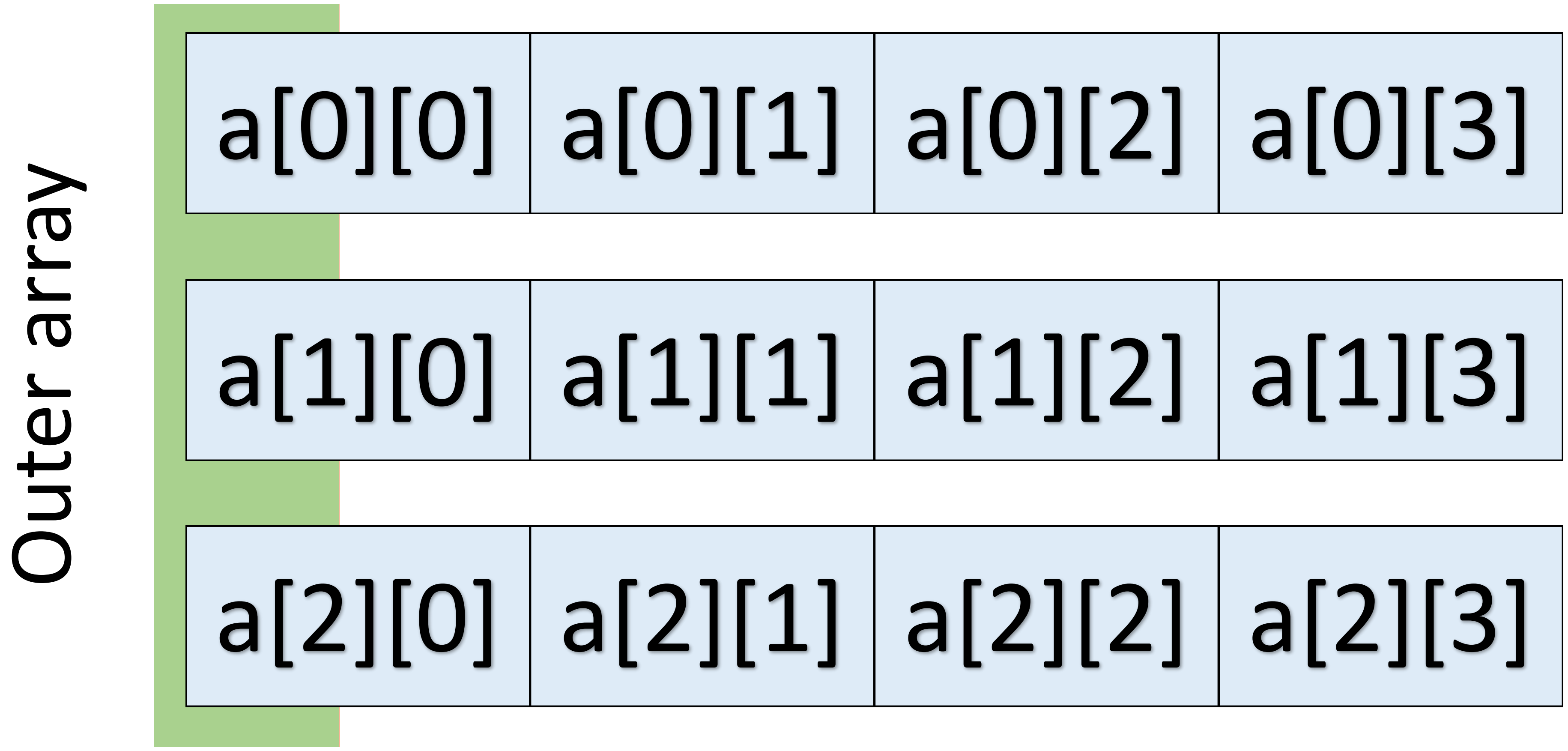
Strings under the hood are 1D Array of chars

```
String str = “cs106a”;
```



2D Arrays = Array of Arrays

```
int[][] a = new int[3][4];
```



Tracing

Primitives and Objects

- **Primitives:** int, double, boolean, char,…
- **Objects**: GRect, GOval, GLine, int[], … (anything with **new**, and that you call methods on)

Parameters

- **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)

Parameters: Primitives

```
public void run() {
	int x = 2;
	addTwo(x);
	println(x);  // x is still 2!
}

private void addTwo(int y) {
	y += 2;
}
```

Parameters: Objects

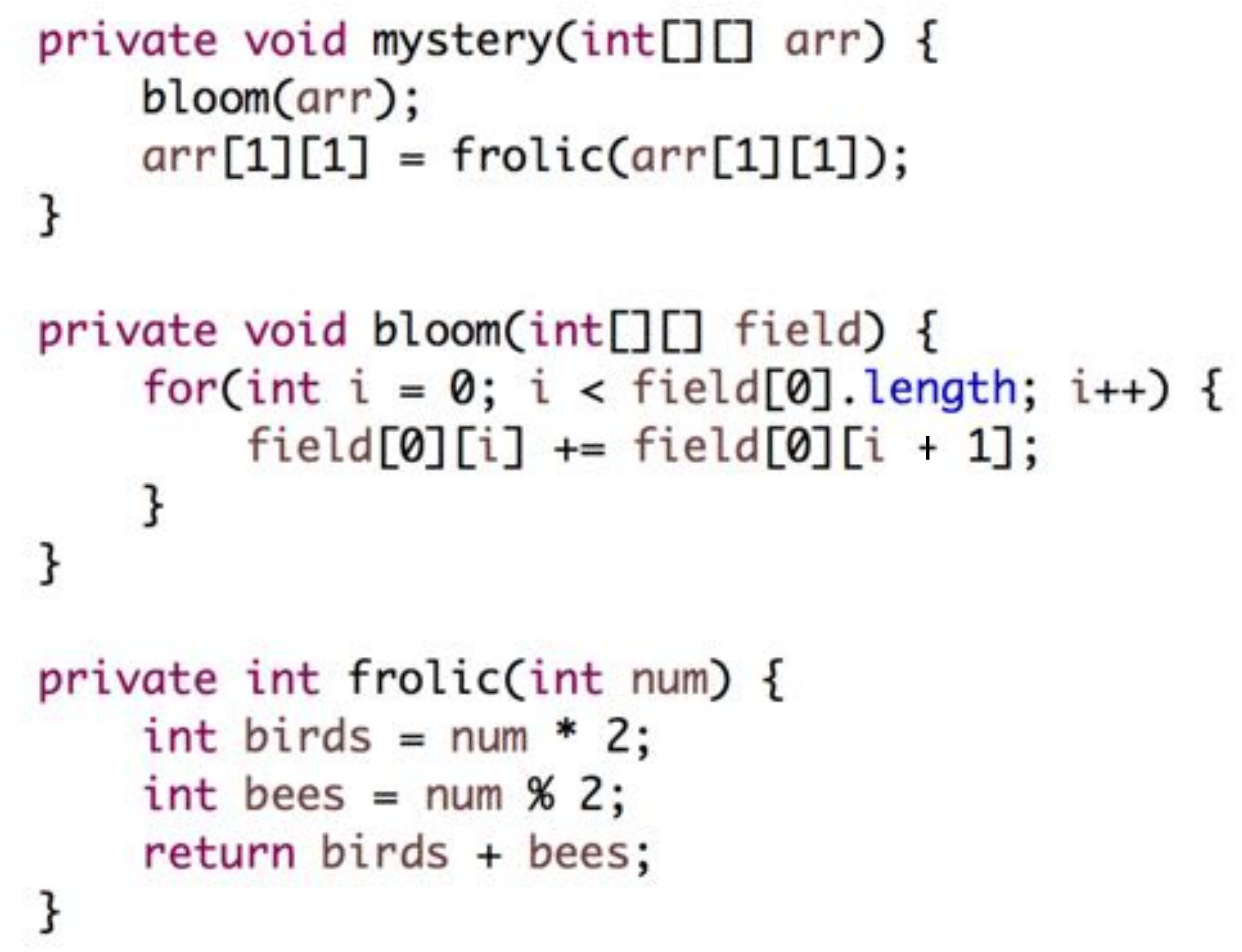
```
public void run() {
     GRect rect = new Grect(0,0,50,50);
     fillBlue(rect);
     add(rect);    // rect is blue!
}

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;
}
```

0	**1**	**2**
3	**4**	**5**

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

Take 1



```
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;
}
```

0	**1**	**2**
3	**4**	**5**

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

Take 2

2D Arrays

Get Max

```
// return the maximum value in the matrix
private double getMax(double[][] matrix) {
```

