

The background features a vibrant blue gradient. At the bottom, there is a white silhouette of a city skyline with various building shapes. Scattered throughout the sky are several light blue, stylized clouds of different sizes and shapes.

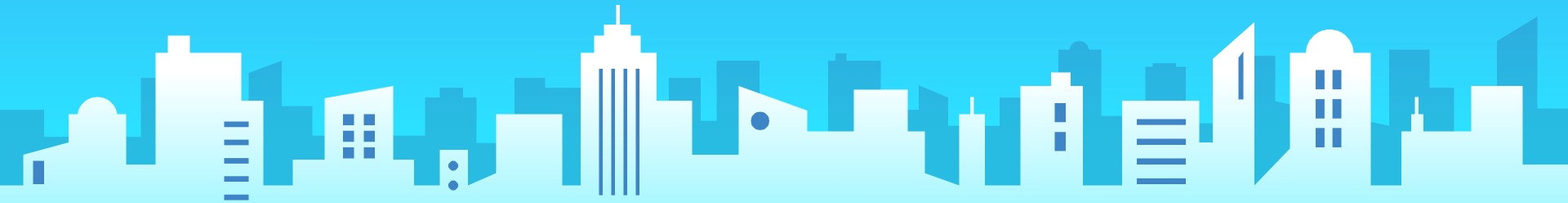
# Assignment 6

# YEAH Hours

Ben Barnett and Avery Wang

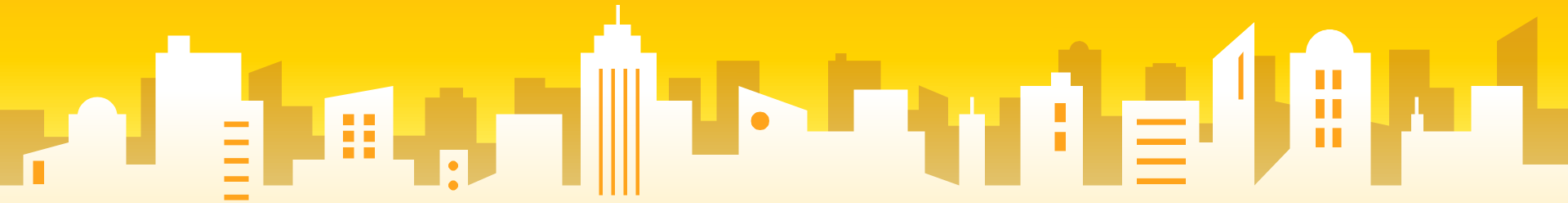
## Overview

1. Review relevant material.
2. Discuss each milestone.
3. Q&A



# Classes

Define your very own variable type!



# What custom variables have you already been using?

Hint: anything not a primitive came from a class!

HashMap

ArrayList

GRect

MouseEvent

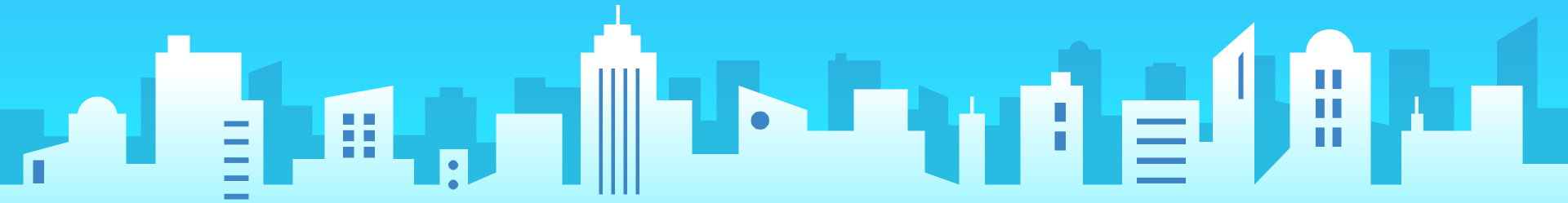
Integer

(the wrapper class)

GImage

String

SuperKarel







# Classes = Blueprints

A class is a blueprint for a custom variable type.

# The blueprint must define three things.

What information does this variable store?

What can you do with this variable type?

How do you create this variable type?

**Instance variables**

**Methods**

**Constructor**

## Example: `ArrayList<Integer>`

What information does this variable store?

What can you do with this variable type?

How do you create this variable type?

Each element, size

`add`, `indexOf`, `contains`, ...

`new ArrayList<Integer>()`

# ArrayList Instance Variables

What information does this variable store?

```
private int size;  
private int elements[];
```

# ArrayList Methods

What can you do with this variable do?

```
public void add(int element) {...}  
public int get(int index) {...}  
public boolean contains(int element) {...}
```

# ArrayList Methods

What can you do with this variable do?

```
public boolean contains(int element) {  
    // something cool  
}
```

# ArrayList Constructor

How do you create this variable type and initialize instance variables?

```
public ArrayList<Integer>() {...}  
public ArrayList<Integer>(int capacity) {...}
```

Sidenote: the array stored in the ArrayList start with capacity 10 if you use the default constructor, and resizes if it needs more than 10 elements, but you start with a different capacity, the initial array will have the given capacity.

# ArrayList Constructor

How do you create this variable type and initialize instance variables?

```
public ArrayList<Integer>() {  
    size = 0; // initialize size  
    elements = new int[10]; // initialize  
elements  
}
```



# ArrayList Used in Our Program

We can now use this custom variable type!

```
public void run() {  
    ArrayList<Integer> scores = new ArrayList<Integer>() // constructor  
    scores.add(100); // method that uses the instance we constructed  
}
```



# Instances

An instance is one object you created using the blueprint.

# Where did we create a new instance of ArrayList?

```
public void run() {  
    ArrayList<Integer> scores = new ArrayList<Integer>() // constructor  
    scores.add(100); // method that uses the instance we constructed  
}
```

The background features a gradient from red at the top to yellow at the bottom. Stylized orange and yellow clouds are scattered in the upper portion. At the bottom, a white silhouette of a city skyline is visible, including various building shapes, a tower with vertical lines, and a building with a red circle on its side.

# Interactors!

Don't press the button!

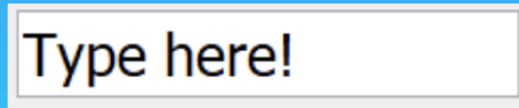
# Main Interactors

**JButtons**



**(click input)**

**JTextFields**



**(text input)**

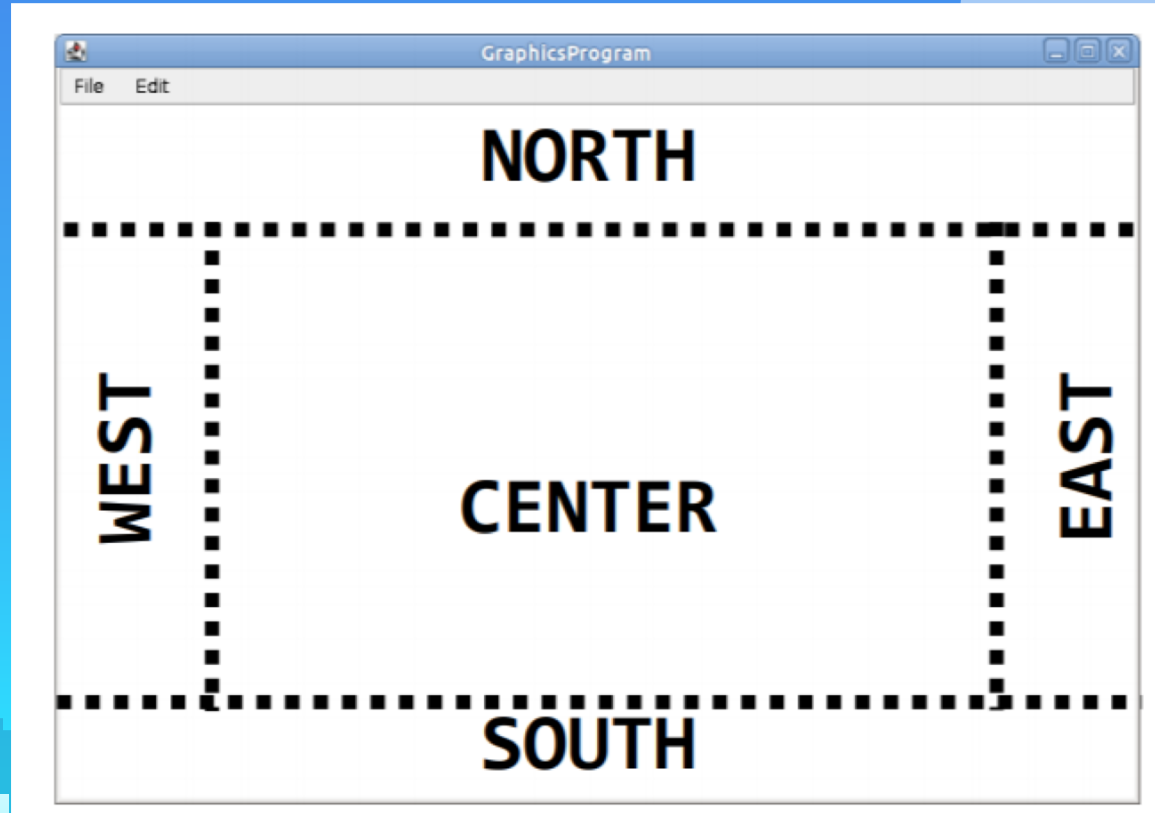
**JLabels**

I'm kinda useless. :(

**(not an interactor)**

# GraphicsProgram Regions

Canvas goes CENTER -  
usually not where you  
put interactors



# JButton Methods

Method	Description
<code>new JButton("text")</code>	Creates new button with given text string
<code>jb.setBackground()</code> <code>jb.setBackground(color);</code>	get or set background color on button
<code>jb.isEnabled()</code> <code>jb.setEnabled(boolean);</code>	get or set whether button is clickable
<code>jb.getFont()</code> <code>jb.setFont(font);</code>	get or set text font used for button text
<code>jb.setForeground()</code> <code>jb.setForeground(color);</code>	get or set text color on button
<code>jb.getIcon()</code> <code>jb.setIcon(icon);</code>	get or set icon image showing on button
<code>jb.getText()</code> <code>jb.setText("text");</code>	set or return text showing on the button

