Review 1

Chris Piech
CS106A, Stanford University

Plan for today

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

Plan for tomorrow

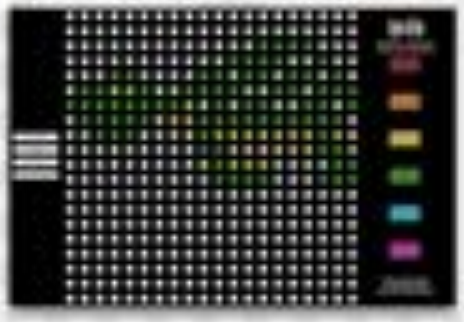
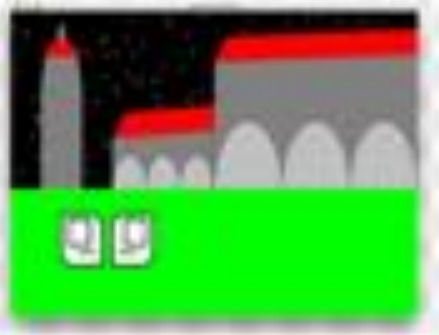
- Announcements/Exam logistics
- HashMaps
- Classes
- Interactors







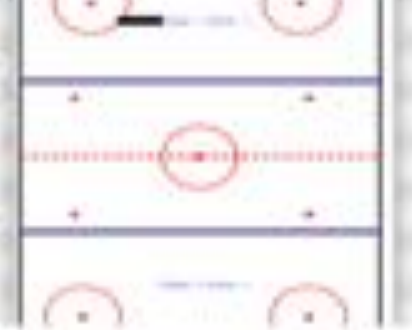






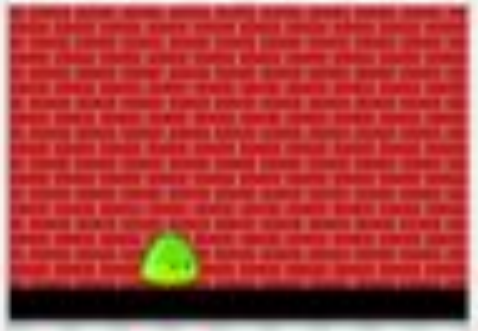


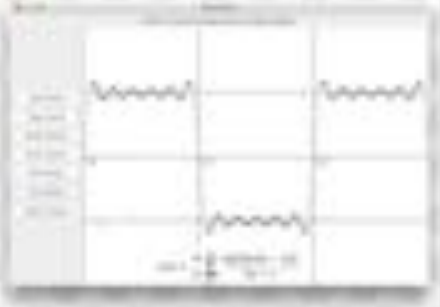
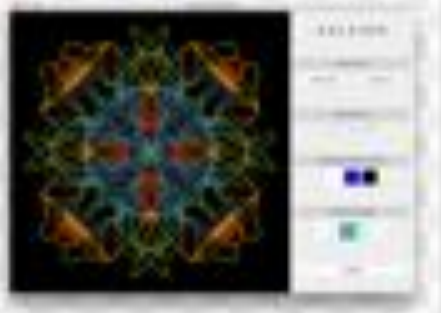