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;
}
```

Get Max

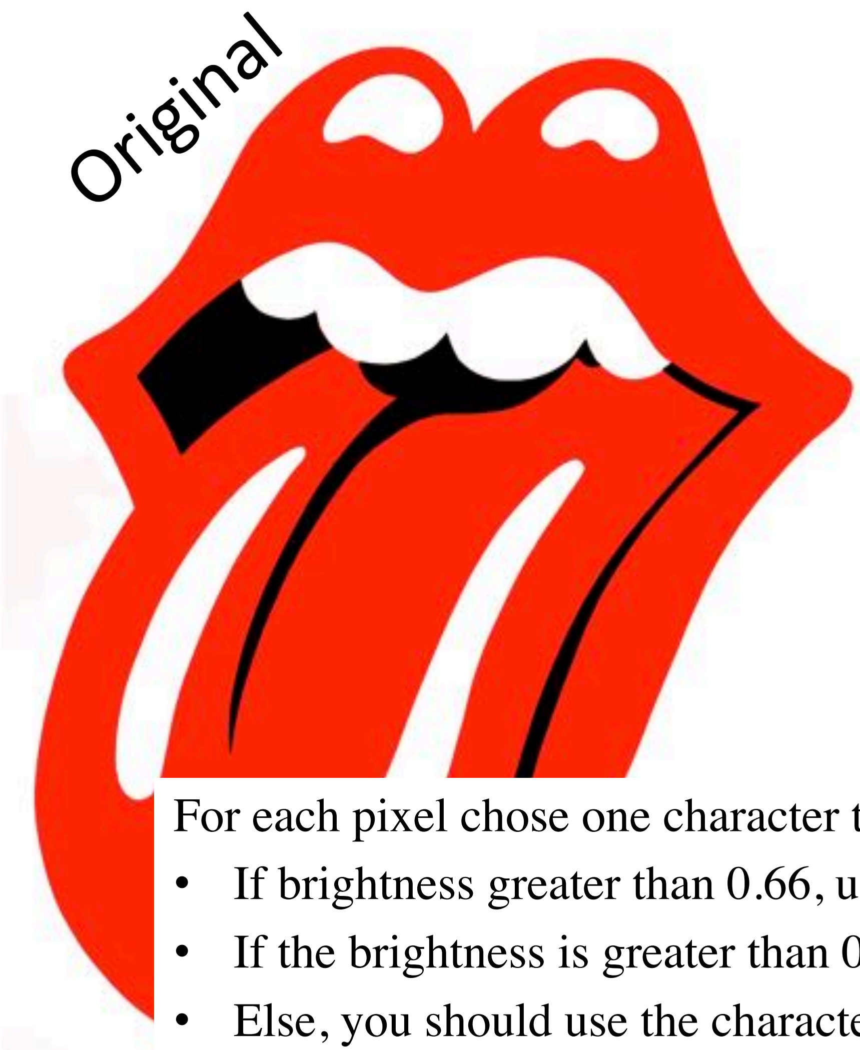
```
// return the maximum value in the matrix
private double getMax(double[][] matrix) {
  double maxValue = matrix[0][0];
  for(int r = 0; r < matrix.length; r++) {
    for(int c = 0; c < matrix[0].length; c++) {
      if(matrix[r][c] > maxValue) {
        maxValue = matrix[r][c];
      }
    }
  }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

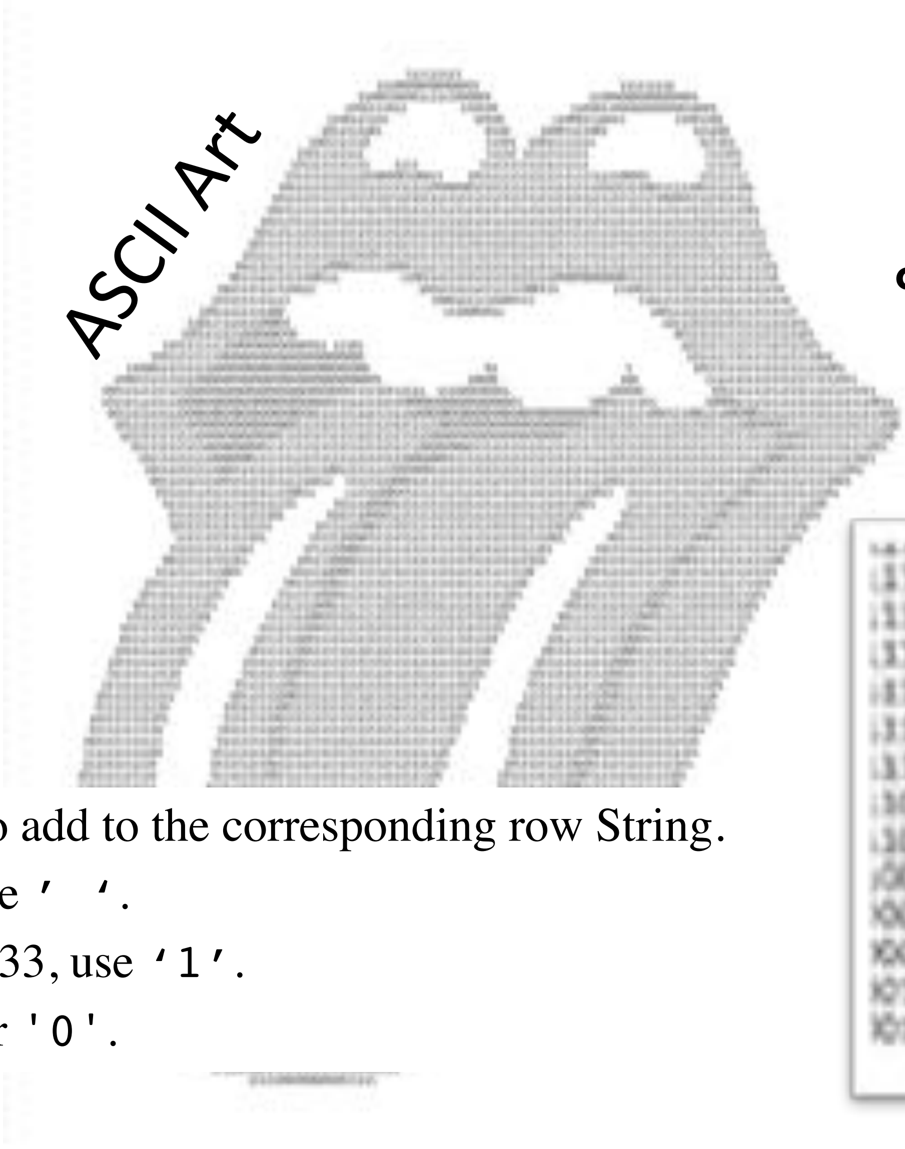
Get Max

```
// 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;
}
```

Make ASCII Art

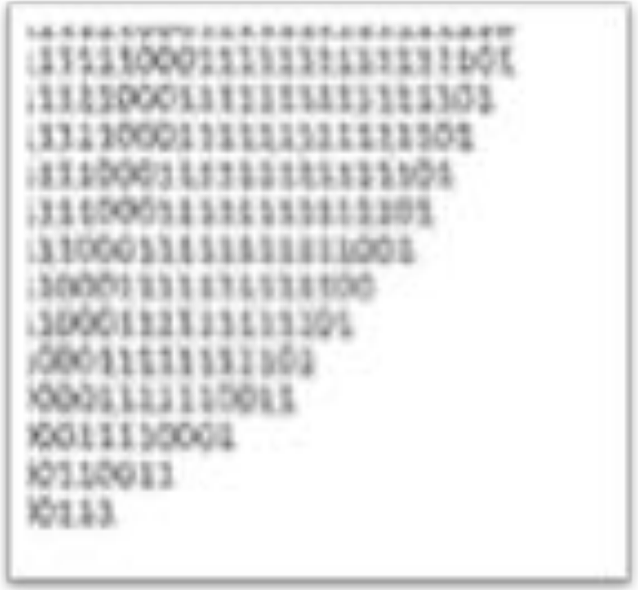
```
private String[] makeAscii(GImage img) {
```





Helper method

```
double[][] brightness =
   img.getPixelBrightness();
```



```
[illegible]
11111000111111111111101
1111100011111111111101
111100011111111111101
11110001111111111101
1111000111111111101
111000111111111001
11000111111111100
1100011111111101
100011111111101
10001111110011
1001111100001
101100011
10111
```

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

ArrayList

- 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

deleteDuplicates()

```
private void deleteDuplicates(ArrayList<String> list)
```