# JTextField Methods

Method	Description
<code>new JTextField("text")</code> <code>new JTextField(<i>columns</i>)</code>	Create new text field of given size
<code><i>jtf</i>.addActionListener(this);</code>	causes action events to occur when the user presses Enter on the field
<code><i>jtf</i>.getActionCommand()</code> <code><i>jtf</i>.setActionCommand("cmd");</code>	set/return a string to identify the action events that will occur in this field
<code><i>jtf</i>.getText()</code> ← <code><i>jtf</i>.setText("text");</code>	set/return text in the field



# JLabel Methods

Method	Description
<code>new JLabel("text")</code>	Create new label with given text
<code>jl.getFont()</code> <code>jl.setFont(font);</code>	get/set text font used for label text
<code>jl.setForeground()</code> <code>jl.setForeground(color);</code>	get or set text color on label
<code>jl.getHorizontalAlignment()</code> <code>jl.setHorizontalAlignment(align);</code>	set or return horizontal alignment of text in the label; pass <code>JLabel.LEFT</code> , <code>JLabel.CENTER</code> , or <code>JLabel.RIGHT</code>
<code>jl.getText()</code> <code>jl.setText("text");</code>	set/return text in the label

# Action Event Methods

Method	Description
<code>e.getActionCommand()</code>	a text description of the event <i>(e.g. the text of the button clicked)</i>
<code>e.getSource()</code>	the interactor that generated the event

# How to use interactors

```
public void init() {  
    JButton button = new JButton("Graph");  
    add(button, NORTH);  
    addActionListeners(); // Listen for all button clicks  
}
```

# How to use interactors

```
public void actionPerformed(ActionEvent e) {  
    if (e.getActionCommand().equals("Graph") ) {  
        // do something  
    }  
}
```

# How to use interactors

```
public void init() {  
    JTextField searchBar= new JTextField(TEXT_FIELD_WIDTH);  
    searchBar.setActionCommand("SearchBar");  
    searchBar.addActionListener(this); // Listen for "ENTER" in text field  
    add(searchBar, SOUTH);  
}
```

# How to use interactors

```
public void actionPerformed(ActionEvent e) {  
    if (e.getActionCommand().equals("SearchBar")) {  
        String text = searchBar.getText(); // gets user input.  
    }  
}
```



# actionPerformed

Called when a button is pressed or textbox is ENTERED.

# How to use many interactors

```
public void actionPerformed(ActionEvent e) {  
    if (e.getActionCommand().equals("SearchBar")) {  
        String text = searchBar.getText(); // gets user input.  
    } else if (e.getActionCommand().equals("StopButton")) {  
        // stop search  
    }  
}
```





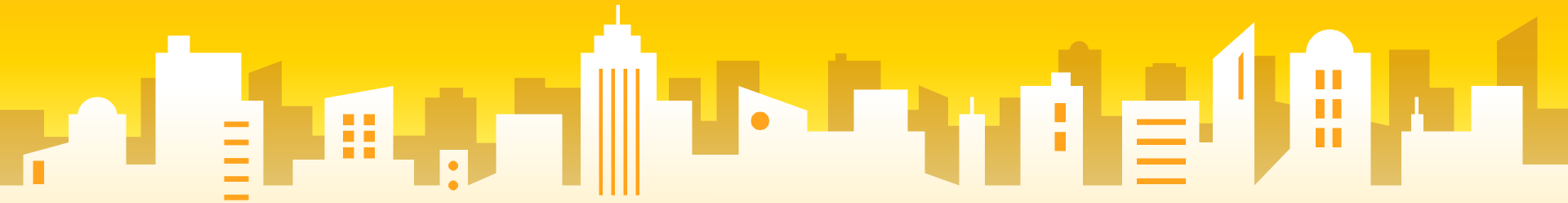
# getActionCommand

Figure out which action was performed by reading the `ActionCommand`.



# NameSurfer

Due Wednesday, March 7, 2018



# NameSurfer Assignment

Putting everything together!

1. Data Structures (arrays, ArrayList, HashMap)
2. Classes (multiple files, custom variables)
3. Graphics and Interactors

# NameSurfer Assignment

- NameSurfer.java
  - NameSurferEntry.java
  - NameSurferDataBase.java
  - NameSurferGraph.java
- NameSurferConstants.java (provided)

# Overview of NameSurfer

## Milestone 2+3:

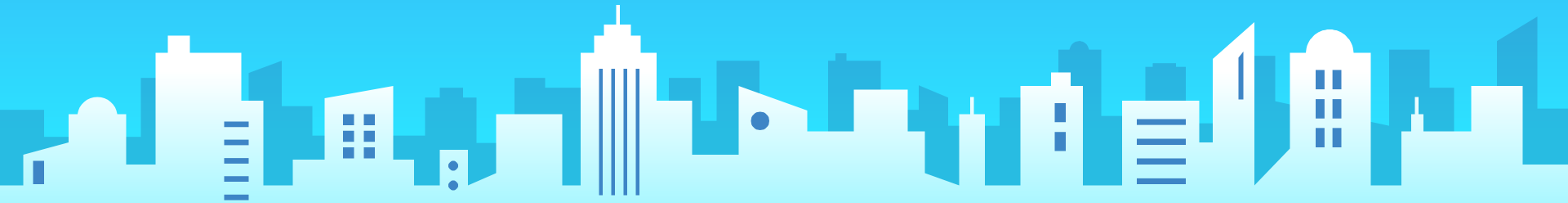
Read from a file, store the data of those files into **custom variable types** that you design.

## Milestone 1+4:

Setting up the console, interacting with the user.

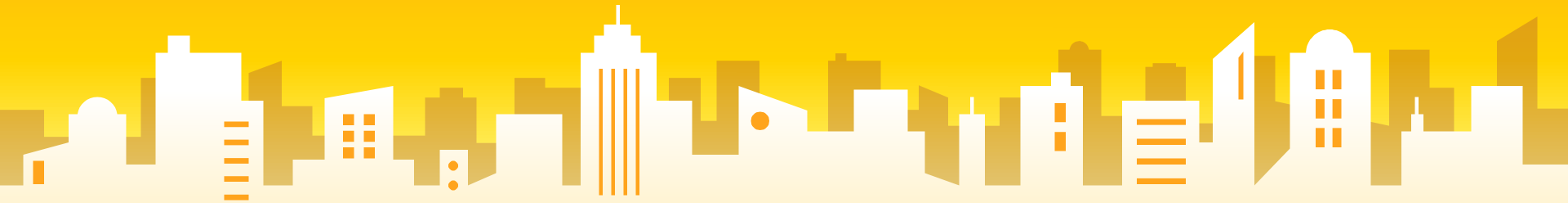
## Milestone 5:

When instructed by user, draw the graph.  
(i.e. a bunch of GLines!)



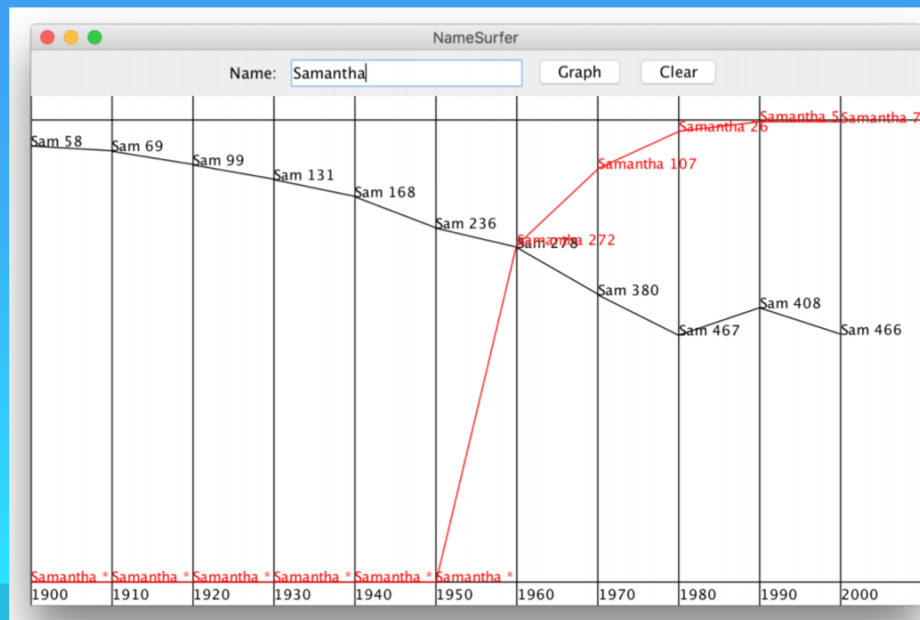
# Milestone 1: Interactors

Add a bunch of buttons and text boxes to the screen.

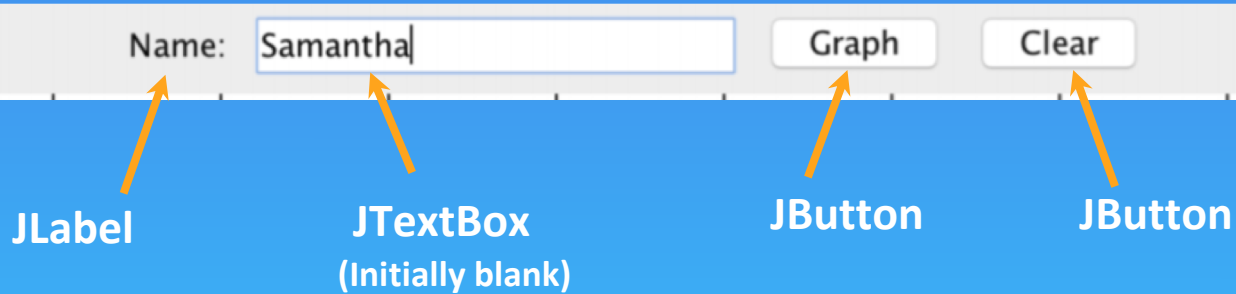


# Milestone 1: Interactors

- Set up all the interactors
- Test to make sure each interactor responds correctly



# Milestone 1: Interactors



If JTextBox entered, or JButton “Graph” pressed, should add Graph.

- Name is case-insensitive
- If no data found, don't add line.