Merry Christmas, Christ
From Rick

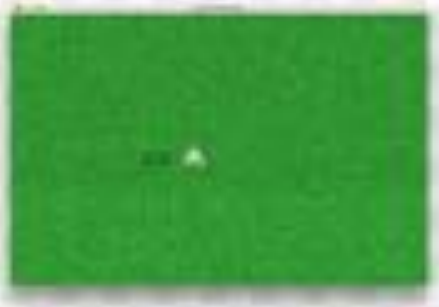


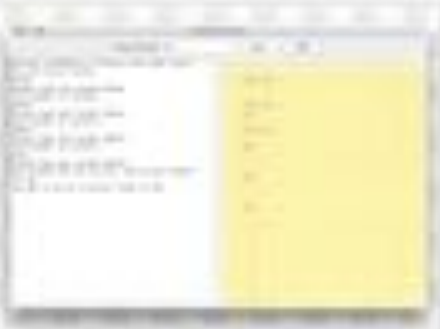


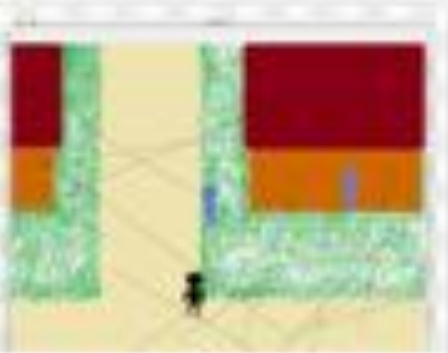












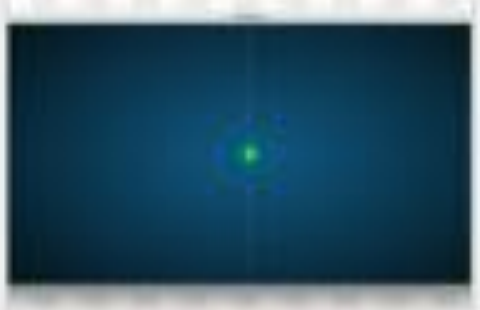
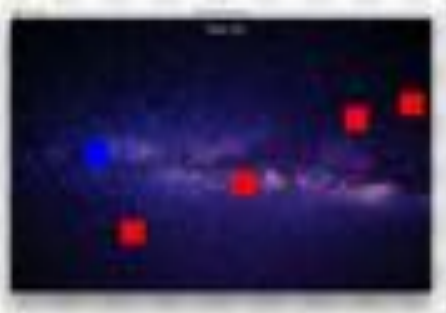
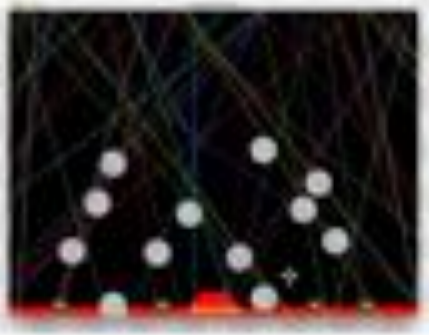
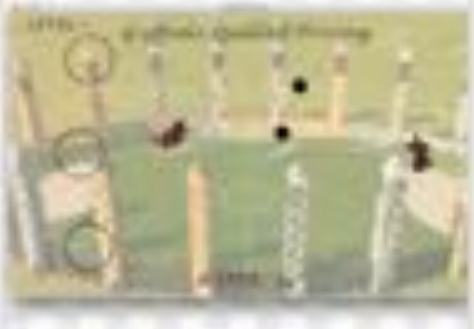












THANK YOU MARION


What should Chris bring to lecture today?

	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

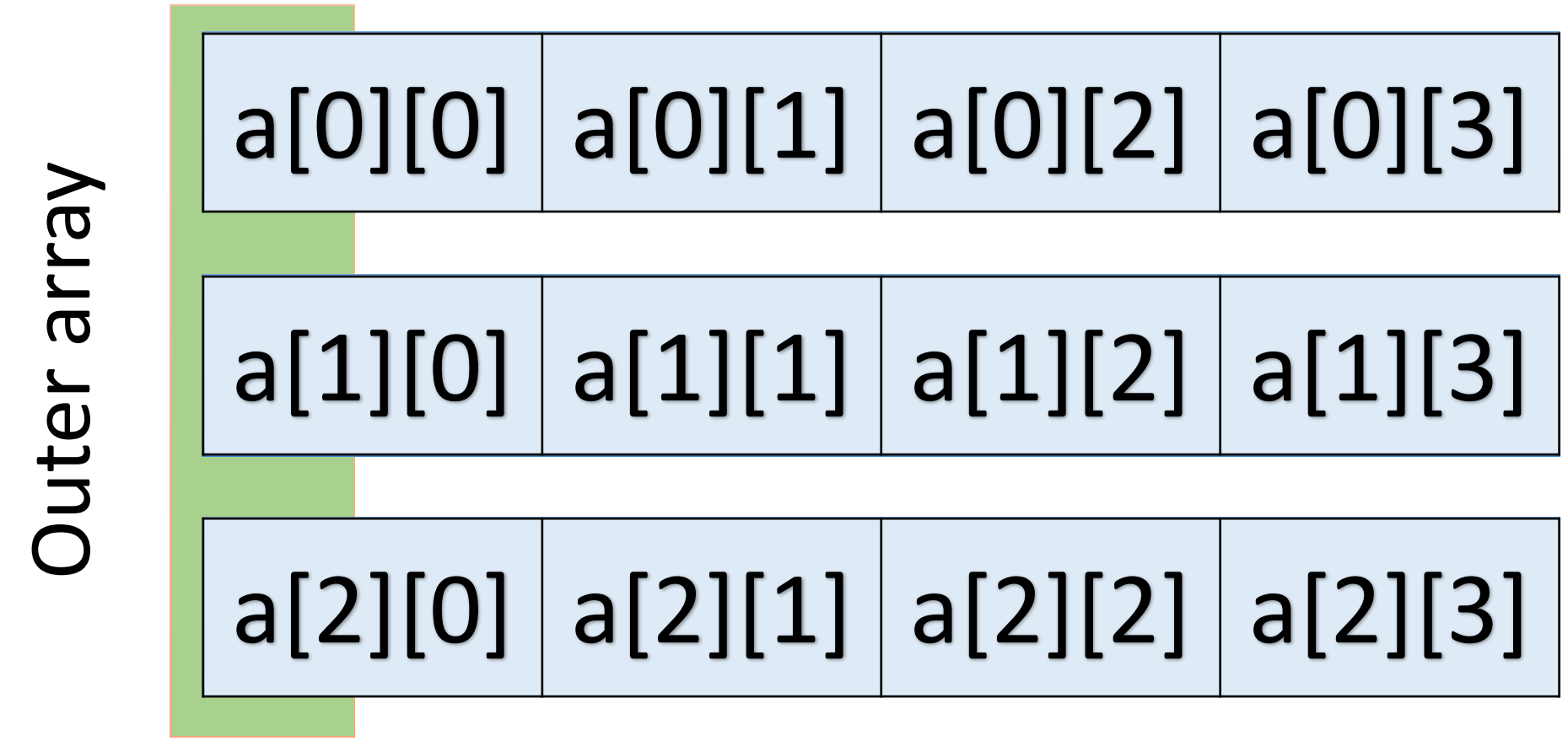
	String	Array	2D Array	ArrayList	HashMap
Model	Sequence of letters or symbols	Fixed length elements in a list	Grid / Matrix of elements	Growable list of elements	Key/Value mapping
Type of element	chars	Objects & Primitives	Objects & Primitives	Objects	Object/Object