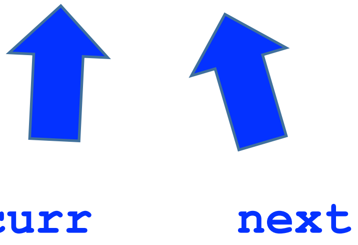
- Guaranteed that list is in sorted order
- {"be", "be", "is", "not", "or”, "or”, "or", "question", "that", "the", "to"} becomes {“be”, “is”, “not”, “or”, “question”, “that”, “the”, “to”}

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

deleteDuplicates()

0 1 2 3 4 5 6 7 8 9 10

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

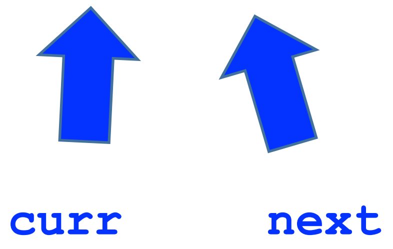


Current Index (**`i`**): 0

deleteDuplicates()

0 1 2 3 4 5 6 7 8 9 10

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



Current Index (`i`): 0

deleteDuplicates()

	0	1	2	3	4	5	6	7	8	9	10
List	{"be",	**"be"**,	"is",	"not",	"or”,	"or”,	"or",	"question",	"that",	"the",	"to"}

Current Index (`i`): 0

deleteDuplicates()

	0	1	2	3	4	5	6	7	8	9
List	{**"be"**,	"is",	"not",	"or”,	"or”,	"or",	"question",	"that",	"the",	"to"}

Current Index (`i`): 0

deleteDuplicates()

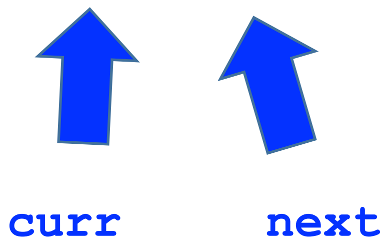
	0	1	2	3	4	5	6	7	8	9
List	{**"be"**,	"is",	"not",	"or”,	"or”,	"or",	"question",	"that",	"the",	"to"}

Current Index (**`i`**): 1

deleteDuplicates()

0 1 2 3 4 5 6 7 8 9

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

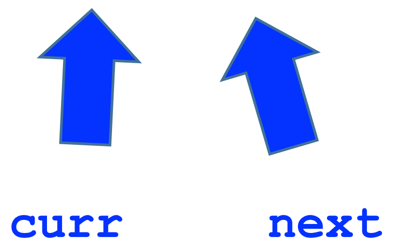


Current Index (**`i`**): 1

Sometime later…

deleteDuplicates()

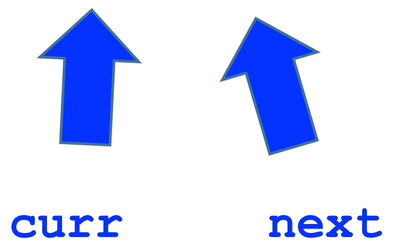
	0	1	2	3	4	5	6	7	8	9
List	{**"be"**,	"is",	"not",	"or”,	"or”,	"or",	"question",	"that",	"the",	"to"}



Current Index (`i`): 3

deleteDuplicates()

	0	1	2	3	4	5	6	7	8	9
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 9

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

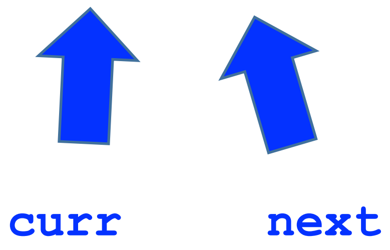
deleteDuplicates()

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

Current Index (`i`): **4**

deleteDuplicates()

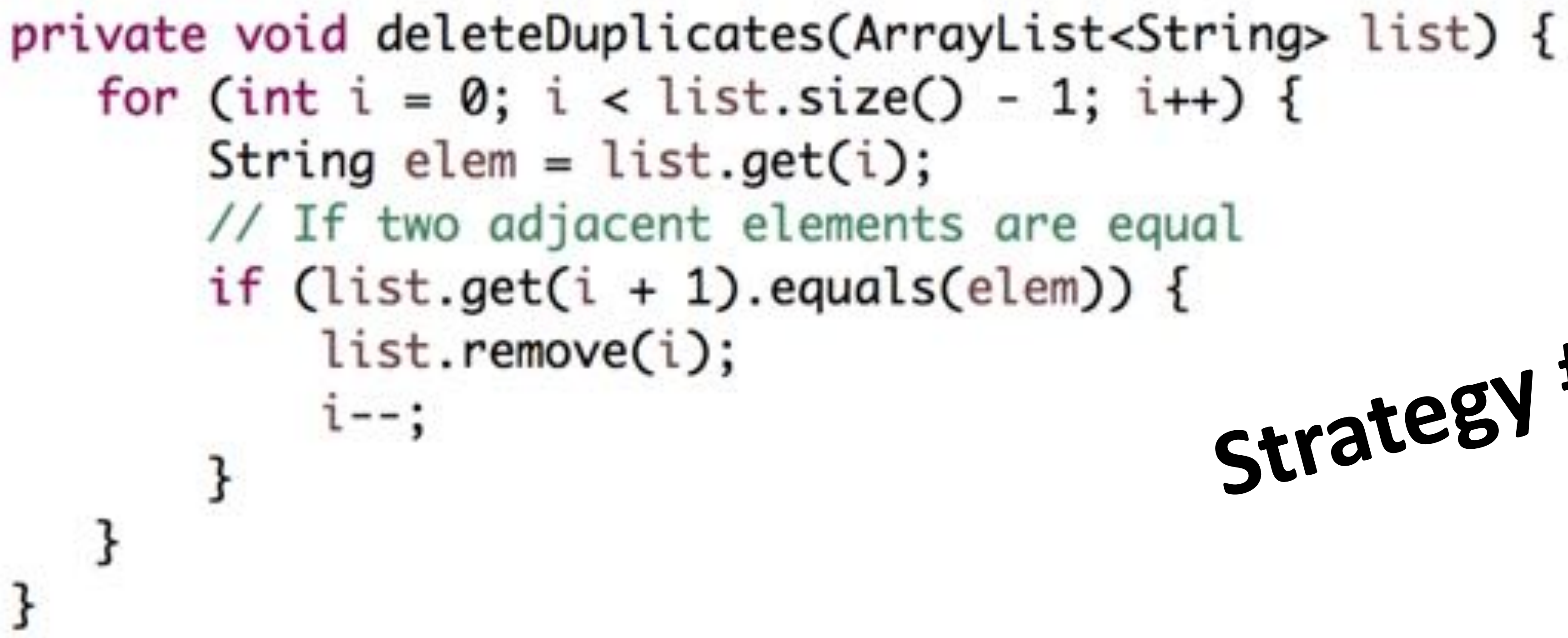
	0	1	2	3	4	5	6	7	8
List	{**"be"**,	"is",	"not",	**"or”**,	"or",	"question",	"that",	"the",	"to"}



Current Index (`i`): **4**

deleteDuplicates()

- Loop through ArrayList
- Compare pairs of elements
- If element.equals(nextElement), remove element from the list