# Milestone 1: Interactors (testing)

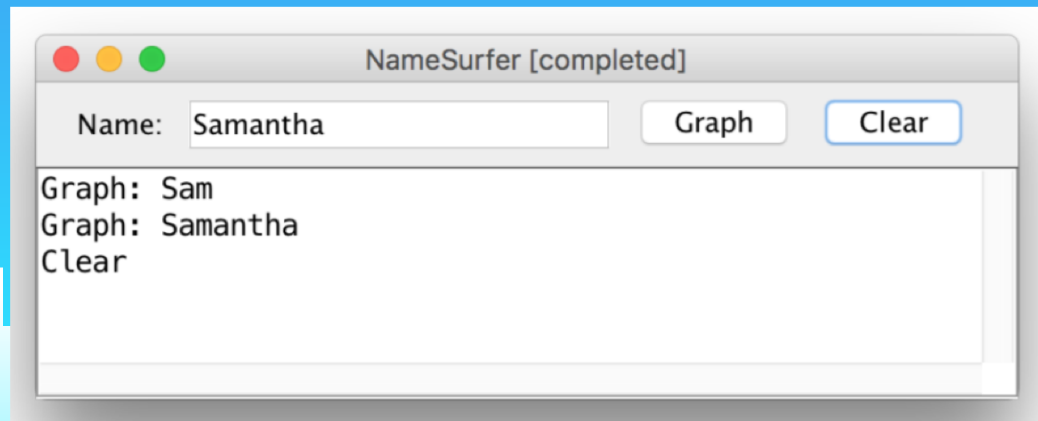
Change

```
public class NameSurfer extends Program
```

to

```
public class NameSurfer extends ConsoleProgram
```

You can check if your  
interactors are working!





# Milestone 2: NameSurferEntry

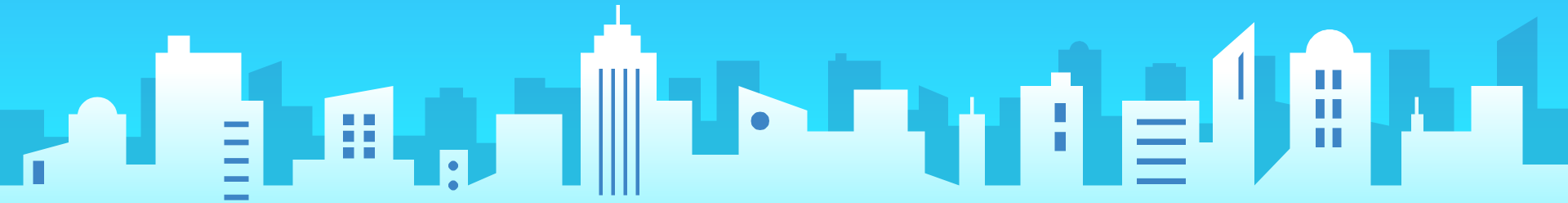
Goal: create a custom variable that stores the information  
in one NameSurfer entry

# NameSurferEntry

One line with name and popularity from 1900 to 2000.

eg. "Sam 58 69 99 131 168 236 278 380 467 408 466"

Need to store parse string and store information.



# The blueprint must define three things.

What information does this variable store?

What can you do with this variable type?

How do you create this variable type?

**Instance variables**

**Methods**

**Constructor**

## Example: NameSurferEntry

What information does this variable store?

???

What can you do with this variable type?

`getName, getRank, toString`

How do you create this variable type?

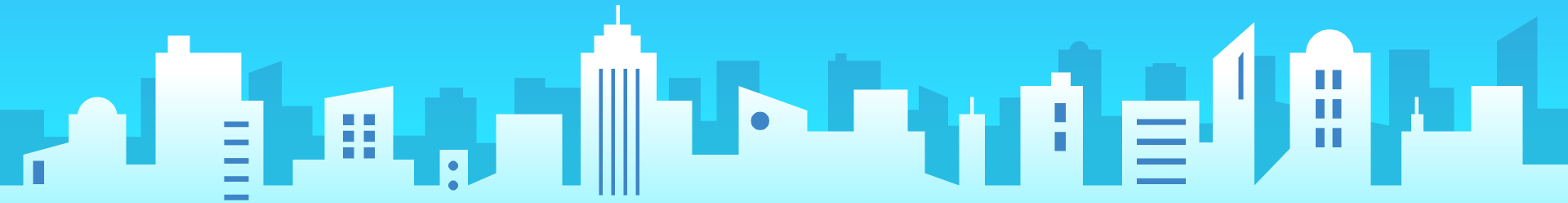
`NameSurferEntry(String dataline)`

# Instance Variables?

What pieces of information make sense as an instance variable?

What **type** do you use? What **collections** do you use?

All instance variables should be **private**.



# Constructor to Implement

You are given “Sam 58 69 99 131 168 236 278 380 467 408 466”

```
public NameSurferEntry (String dataline) {  
    // initialize instance variables using  
    dataline  
}
```

Relevant Concepts: String processing

Hint: there's a useful parsing method `str.split(" ")`.

# Methods to Implement

Dataline was: Sam 58 69 99 131 168 236 278 380 467 408 466

```
public String getName() {...} // should return "Sam"
```

```
public int getRank(int decadesSince1900) {...} // getRank (3) returns 131
```

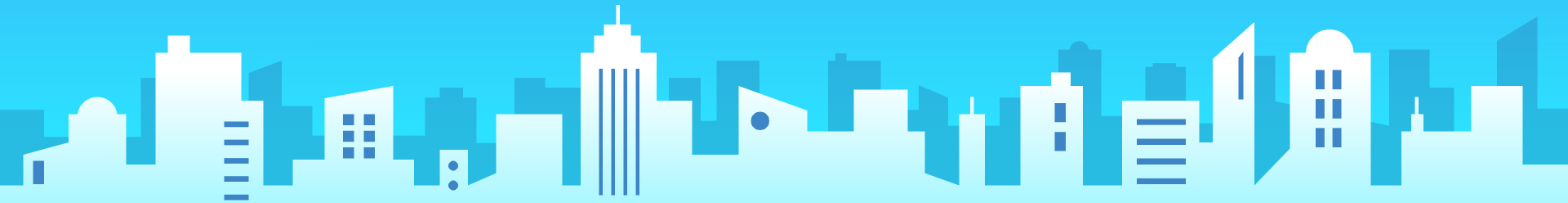
```
public String toString() {...} // should return "Sam [58, 69, ..., 466]"
```

Relevant Concepts: String processing, data structures



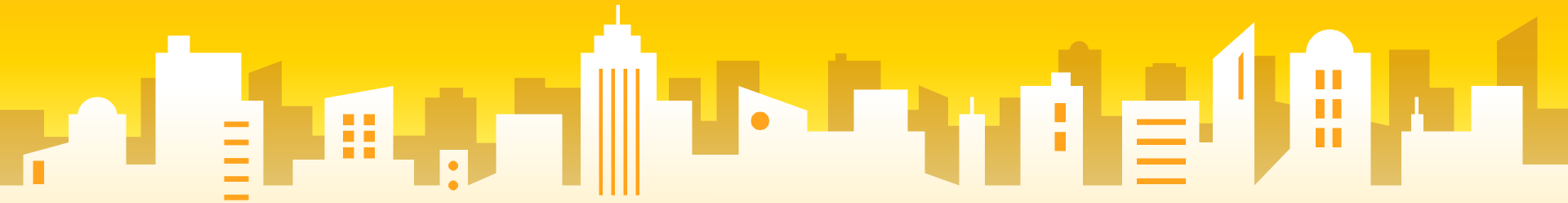
# Other Methods?

You may (and probably should) implement more **private** methods, but may not add/remove/change the headers of any **public** methods



# Milestone 3: NameSurferDataBase

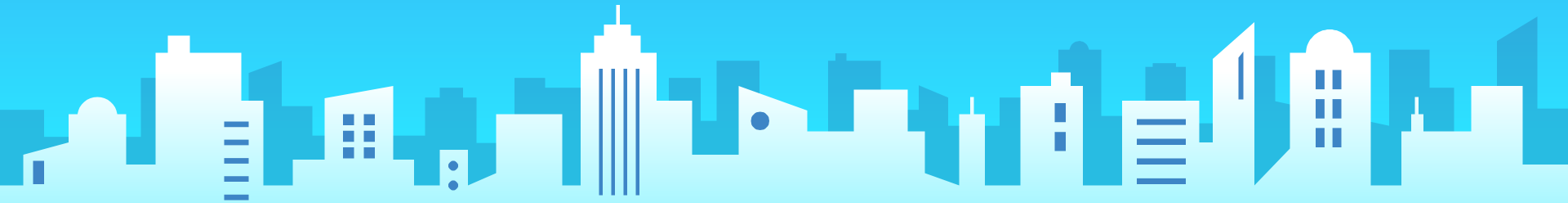
Goal: store collection of NameSurferEntries



# NameSurferDataBase

Text file contains data for one name/one line.

Need to read file and store a collection of NameSurferEntries (each line should be stored as a NameSurferEntry)



# The blueprint must define three things.

What information does this variable store?

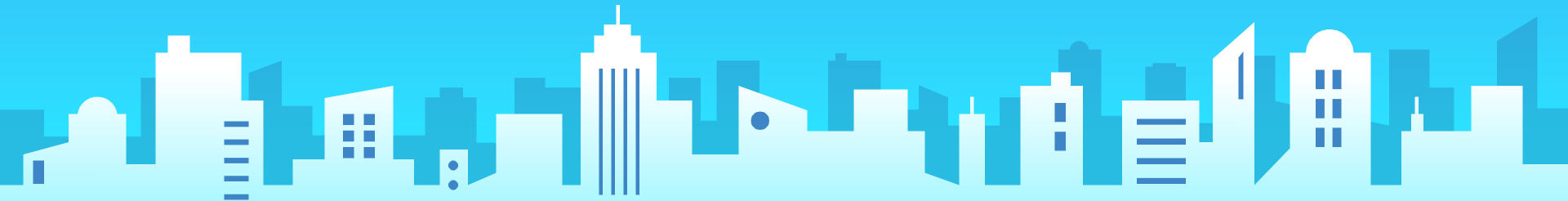
What can you do with this variable type?

How do you create this variable type?

**Instance variables**

**Methods**

**Constructor**



# Example: NameSurferDataBase

What information does this variable store?

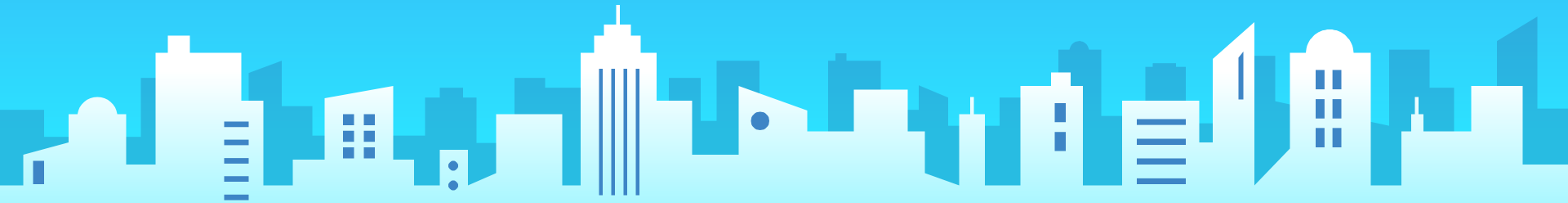
???