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, FPDatabase

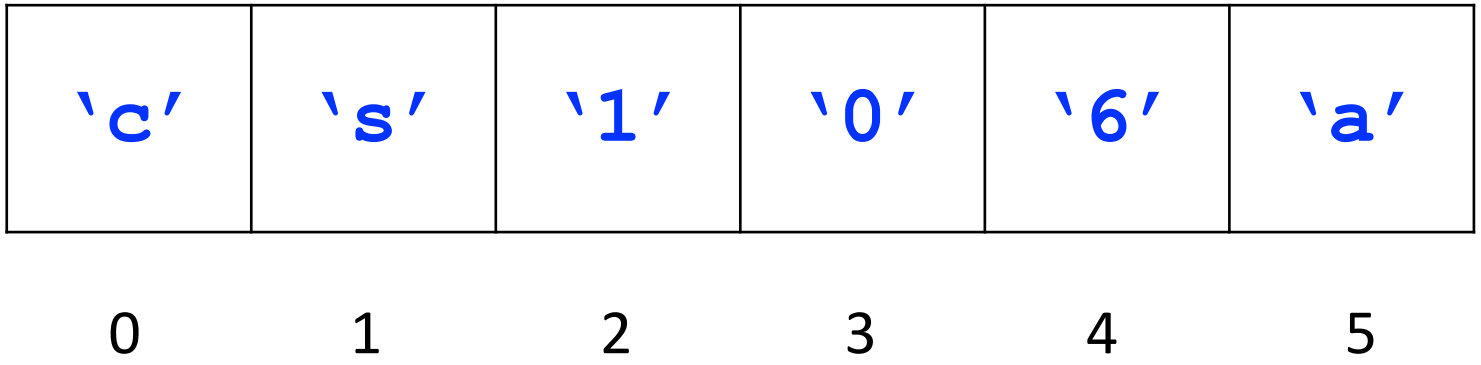
2D Arrays = Array of Arrays

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



Strings under the hood are 1D Array of chars

```
String str = “cs106a”;
```



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 *heap 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);
}
```

Program Traces

- **Approaching program traces**
  - Local variables are *separate* across methods
  - Parameters are just assigned names by the order in which they're passed
  - Write values above variable names as you go through the program (or draw stack frame boxes)
  - Pass-by-reference vs. pass-by-value