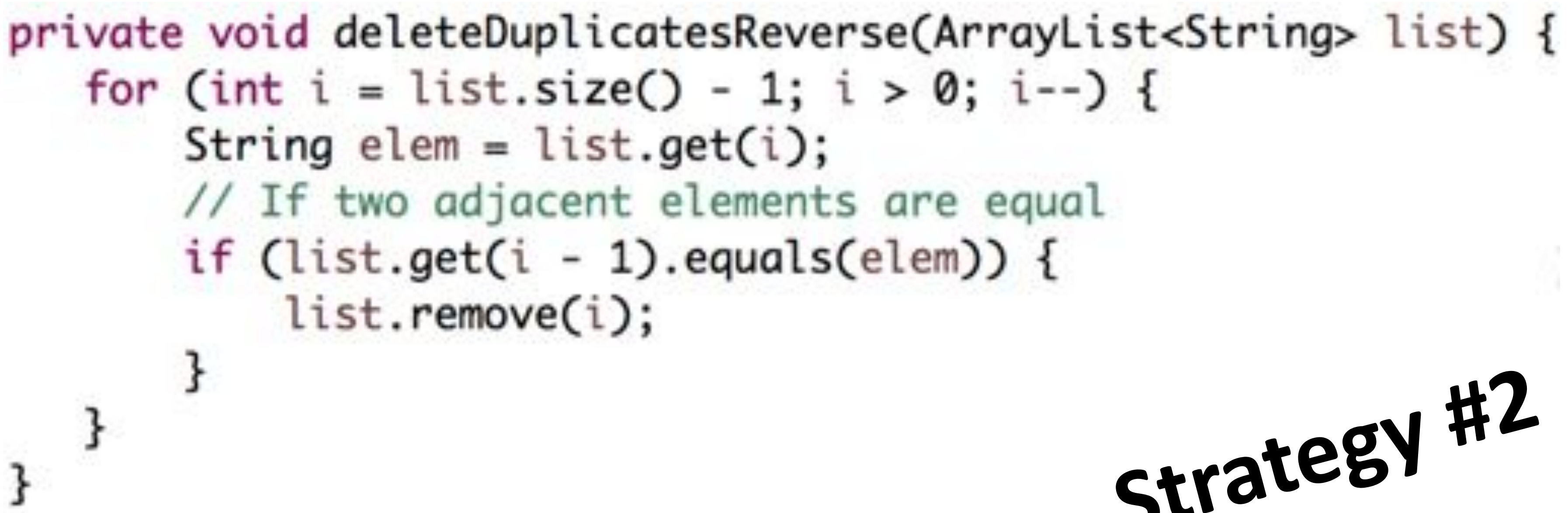


```java
private void deleteDuplicates(ArrayList<String> list) {
    for (int i = 0; i < list.size() - 1; i++) {
        String elem = list.get(i);
        // If two adjacent elements are equal
        if (list.get(i + 1).equals(elem)) {
            list.remove(i);
            i--;
        }
    }
}
```

Strategy #1

deleteDuplicates()

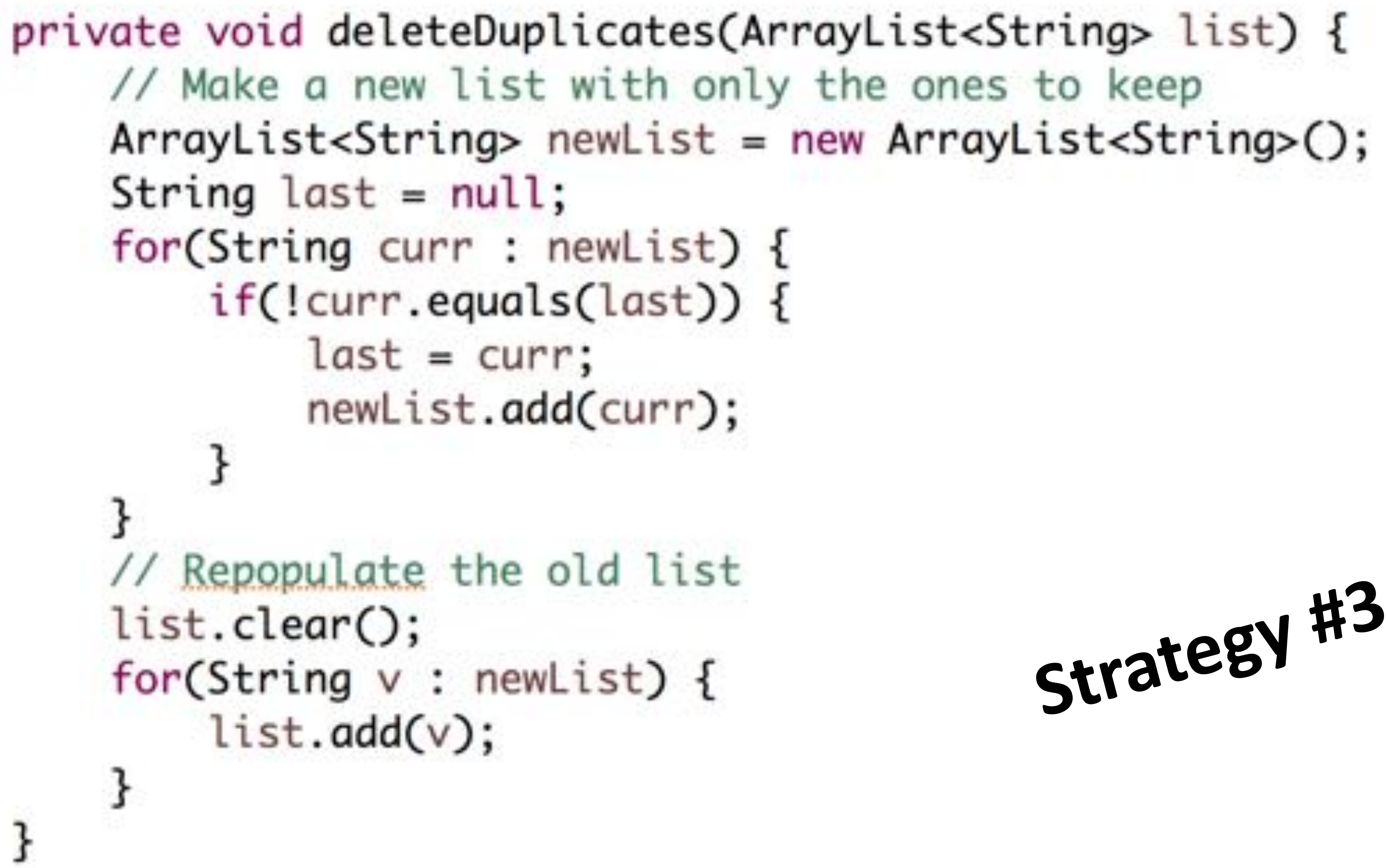
- Loop through ArrayList **in reverse**
- Compare pairs of elements
- If element.equals(**previous**Element), remove element from the list



```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);
        }
    }
}
```

Strategy #2

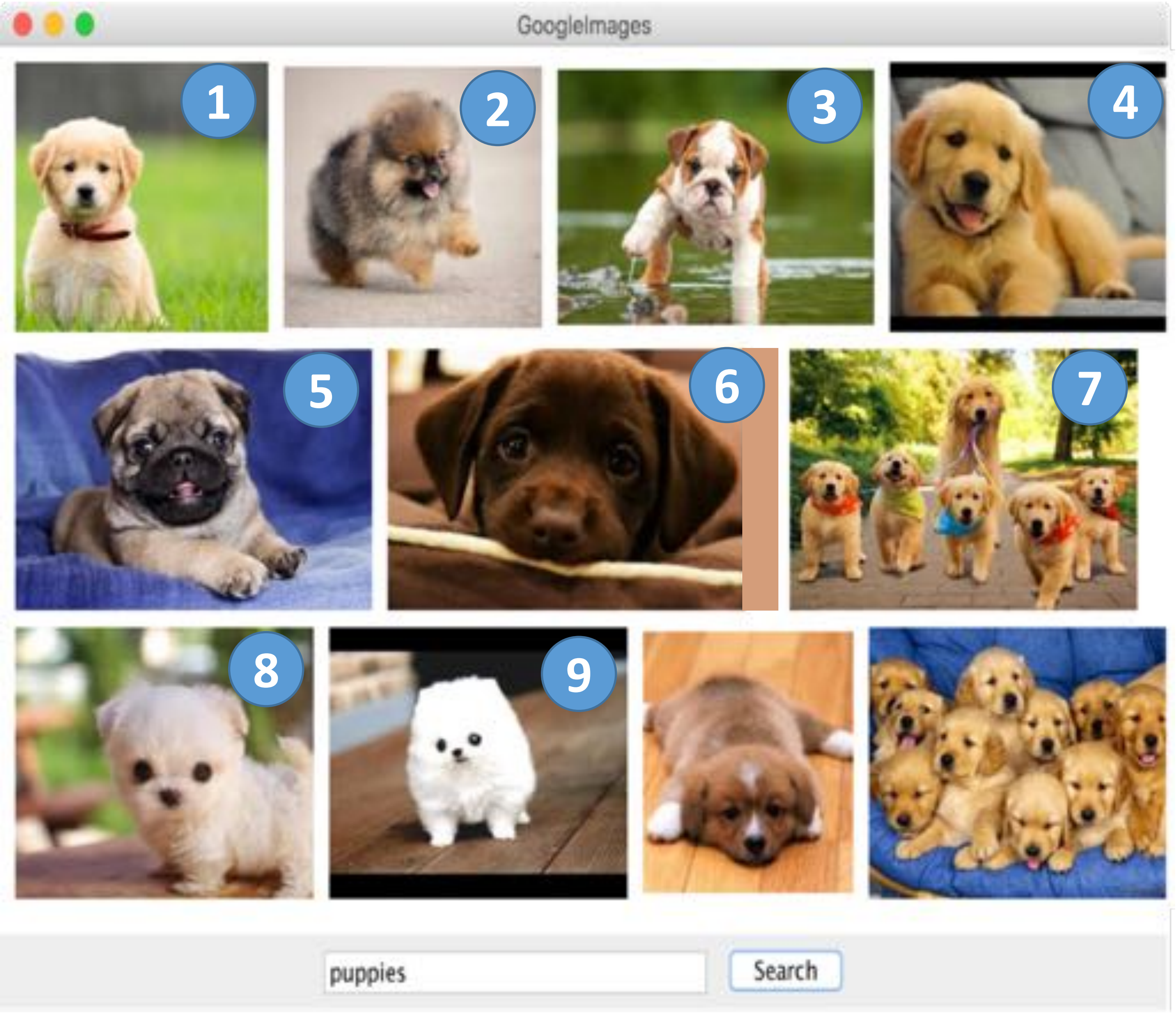
deleteDuplicates()



```
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);
    }
}
```