What can you do with this variable type?

`findEntry(String name)`

How do you create this variable type?

`NameSurferDataBase(String filename)`

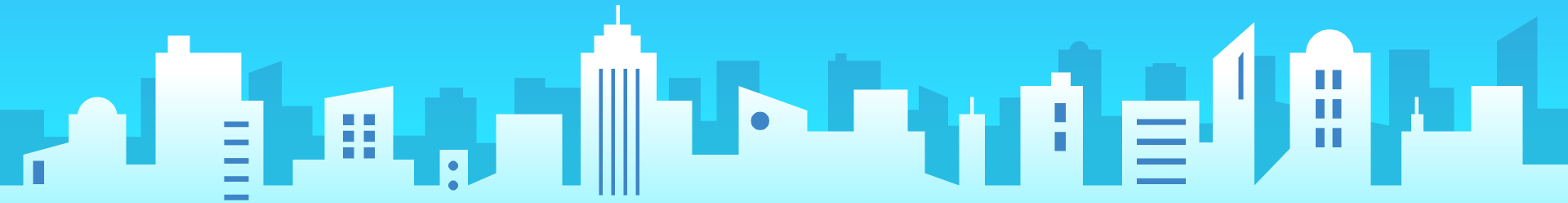


# Instance Variables?

What pieces of information make sense as an instance variable?

What **type** do you use? What **collections** do you use?

All instance variables should be **private**.



# Constructor to Implement

```
public NameSurferDataBase (String filename) {  
    // initialize instance variables using file  
    // probably expect some file reading  
    here  
}
```

Relevant Concepts: File reading

# Methods to Implement

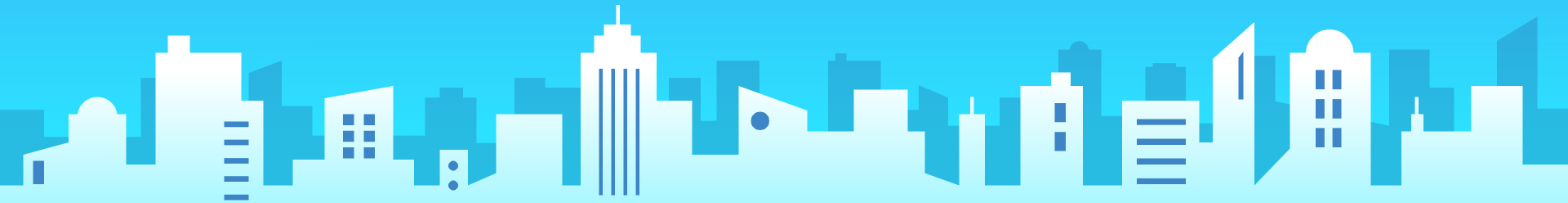
```
public NameSurferEntry findEntry(String name) {...}  
// returns a NameSurferEntry (custom variable you defined!)
```

Relevant Concepts: variables, data structures

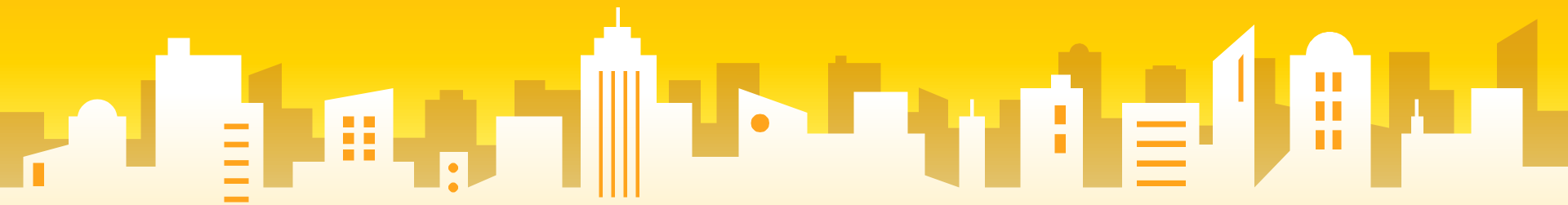


# Other Methods?

You may (and probably should) implement more **private** methods, but may not add/remove/change the headers of any **public** methods



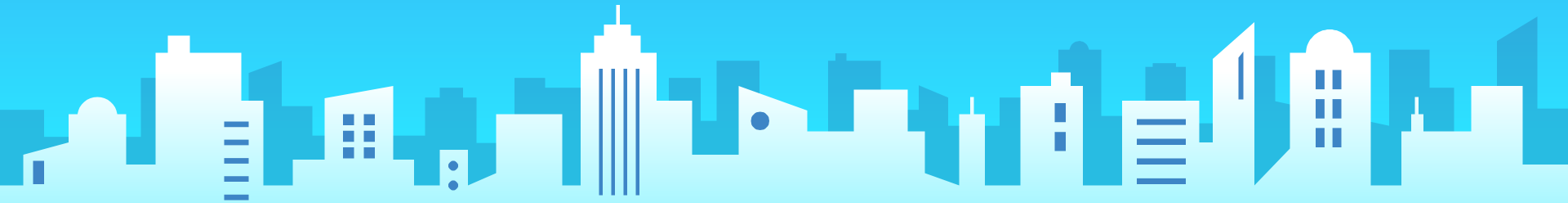
# Milestone 4: NameSurferGraph (part 1)



# NameSurferGraph

Responsible for graphing the data for each name.

Should store all entries that are currently graphed, so it can redraw if window is resized.



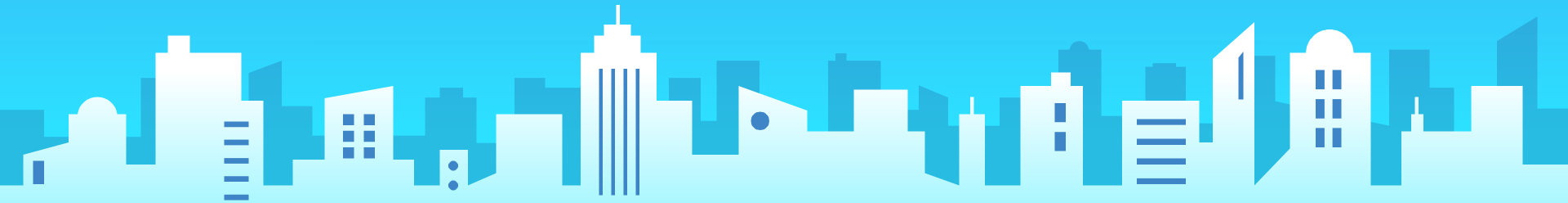
# A NameSurferGraph is a GCanvas!

```
public class NameSurferGraph extends GCanvas
```

Therefore, you can call methods like `getWidth`, `add`, `remove`.

You can add `GLines` to a `NameSurferGraph`!

(see lecture/section for examples)

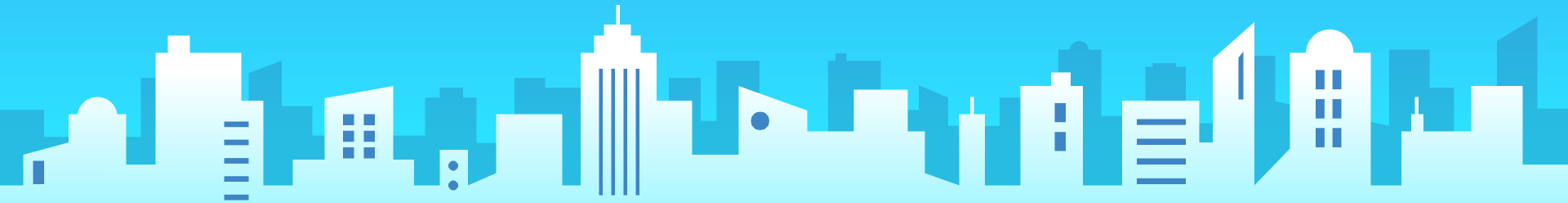


# Step 1: Grid Lines

Set up the Grid Lines.

(this should later be incorporated into the update method)

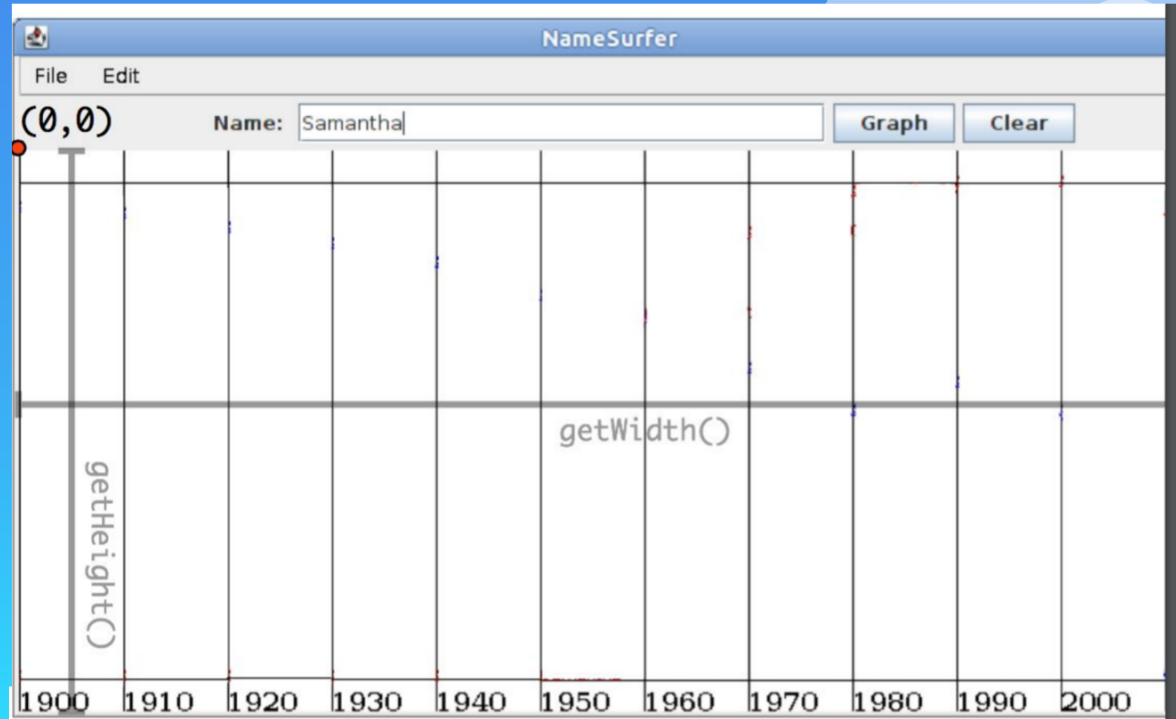
Relevant Concepts: calculating coordinates.