```
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

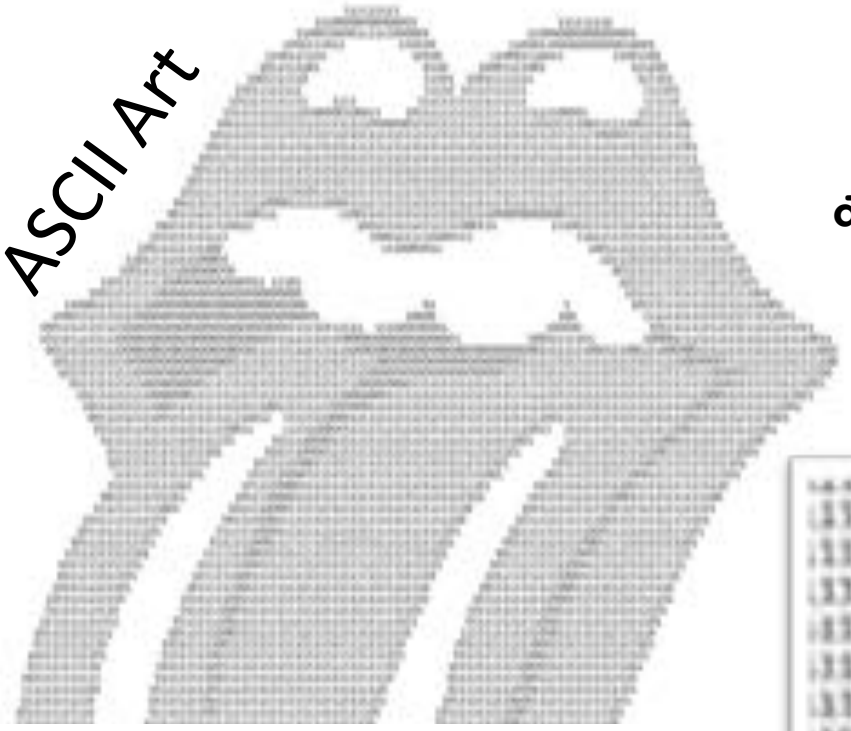
```
// return the maximum value in the matrix
private double getMax(double[][] matrix) {
  double maxValue = matrix[0][0];
  for(int r = 0; r < matrix.length; r++) {
    for(int c = 0; c < matrix[0].length; c++) {
      if(matrix[r][c] > maxValue) {
        maxValue = matrix[r][c];
      }
    }
  }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

Make ASCII Art

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

Original

ASCII Art



Helper method

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

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()

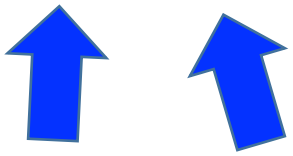
```
private void deleteDuplicates(ArrayList<String> list)
```

- 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"}

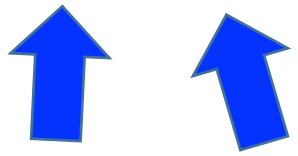


`curr` `next`

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"



`curr` `next`

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

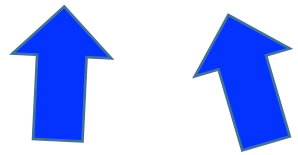
deleteDuplicates()

	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"}

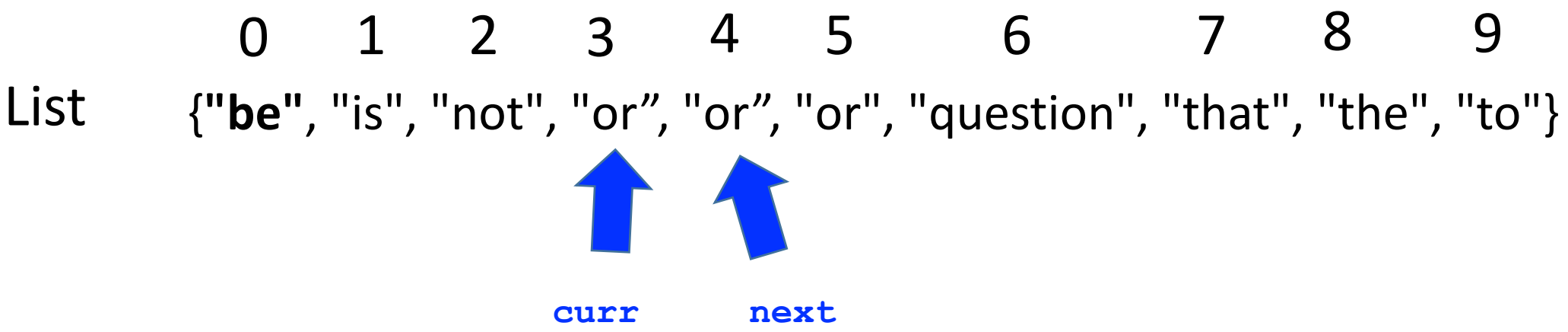


`curr` `next`

Current Index (`i`): `1`

Sometime later…

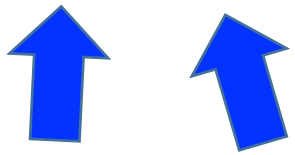
deleteDuplicates()



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"}



`curr` `next`

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

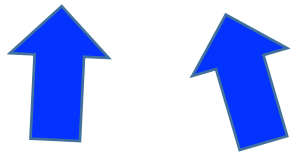
deleteDuplicates()

	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"}



`curr` `next`

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

deleteDuplicates()

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