Strategy #3

Google Images



`public void displayQuery(String query)`

Use a helper method:

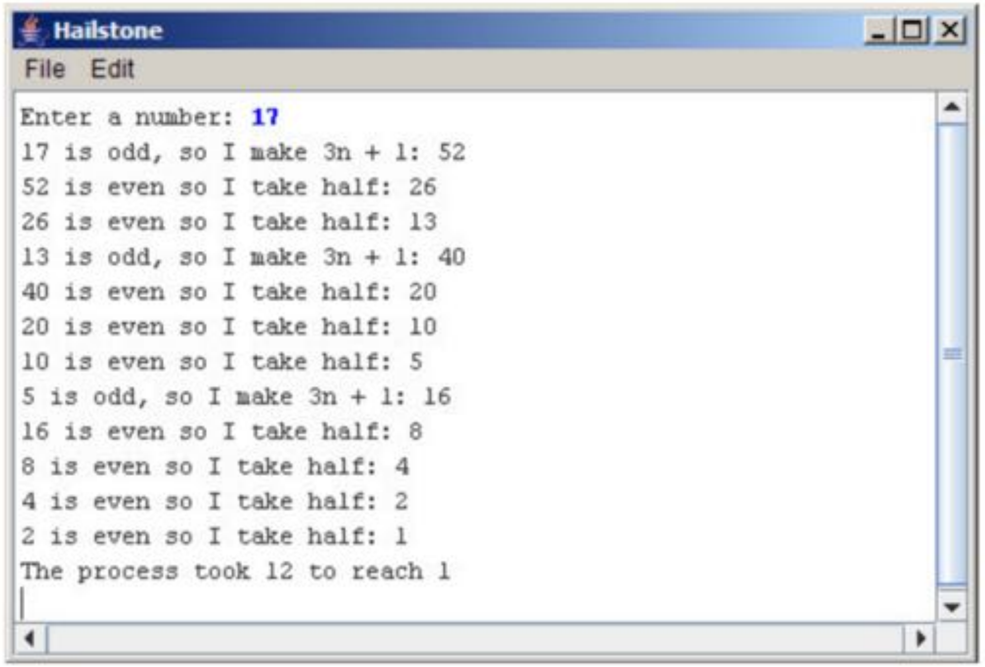
```
ArrayList<GImage> results =
        getSearchResults(query);
```

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

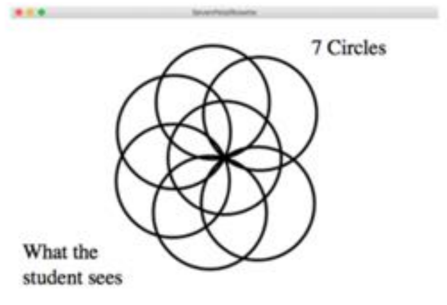
There is a spacing of **`GAP`** pixels between each picture. You can optionally include the `GAP` between the pictures and the border of the window.

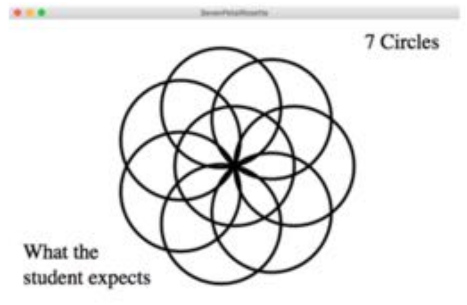
No image should go off the screen. You should not display all 100 returned images – only display the ones that fit into the three rows.

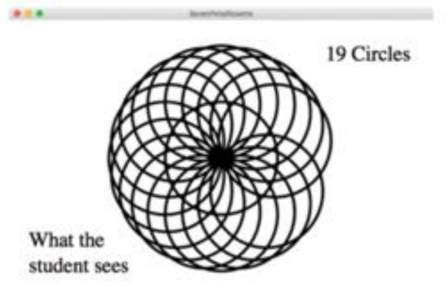
You have come a long way

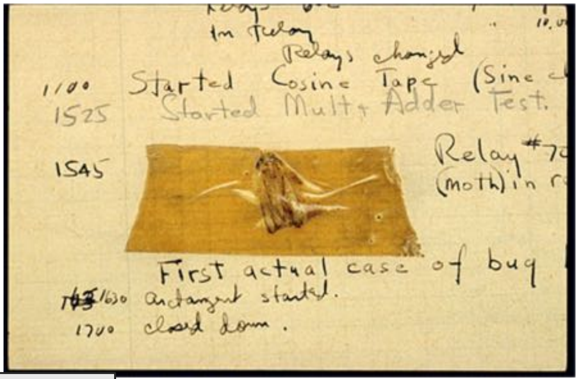


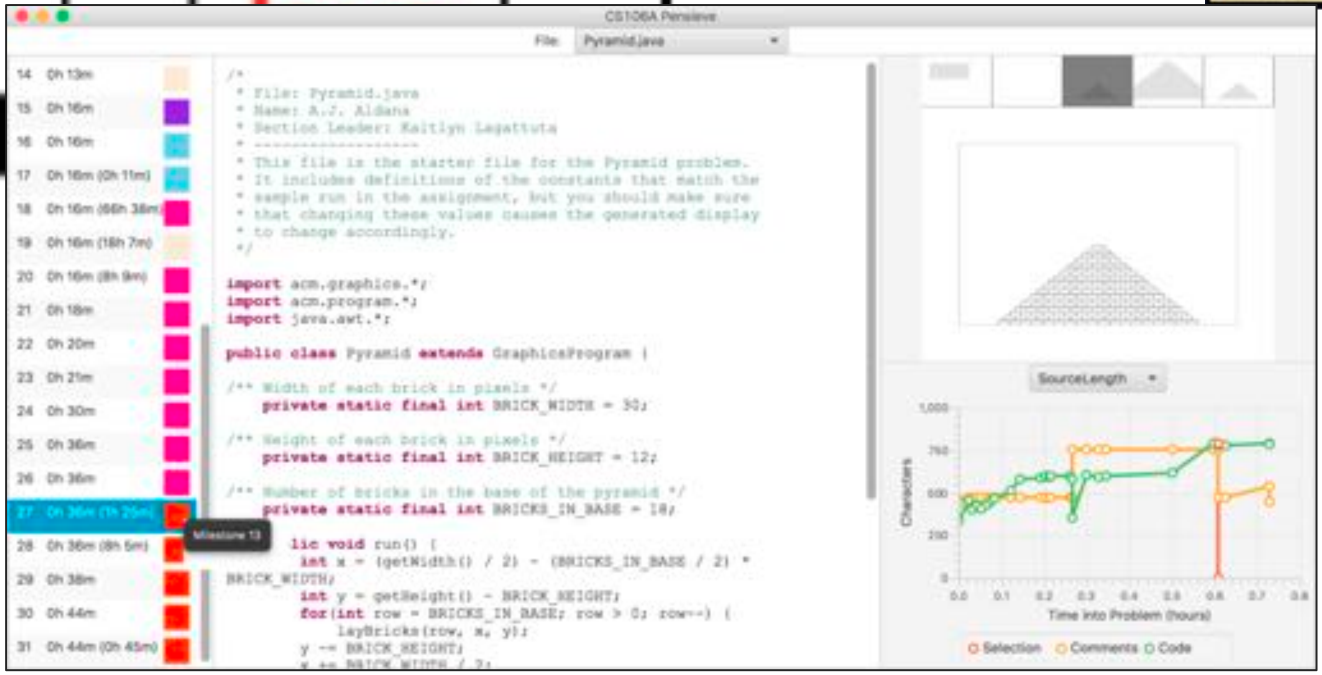
```
Enter a number: 17
17 is odd, so I make 3n + 1: 52
52 is even so I take half: 26
26 is even so I take half: 13
13 is odd, so I make 3n + 1: 40
40 is even so I take half: 20
20 is even so I take half: 10
10 is even so I take half: 5
5 is odd, so I make 3n + 1: 16
16 is even so I take half: 8
8 is 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
```