# Important Details

Year 1900  
(at left of screen)

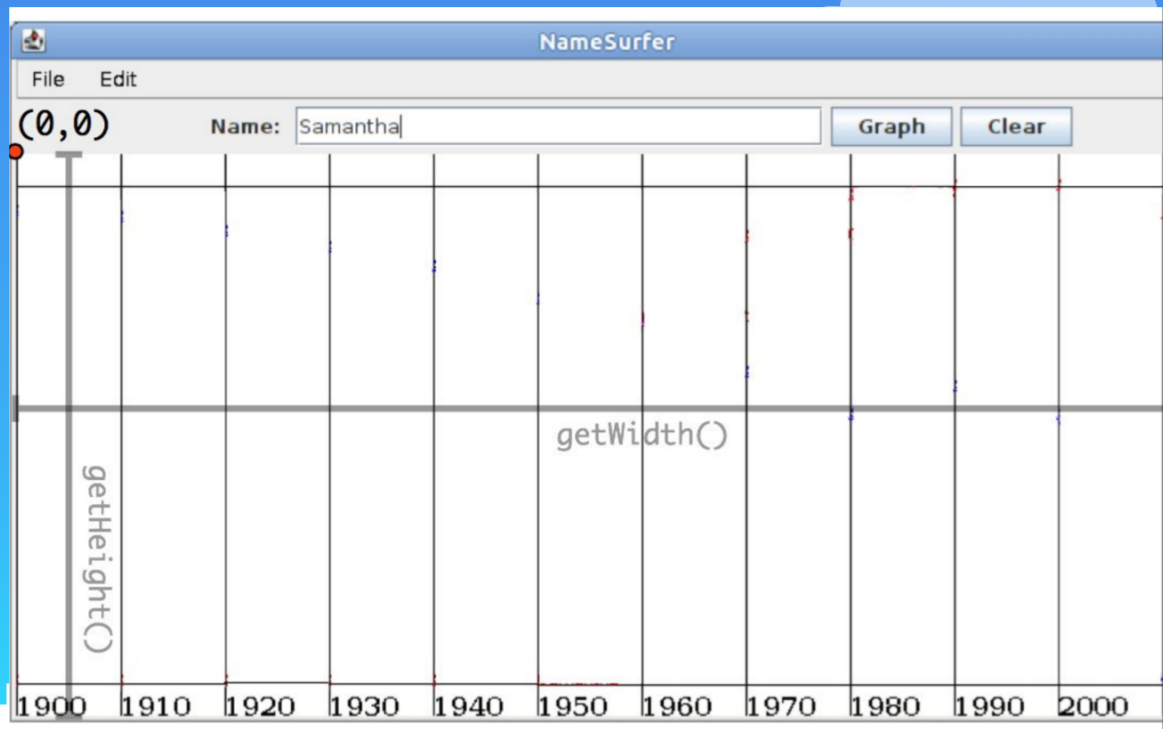
Year 2000  
(a bit off from right of screen!)



# Important Details

Can use `getHeight()`,  
`getWidth()`

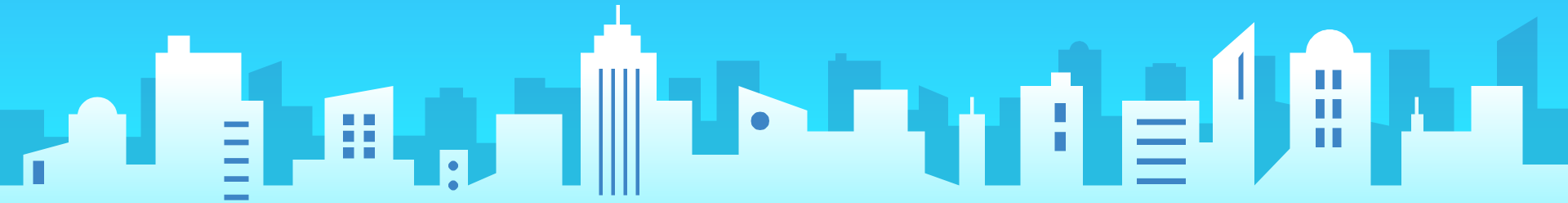
**Very common bug:**  
Cannot call `getHeight()` before  
adding `NameSurferGraph`  
to the canvas.



## Step 2: Managing NameSurferEntries

Figure out a way to store the NameSurferEntries that should be drawn.

Relevant Concepts: Classes and instances, data structures.





# In NameSurfer.java

```
public void init() {  
    graph = new NameSurferGraph(); // new instance of  
    NameSurferGraph  
    add(graph); // now our graph is on the canvas - can use getHeight()  
    now  
}  
public void run() {  
    // something to create NameSurferEntry entry  
    graph.add(entry); // does not add the entry to graph yet  
    graph.update(); // now it does!
```

# The blueprint must define three things.

What information does this variable store?

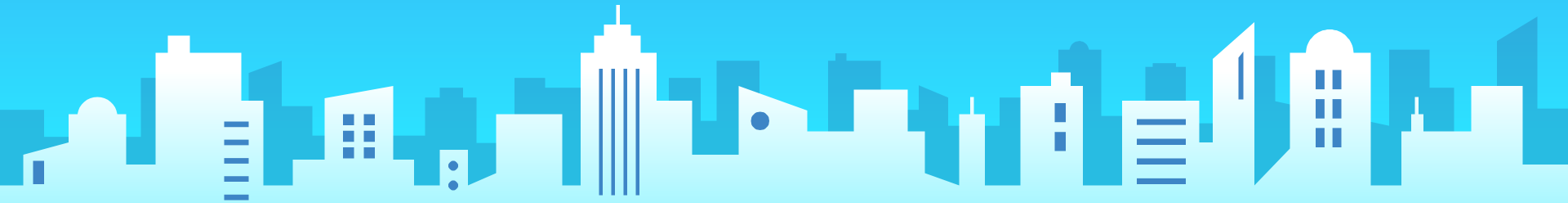
What can you do with this variable type?

How do you create this variable type?

**Instance variables**

**Methods**

**Constructor**



# Example: NameSurferGraph

What information does this variable store?

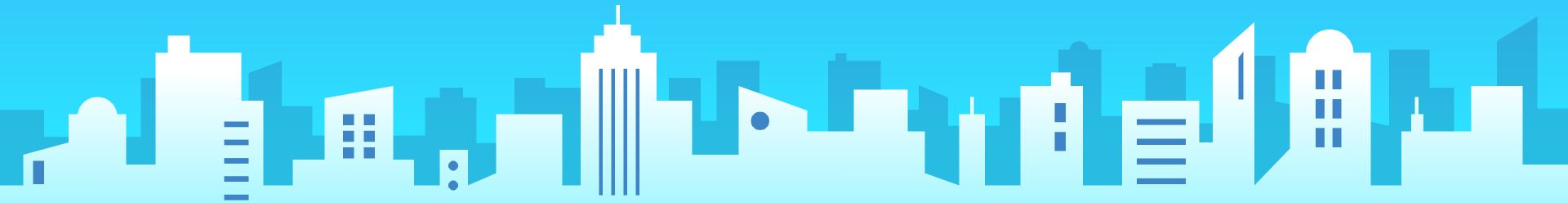
???

What can you do with this variable type?

Clear, addEntry, update

How do you create this variable type?

NameSurferGraph()

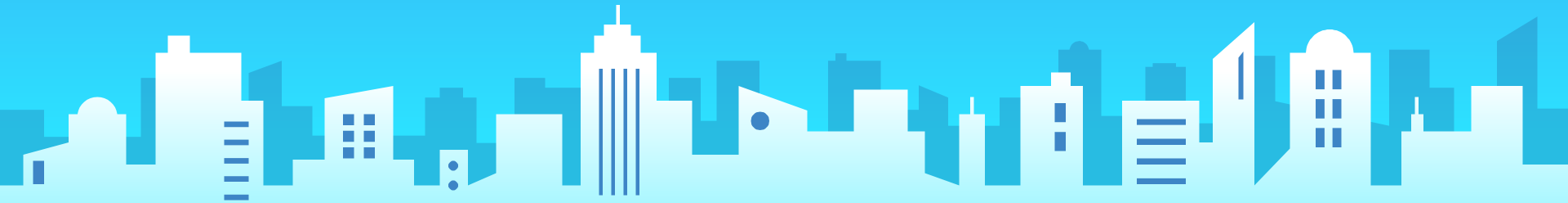


# Instance Variables?

What pieces of information make sense as an instance variable?

What **type** do you use? What **collections** do you use?

All instance variables should be **private**.



# Constructor to Implement

```
public NameSurferGraph () {  
    // initialize instance variables  
}
```

## Methods to Implement

*// does not actually clear graph, just deletes the entries.*

```
public void clear(String name) {...}
```

*// does not actually draw graph, just stores the entry.*

```
public void addEntry(NameSurferEntry entry) {...}
```

*// delete all GObjects and reassemble them all*

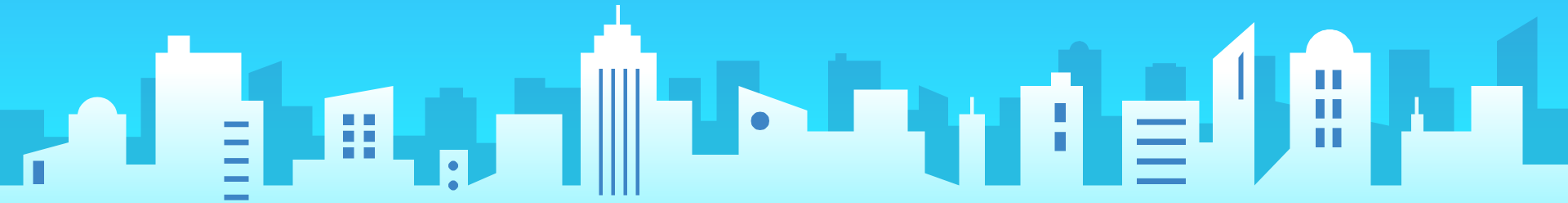
```
public void update() {...}
```

# update()?

The add and clear method should not add/remove any GObjects from the graph.