```
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

```
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);
        }
    }
}
```

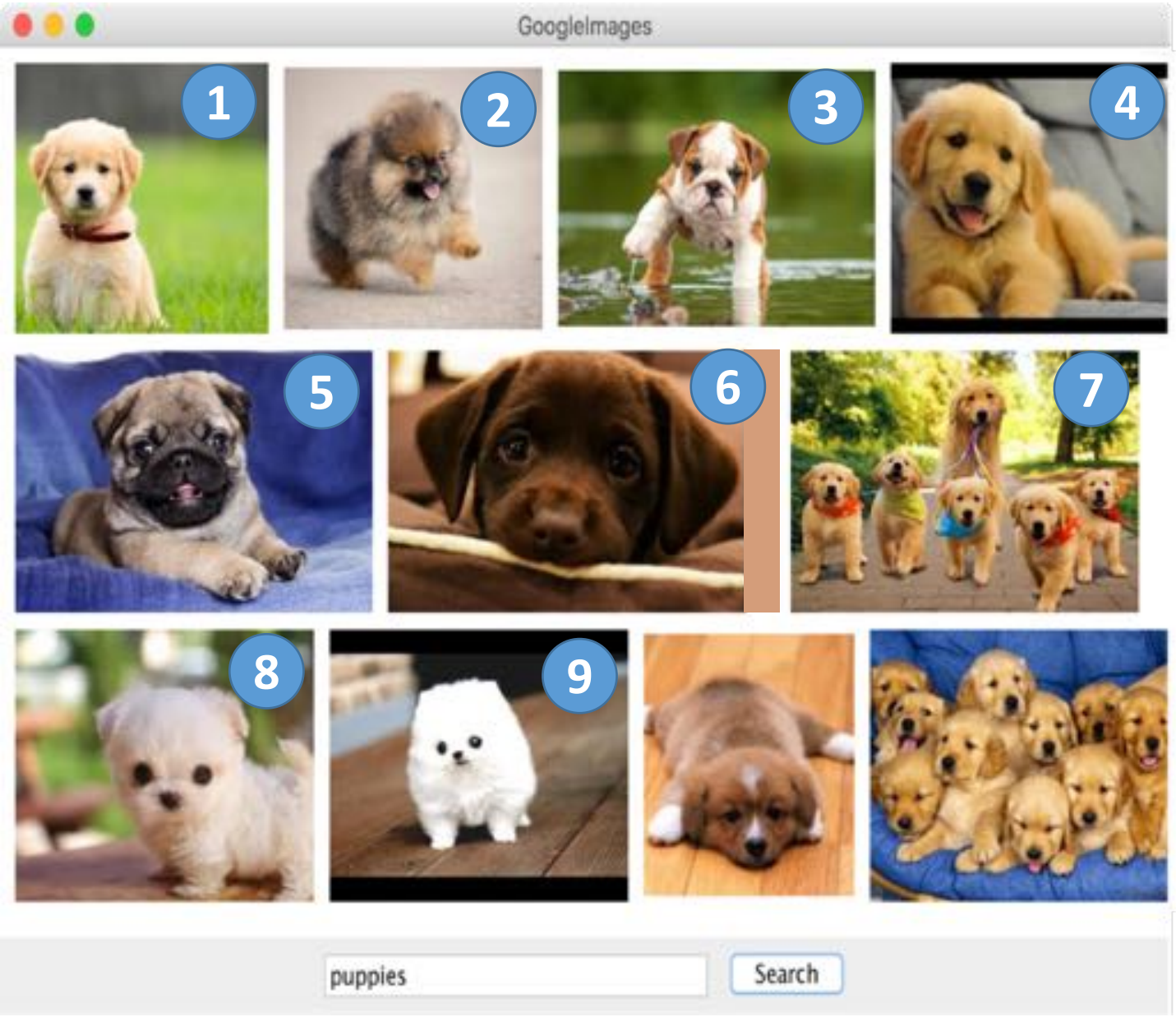
Strategy #2

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:

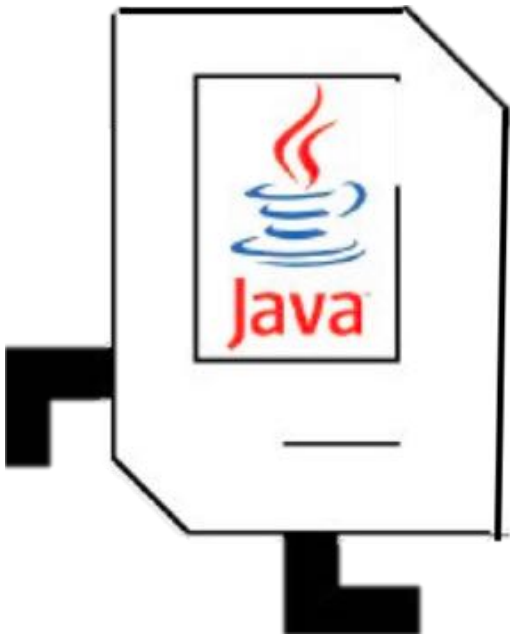
```
ArrayList<GImage> results =
        getSearchResults(query);
```

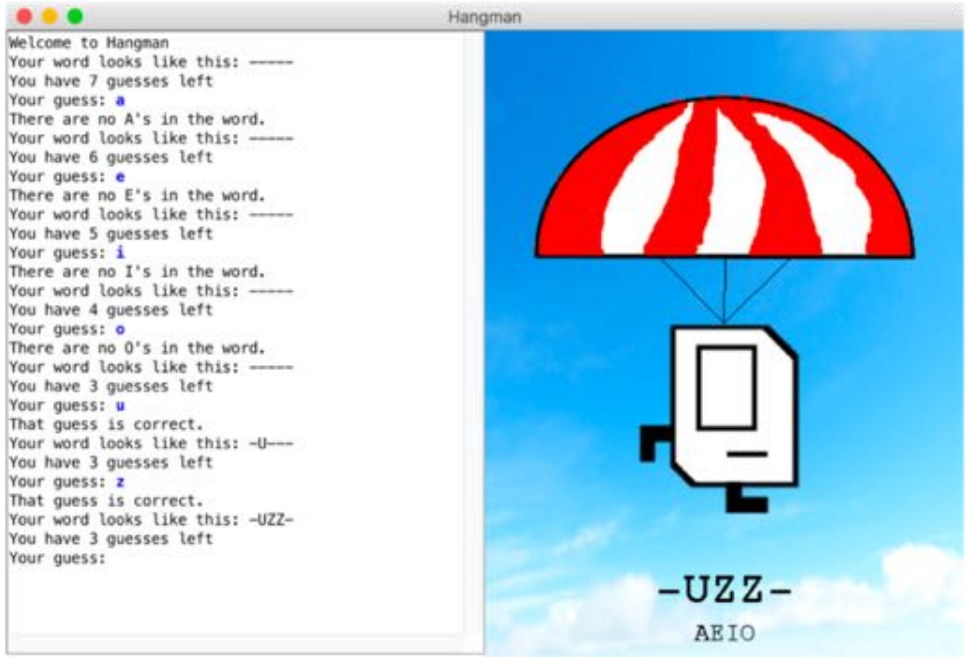
display your images in three rows of fixed height **`ROW_HEIGHT`** . You can scale images, but should maintain the ratio of their width to height. You can change the size of a GImage using it's **`setSize(width, height)`** method