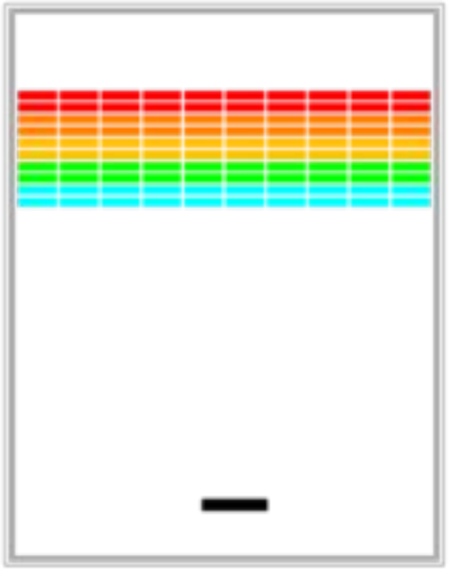
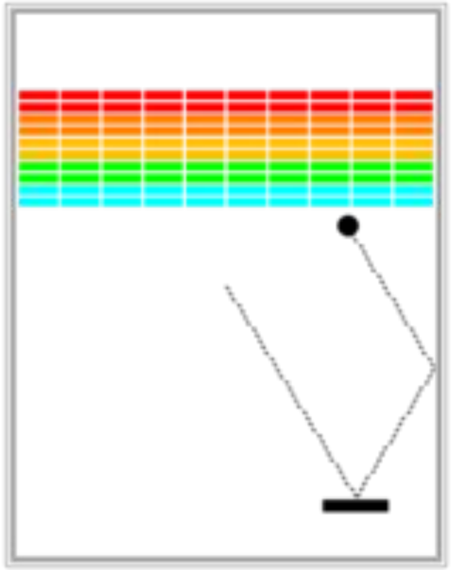
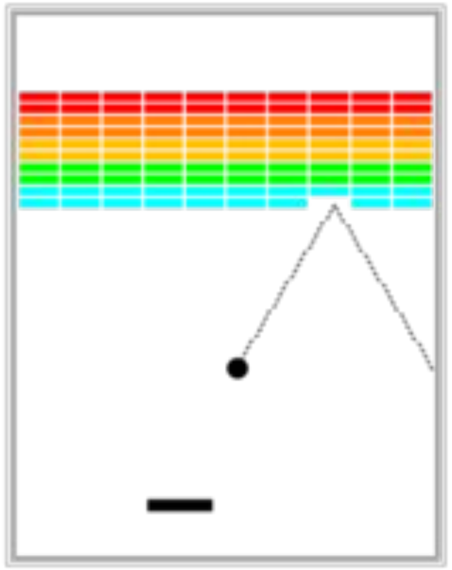
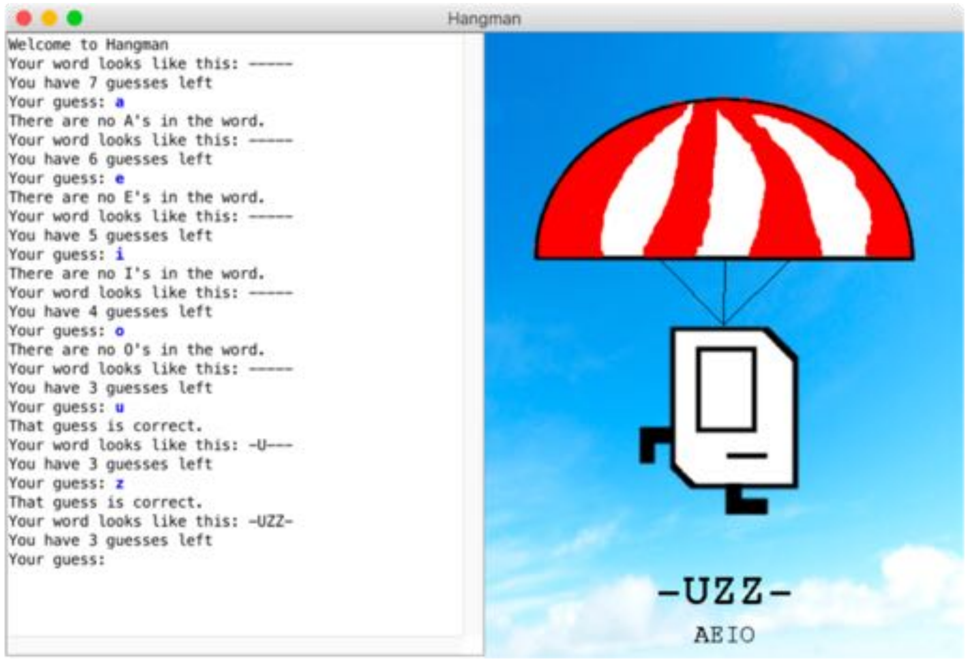



```
1100  Started Cosine Tape (Sine
1525  Started Mult+ Adder Test.
1545                         Relay #70
                             (moth) in
      First actual case of bug
1630  Antangent started.
1700  closed down.
```

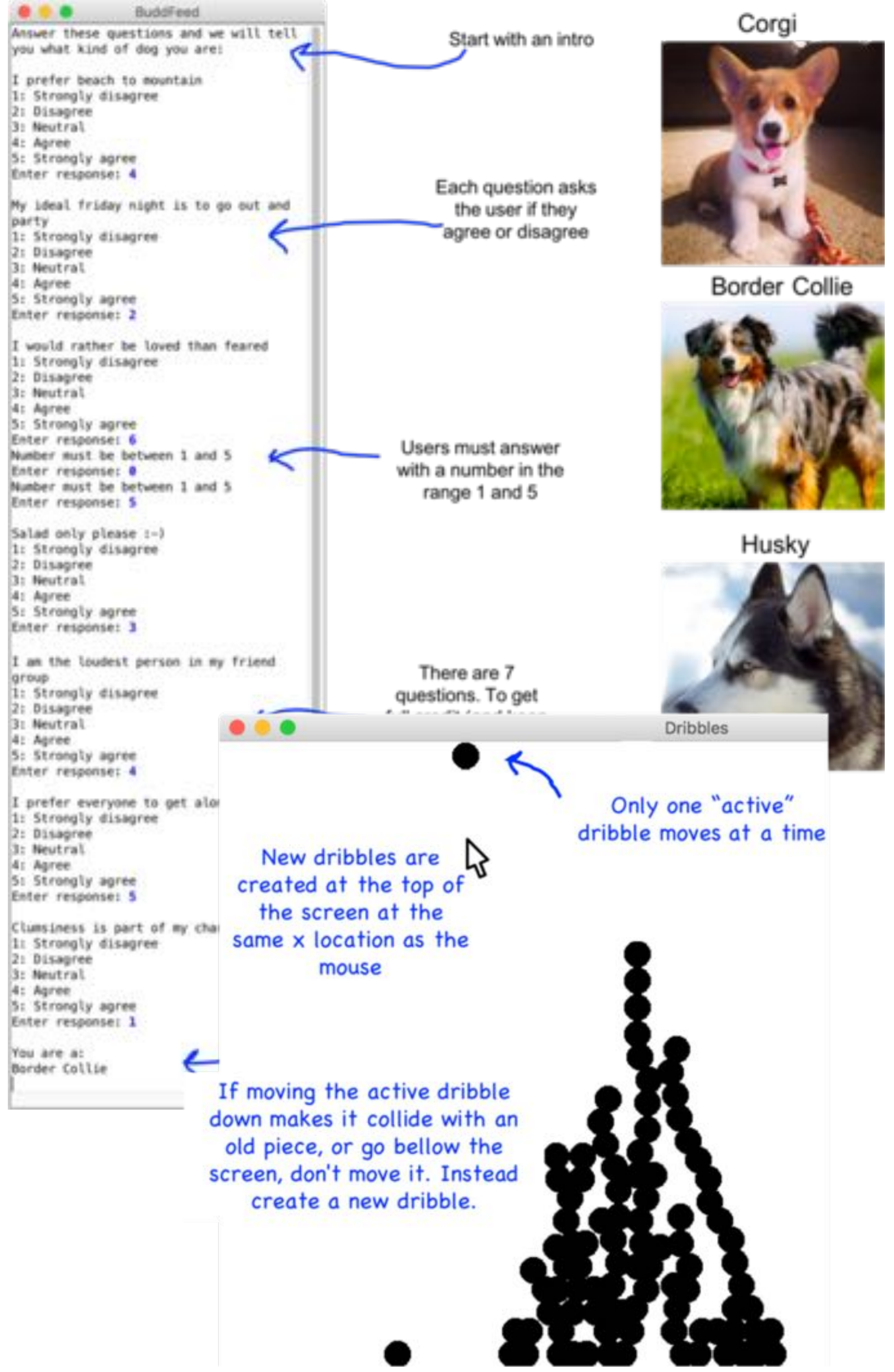


```
14  0h 13m
15  0h 16m
16  0h 16m
17  0h 16m (0h 11m)
18  0h 16m (66h 38m)
19  0h 16m (18h 7m)
20  0h 16m (8h 9m)
21  0h 18m
22  0h 20m
23  0h 21m
24  0h 30m
25  0h 36m
26  0h 36m
27  0h 36m (1h 20m)
28  0h 36m (8h 5m)
29  0h 38m
30  0h 44m
31  0h 44m (0h 45m)
```