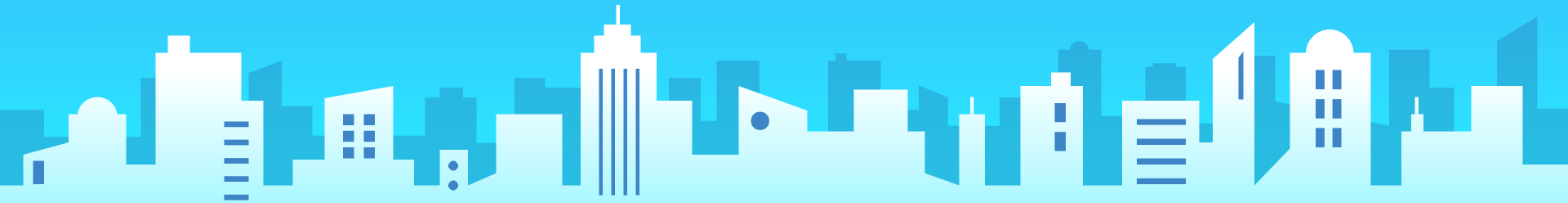
The update method will remove **everything** from the graph, then reassemble **everything** based on the NameEntries stored.

Why? Only update is called when window is resized - must change **everything**, even the grid lines.




# Other Methods?

You may (and probably should) implement more **private** methods, but may not add/remove/change the headers of any **public** methods

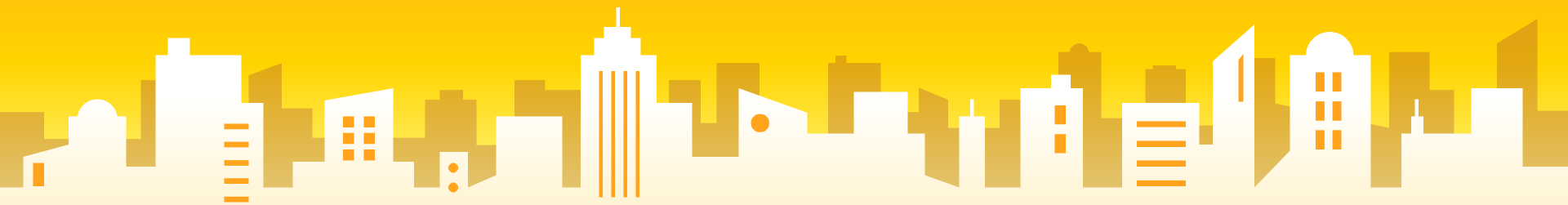






# Milestone 5: NameSurferGraph (part 2)

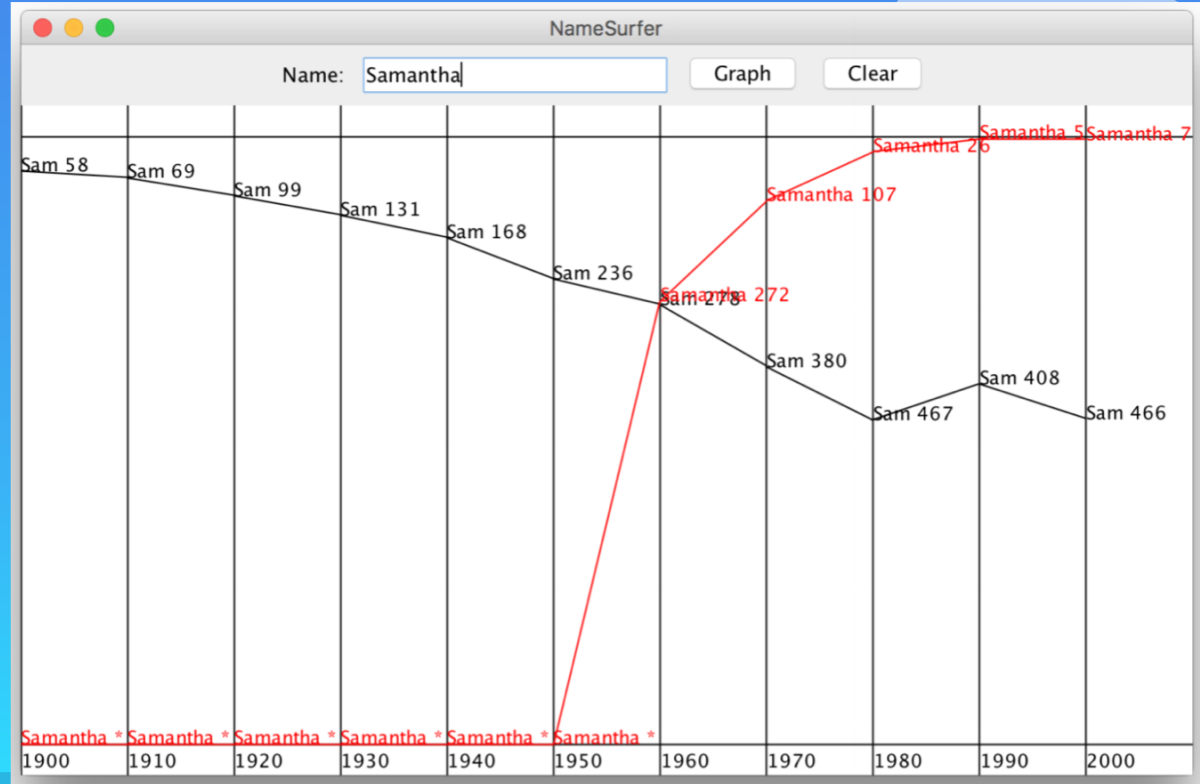
Goal: Finish implementing `update()`, and make interactors work!



# Important Details

Colors cycle: black, red, blue, magenta, repeat.

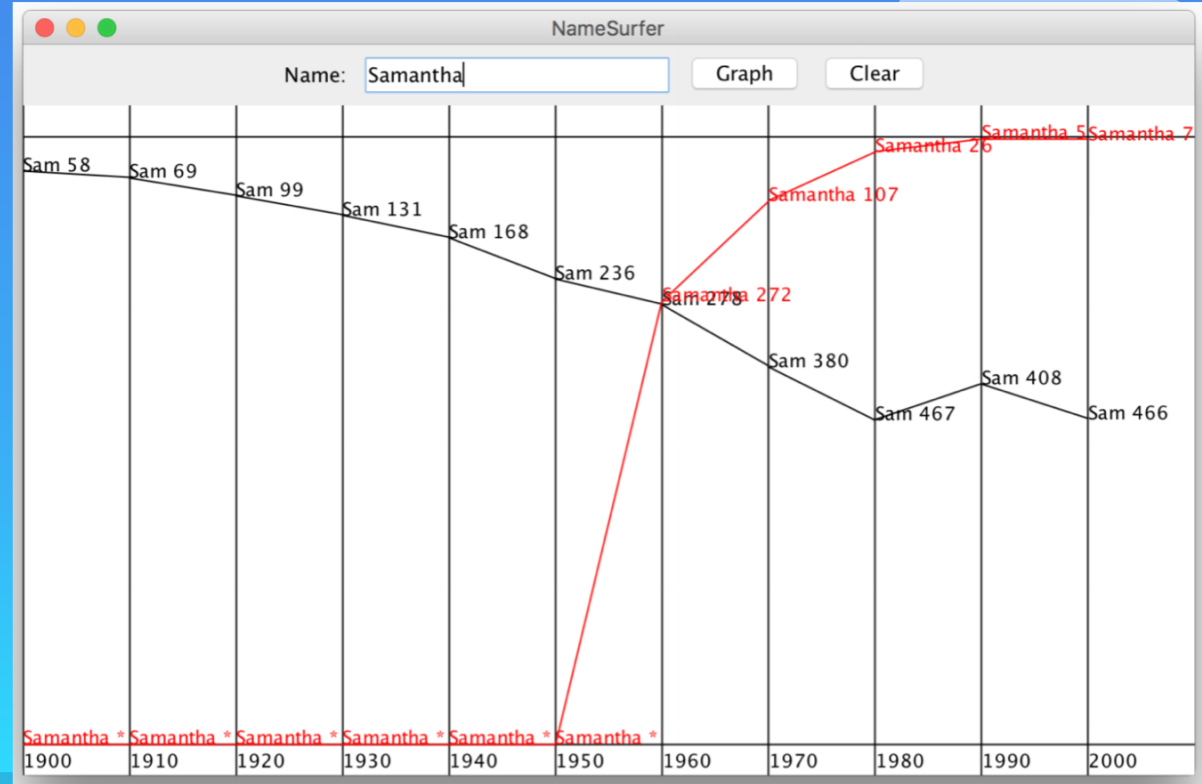
Rank 0 (not in top 1000): store at bottom (MAX\_RANK), with an asterisk (\*).