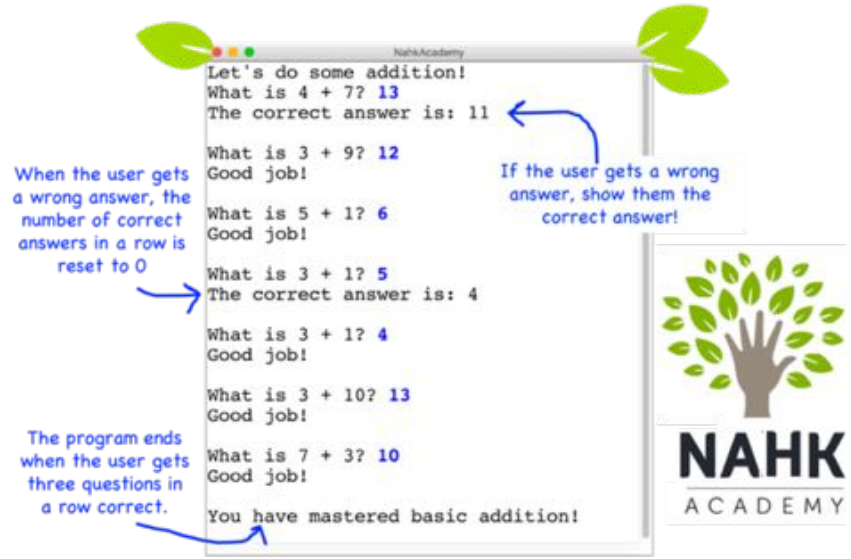
There is a spacing of **`GAP`** pixels between each picture. You can optionally include the `GAP` between the pictures and the border of the window.

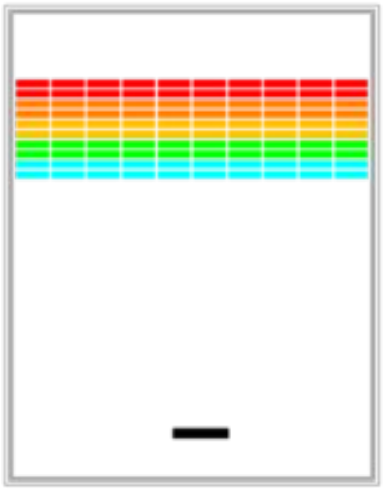
No image should go off the screen. You should not display all 100 returned images – only display the ones that fit into the three rows.