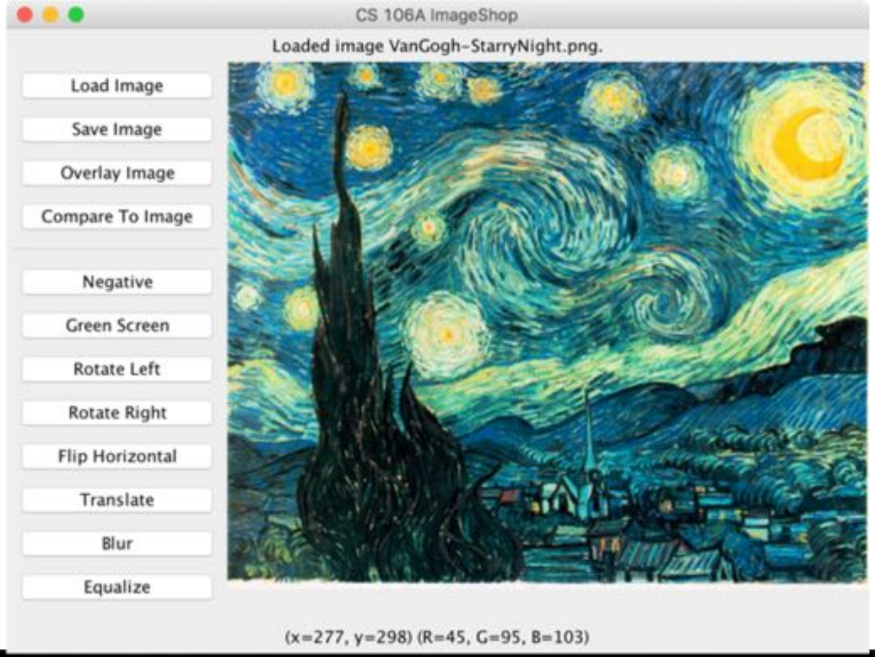
```
/*
 * File: Pyramid.java
 * Name: A.J. Aldana
 * Section Leader: Kaitlyn Lagattuta
 * ------------------
 * This file is the starter file for the Pyramid problem.
 * It includes definitions of the constants that match the
 * sample run in the assignment, but you should make sure
 * that changing these values causes the generated display
 * to change accordingly.
 */

import acm.graphics.*;
import acm.program.*;
import java.awt.*;

public class Pyramid extends GraphicsProgram {

/** Width of each brick in pixels */
    private static final int BRICK_WIDTH = 30;

/** Height of each brick in pixels */
    private static final int BRICK_HEIGHT = 12;

/** Number of bricks in the base of the pyramid */
    private static final int BRICKS_IN_BASE = 18;

      lic void run() {
       int x = (getWidth() / 2) - (BRICKS_IN_BASE / 2) *
BRICK_WIDTH;
       int y = getHeight() - BRICK_HEIGHT;
       for(int row = BRICKS_IN_BASE; row > 0; row--) {
            layBricks(row, x, y);
       y -= BRICK_HEIGHT;
       x += BRICK_WIDTH / 2;
```