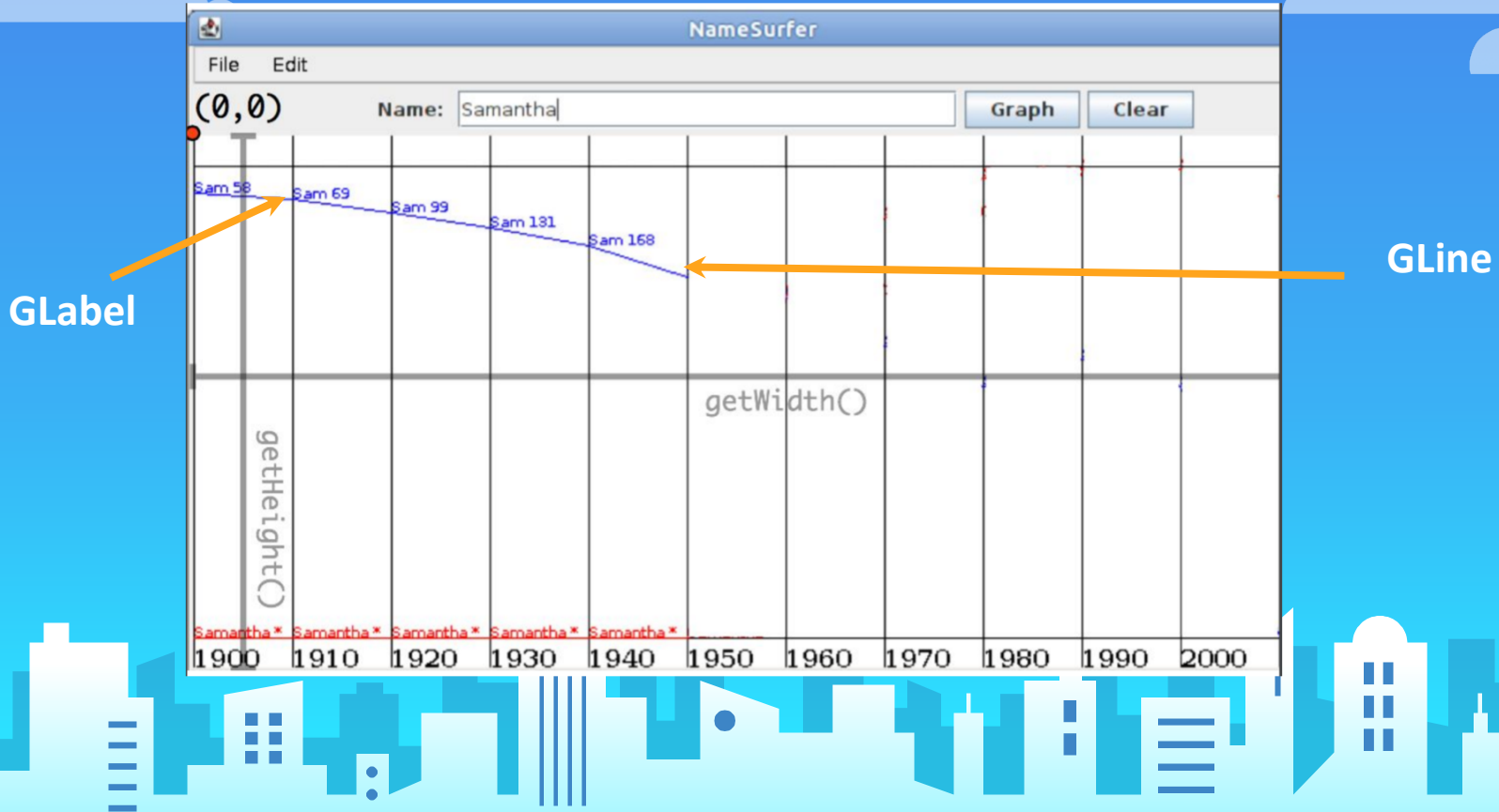
# Important Details

Rank 1 at top  
(notice the margin)

Rank MAX\_RANK  
at the bottom  
(also notice margin)

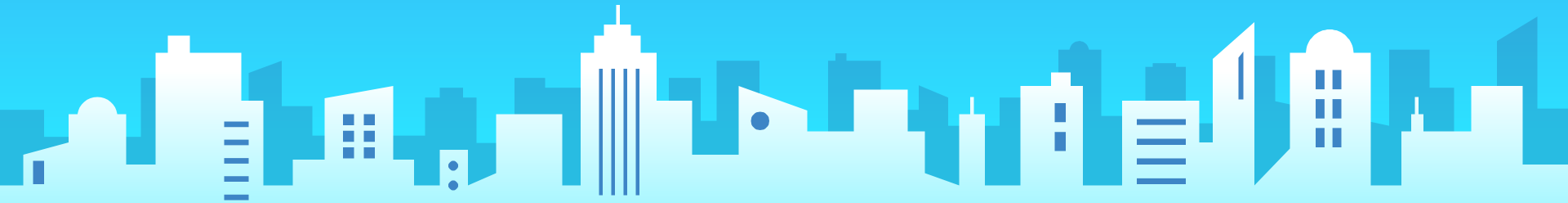


# Drawing the GLines/GLabels

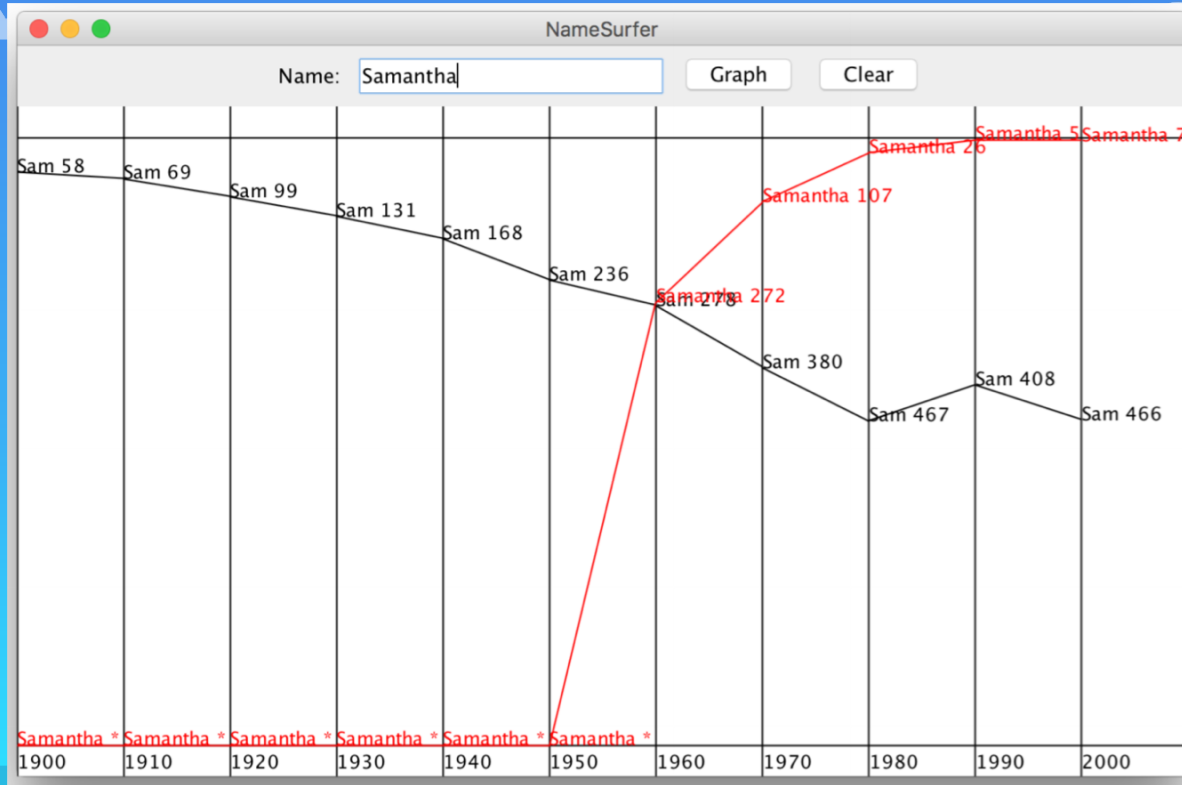


# Finally...

**Revisit the interactors from milestone 1, and change the code so the user can type in a name and click the button.**



# Fully Functioning Program



# Common Pitfalls: Off-by-1

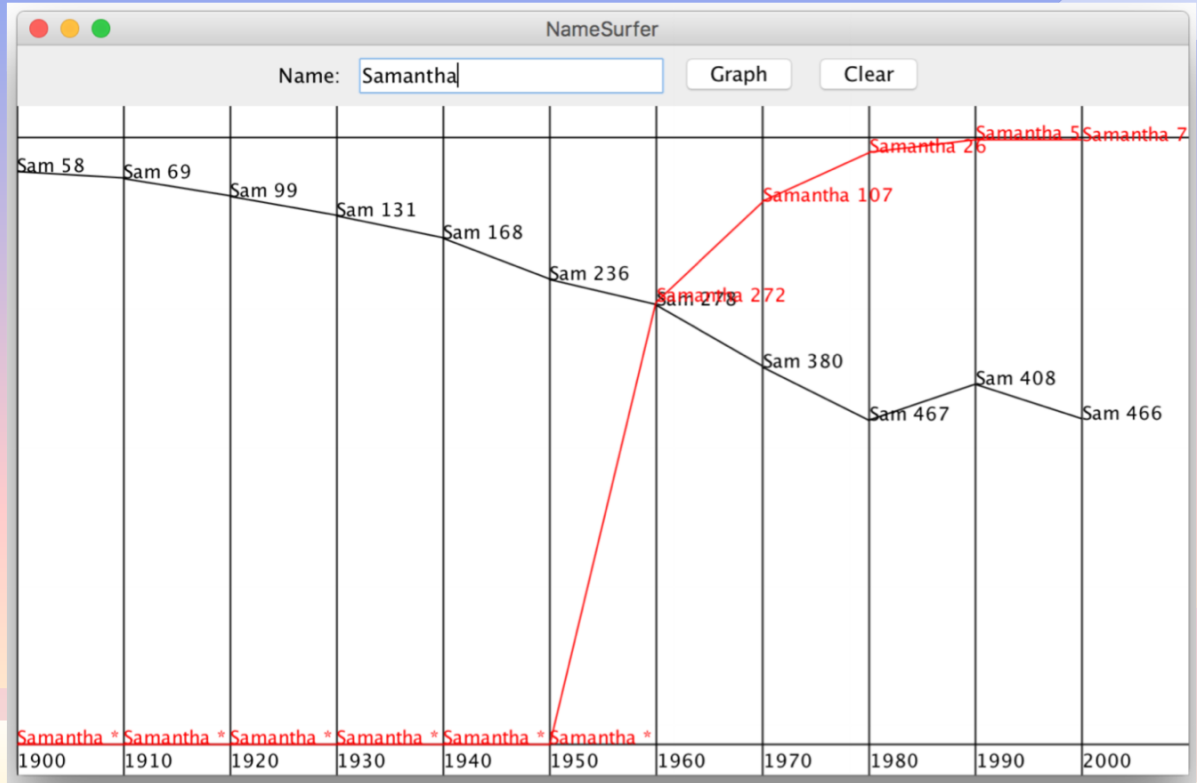
Notice there are

11 GLabels

11 decade lines

**BUT**

10 connecting GLines



# Common Pitfalls: getHeight() is zero!

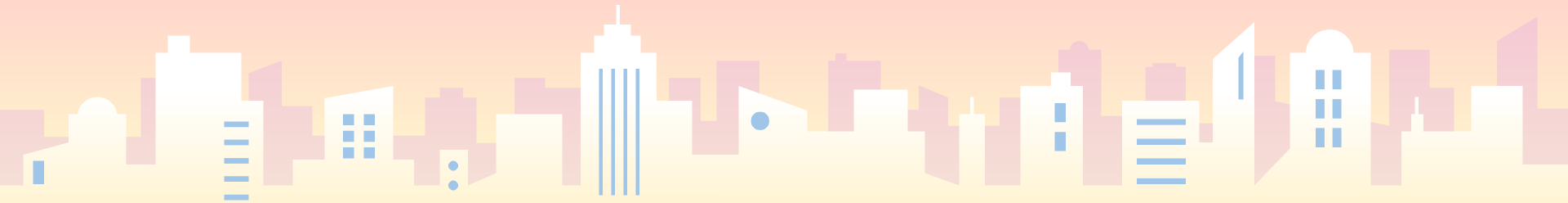
You can only use getHeight() after graph is added to the canvas.