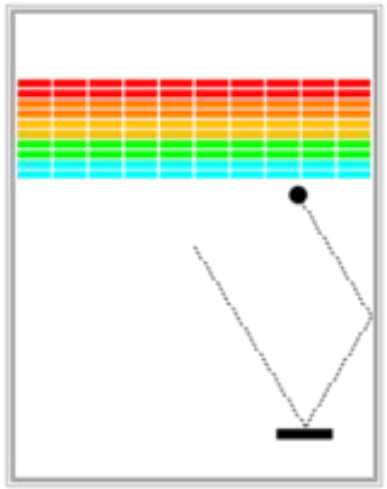
You have come a long way

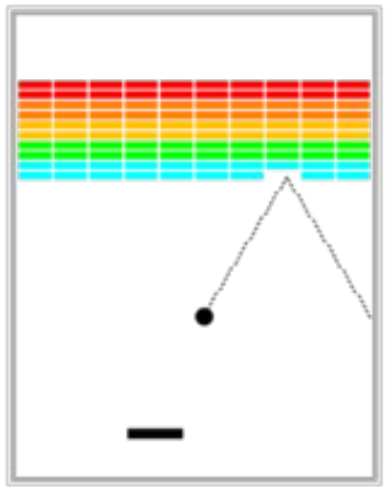

Java


Hangman
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:
-UZZ-
AEIO

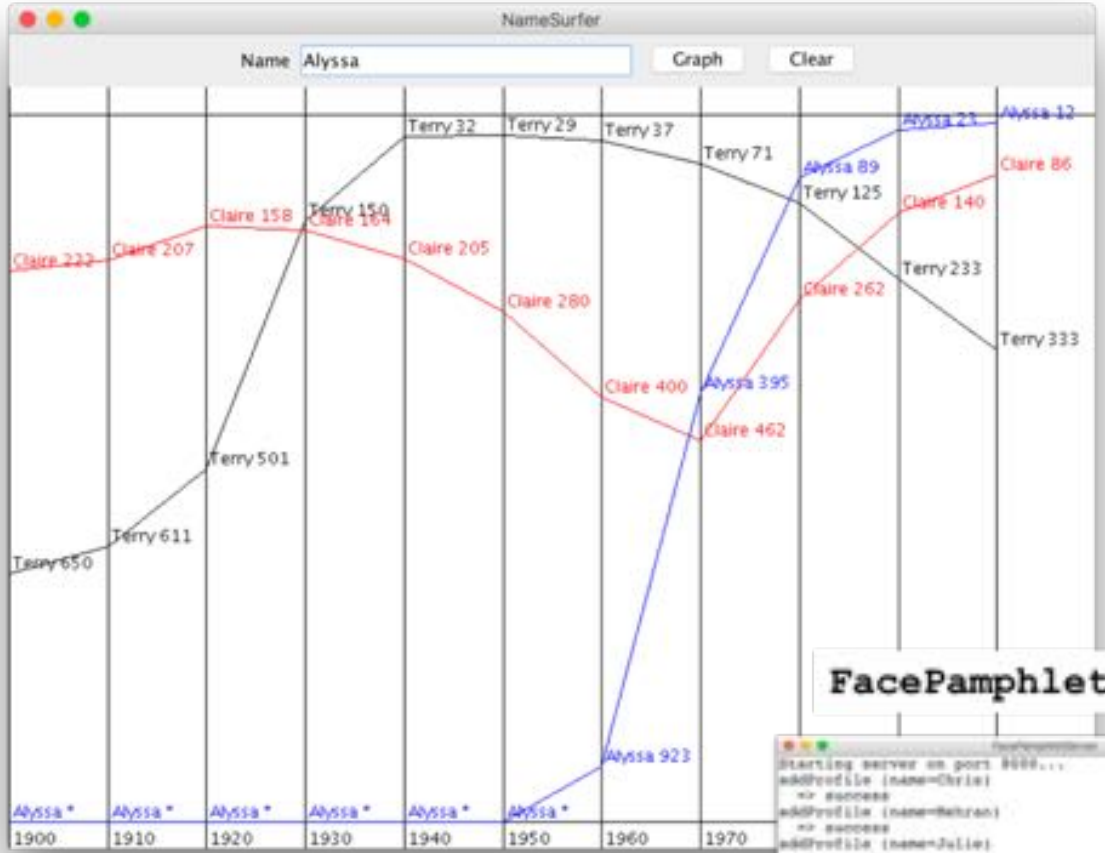

When the user gets a wrong answer, the number of correct answers in a row is reset to 0
If the user gets a wrong answer, show them the correct answer!
The program ends when the user gets three questions in a row correct.
NahkAcademy
Let's do some addition!
What is 4 + 7? 13
The correct answer is: 11
What is 3 + 9? 12
Good job!
What is 5 + 1? 6
Good job!
What is 3 + 1? 5
The correct answer is: 4
What is 3 + 1? 4
Good job!
What is 3 + 10? 13
Good job!
What is 7 + 3? 10
Good job!
You have mastered basic addition!
NAHK
ACADEMY

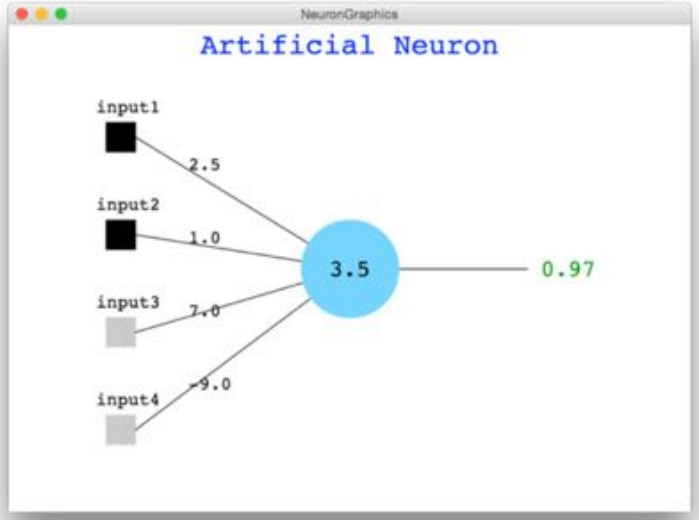







CS 106A ImageShop
Loaded image VanGogh-StarryNight.png.
Load Image
Save Image
Overlay Image
Compare To Image
Negative
Green Screen
Rotate Left
Rotate Right
Flip Horizontal
Translate
Blur
Equalize
(x=277, y=298) (R=45, G=95, B=103)





FacePamphletServer

Communicate via the internet

FacePamphletClient

You have my respect.

Why Study CS?

Joy of Building

Interdisciplinary

Closest Thing To Magic

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;
        }
    }
}
```