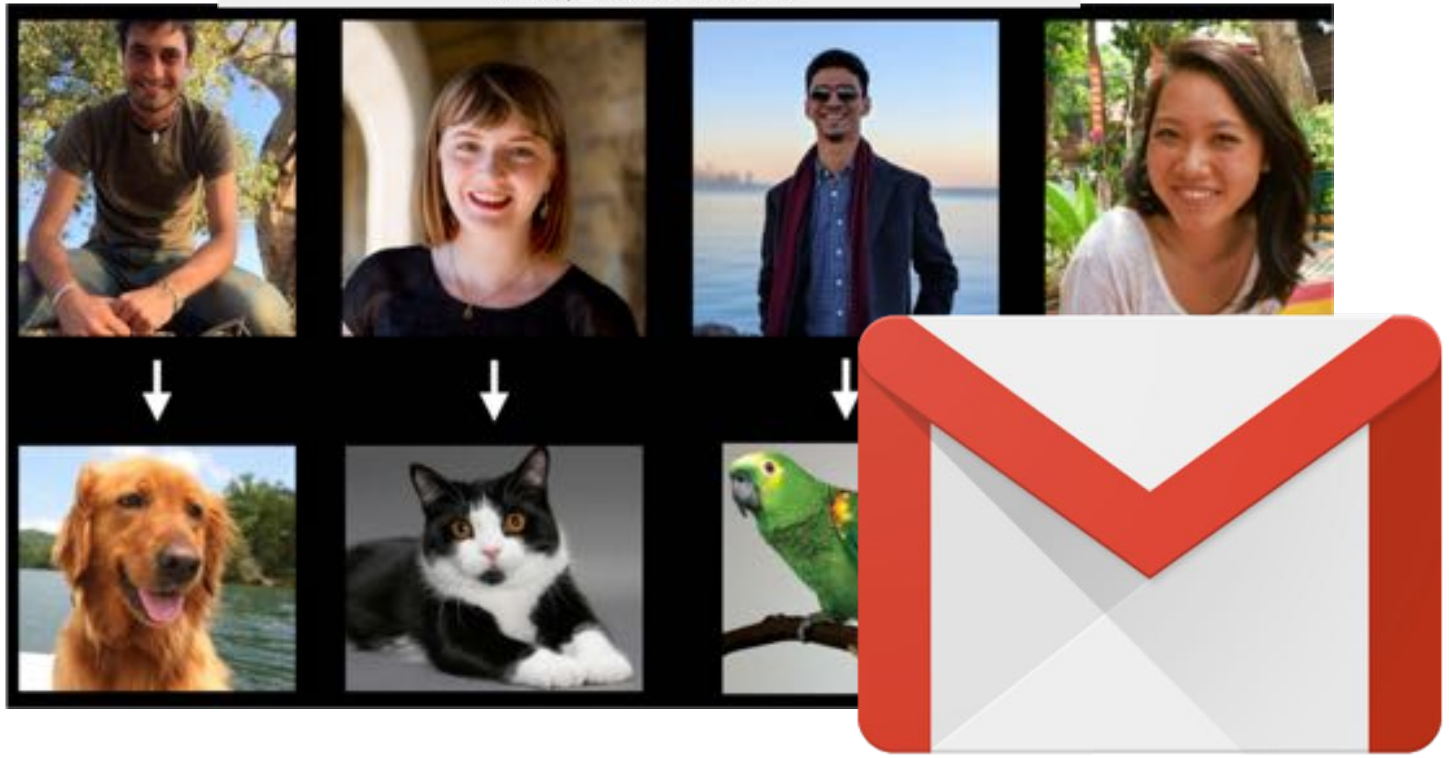





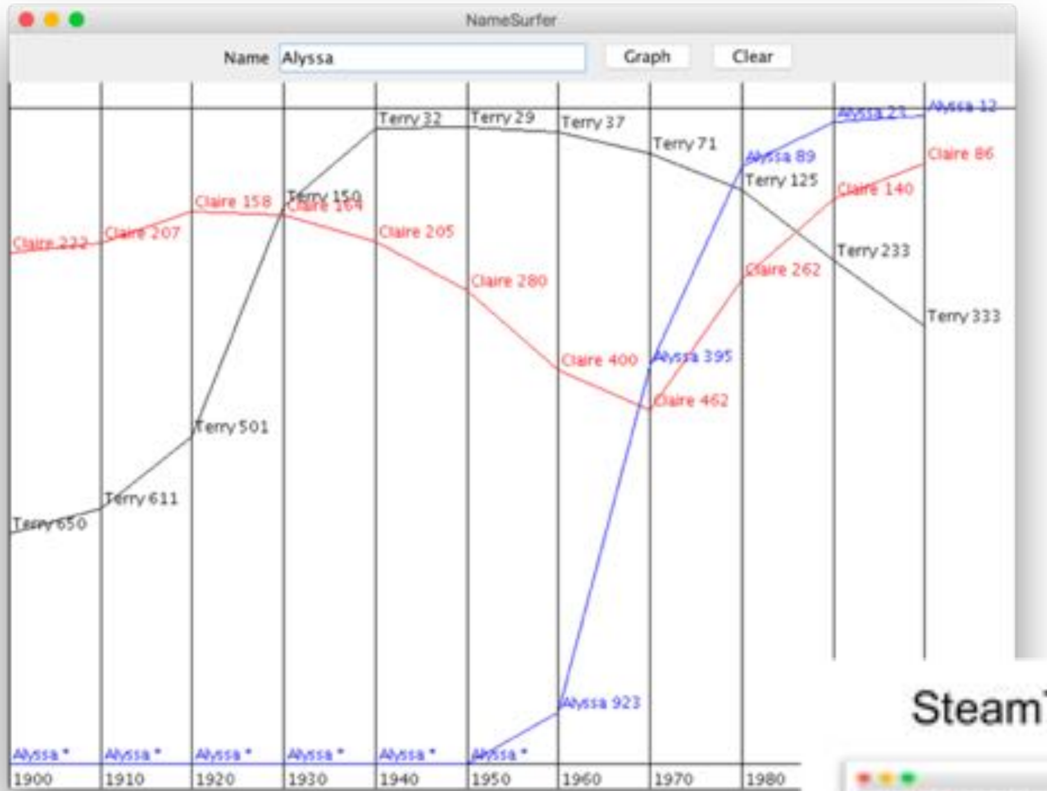
```
Welcome to Hangman
Your word looks like this: -----
You have 7 guesses left
Your guess: a
There are no A's in the word.
Your word looks like this: -----
You have 6 guesses left
Your guess: e
There are no E's in the word.
Your word looks like this: -----
You have 5 guesses left
Your guess: i
There are no I's in the word.
Your word looks like this: -----
You have 4 guesses left
Your guess: o
There are no O'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: z
That guess is correct.
Your word looks like this: -UZZ-
You have 3 guesses left
Your guess:
```

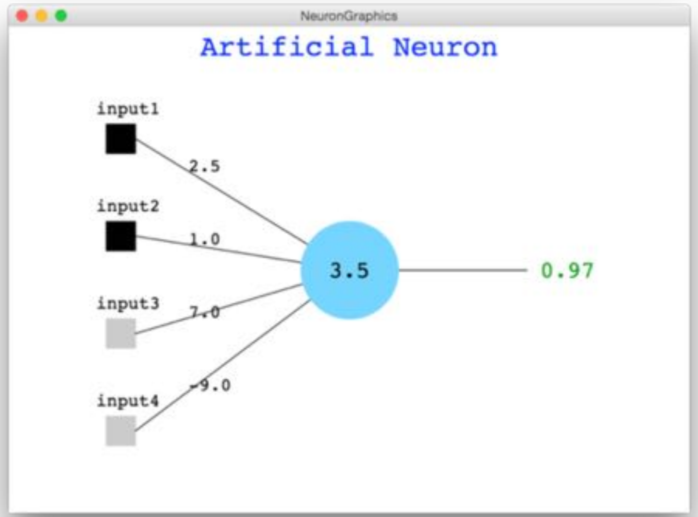

BuddFeed
Start with an intro
Each question asks the user if they agree or disagree
Users must answer with a number in the range 1 and 5
There are 7 questions. To get
Corgi
Border Collie
Husky
Dribbles
Only one "active" dribble moves at a time
New dribbles are created at the top of the screen at the same x location as the mouse
If moving the active dribble down makes it collide with an old piece, or go bellow the screen, don't move it. Instead create a new dribble.

```
Answer these questions and we will tell
you what kind of dog you are:

I prefer beach to mountain
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 4

My ideal friday night is to go out and
party
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 2

I would rather be loved than feared
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 6
Number must be between 1 and 5
Enter response: 0
Number must be between 1 and 5
Enter response: 5

Salad only please :-)
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 3

I am the loudest person in my friend
group
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 4

I prefer everyone to get alo
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 5

Clumsiness is part of my cha
1: Strongly disagree
2: Disagree
3: Neutral
4: Agree
5: Strongly agree
Enter response: 1

You are a:
Border Collie
```

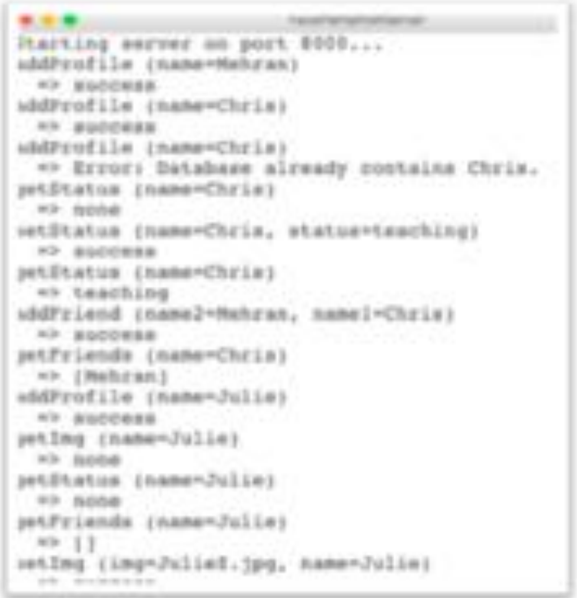








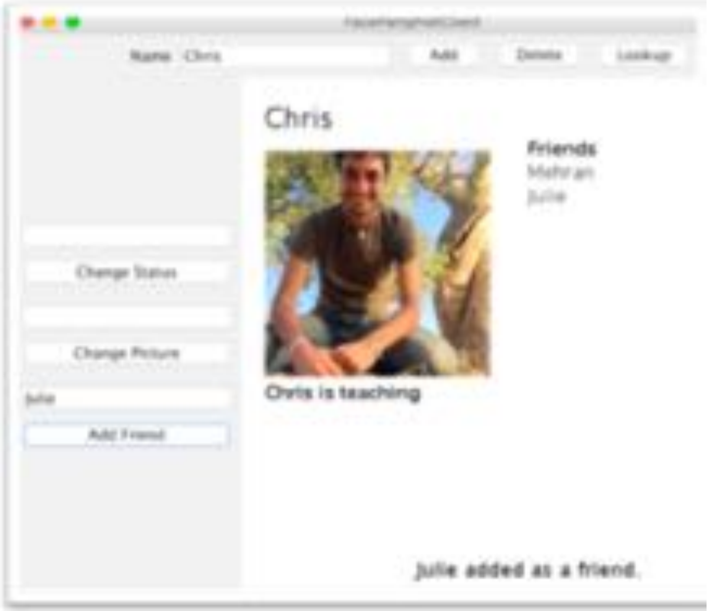
SteamTunnelServer



```
Starting server on port 8000...
addProfile (name=Mehran)
  => success
addProfile (name=Chris)
  => success
addProfile (name=Chris)
  => Error: Database already contains Chris.
getStatus (name=Chris)
  => none
setStatus (name=Chris, status=teaching)
  => success
getStatus (name=Chris)
  => teaching
addFriend (name2=Mehran, name1=Chris)
  => success
getFriends (name=Chris)
  => [Mehran]
addProfile (name=Julie)
  => success
getImg (name=Julie)
  => none
getStatus (name=Julie)
  => none
getFriends (name=Julie)
  => []
setImg (img=Julie8.jpg, name=Julie)
```

Communicate via the internet

SteamTunnelClient



Chris is teaching

Julie added as a friend.

By the numbers

7 hard assignments

14,000 person hours programming

350 pieces of fruit

1 class ☺

You have my respect.

Why Study CS?

Joy of Building

Interdisciplinary

Closest Thing To Magic

Now is the Time

Technology Review
Sex Cells
David Byrne on Today's Gadgets
Rethinking the Skills Gap
MEET TECH'S RISING STARS
03
Electronics Research Laboratory
Drivers not required.
www.lew
Backrub by two Stanford PhD students -Larry Page & Sergey Brin

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;
        }
    }
}
```