- **Can't use it in NameSurferGraph constructor**
  - **Can't use it in NameSurfer init**



# Common Pitfalls: Another off by 1

**The ranks go from 1 to 1000. How many space divisions do we make? (Be careful!)**



# Common Pitfalls: Not using constants

```
1 public interface NameSurferConstants {
2
3
4 /** The width of the application window */
5     public static final int APPLICATION_WIDTH = 800;
6
7 /** The height of the application window */
8     public static final int APPLICATION_HEIGHT = 600;
9
10 /** The name of the file containing the data */
11     public static final String NAMES_DATA_FILE = "names-data.txt";
12
13 /** The width of the text field in the NORTH of the window */
14     public static final int TEXT_FIELD_WIDTH = 16;
15
16 /** The first decade in the database */
17     public static final int START_DECADE = 1900;
18
19 /** The number of decades */
20     public static final int NDECADES = 11;
21
22 /** The maximum rank in the database */
23     public static final int MAX_RANK = 1000;
24
25 /** The number of pixels to reserve at the top and bottom */
26     public static final int GRAPH_MARGIN_SIZE = 20;
27
28 /** The number of pixels between the baseline of the decade labels and the bottom of the window */
29     public static final int DECADE_LABEL_MARGIN_SIZE = GRAPH_MARGIN_SIZE / 4;
30 }
31 }
```

# Common Pitfalls: Changing Method Names

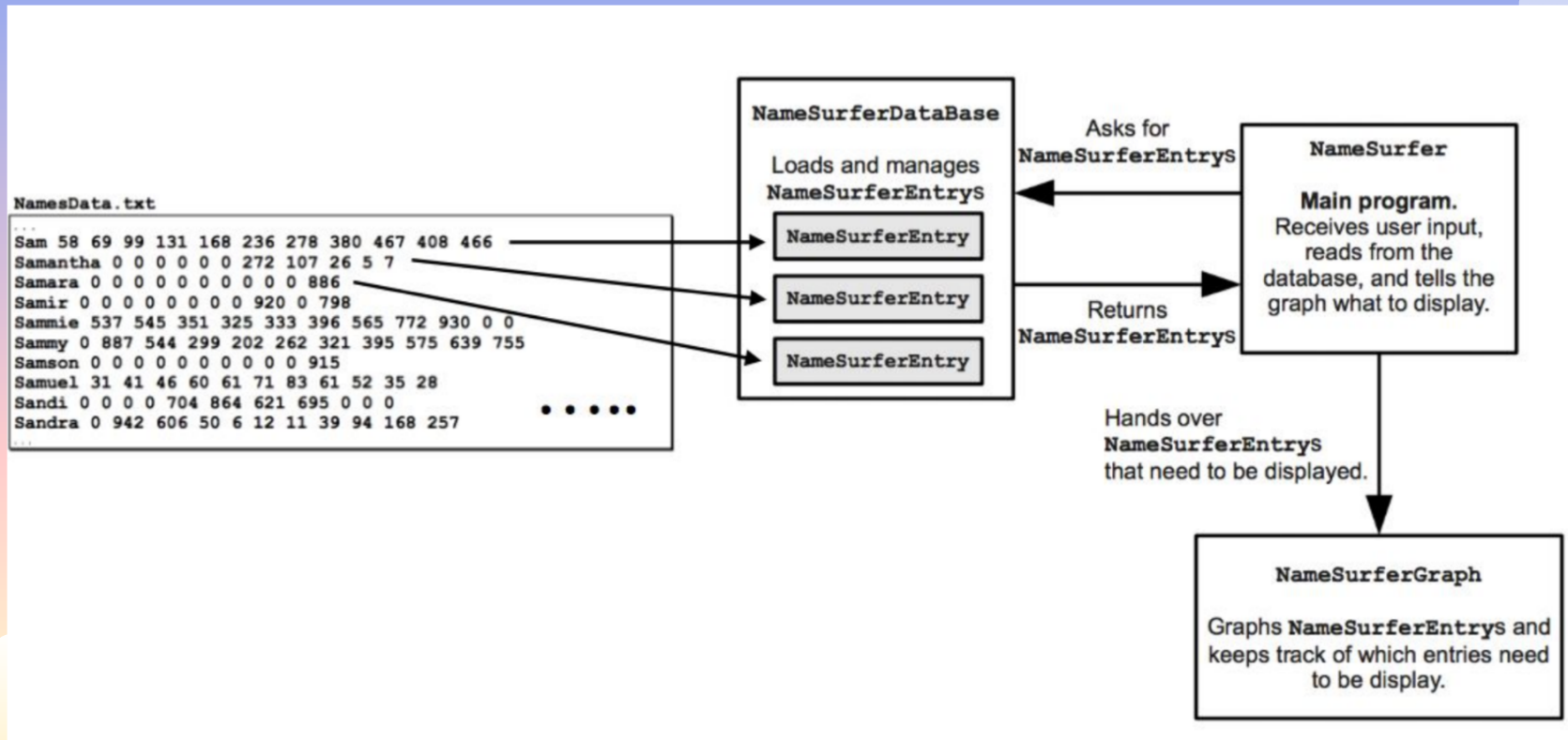
Do not add/remove/change the headers of any **public** method.

You may (and should) add **private** methods.

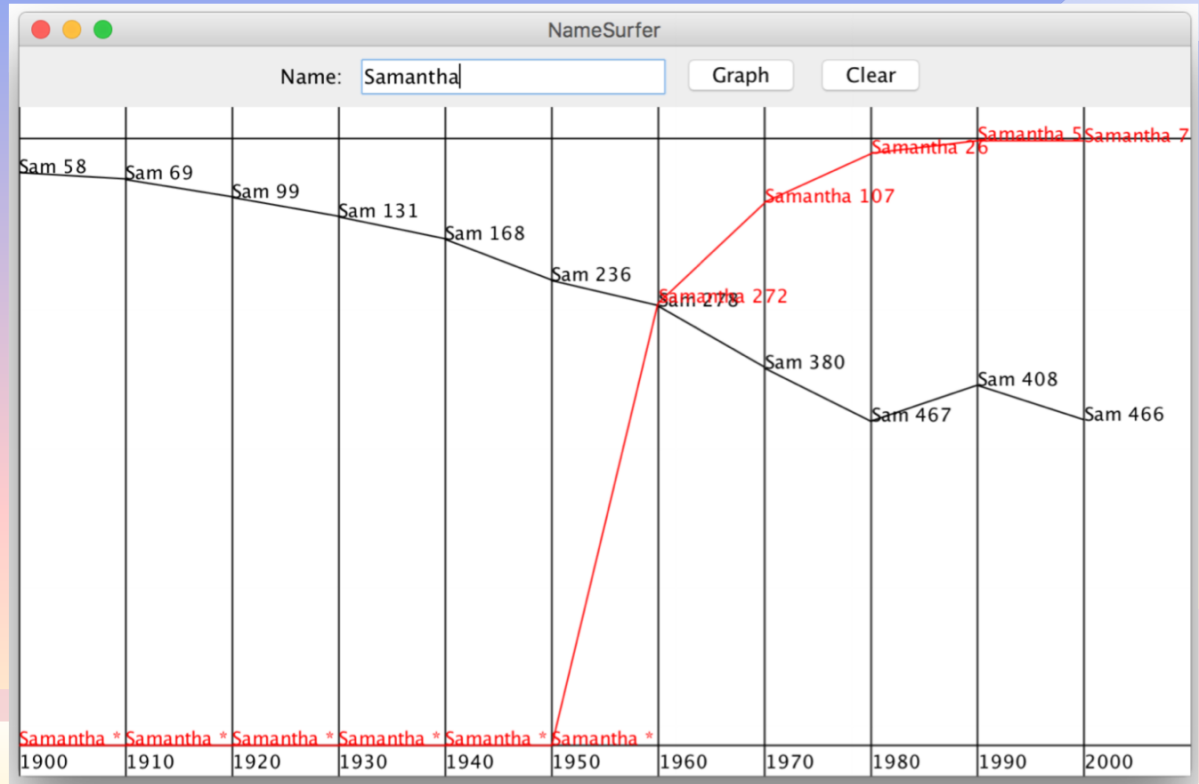
# What is good style?

- **All the guidelines from Assignment 1-5.**
- **Separating responsibilities between classes.**
- **Using instance variables appropriately.**
- **Choosing appropriate data structures to use.**

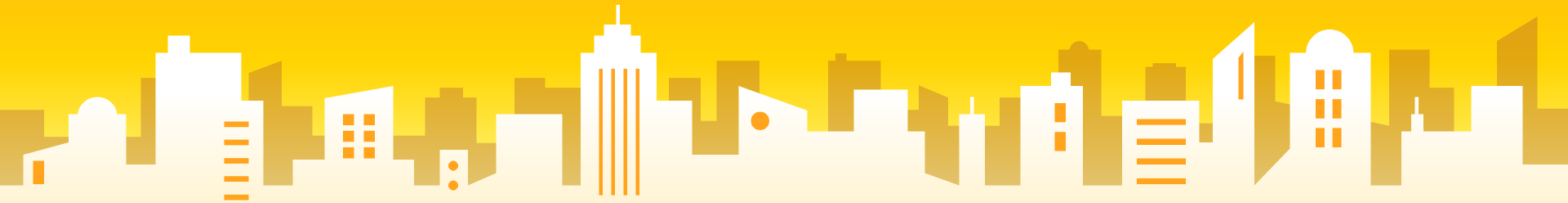
# NameSurfer Overview



# Ideas for Extensions?



**Any Questions?**



Good luck on NameSurfer!

Have a good night, and see you at the LaIR.

