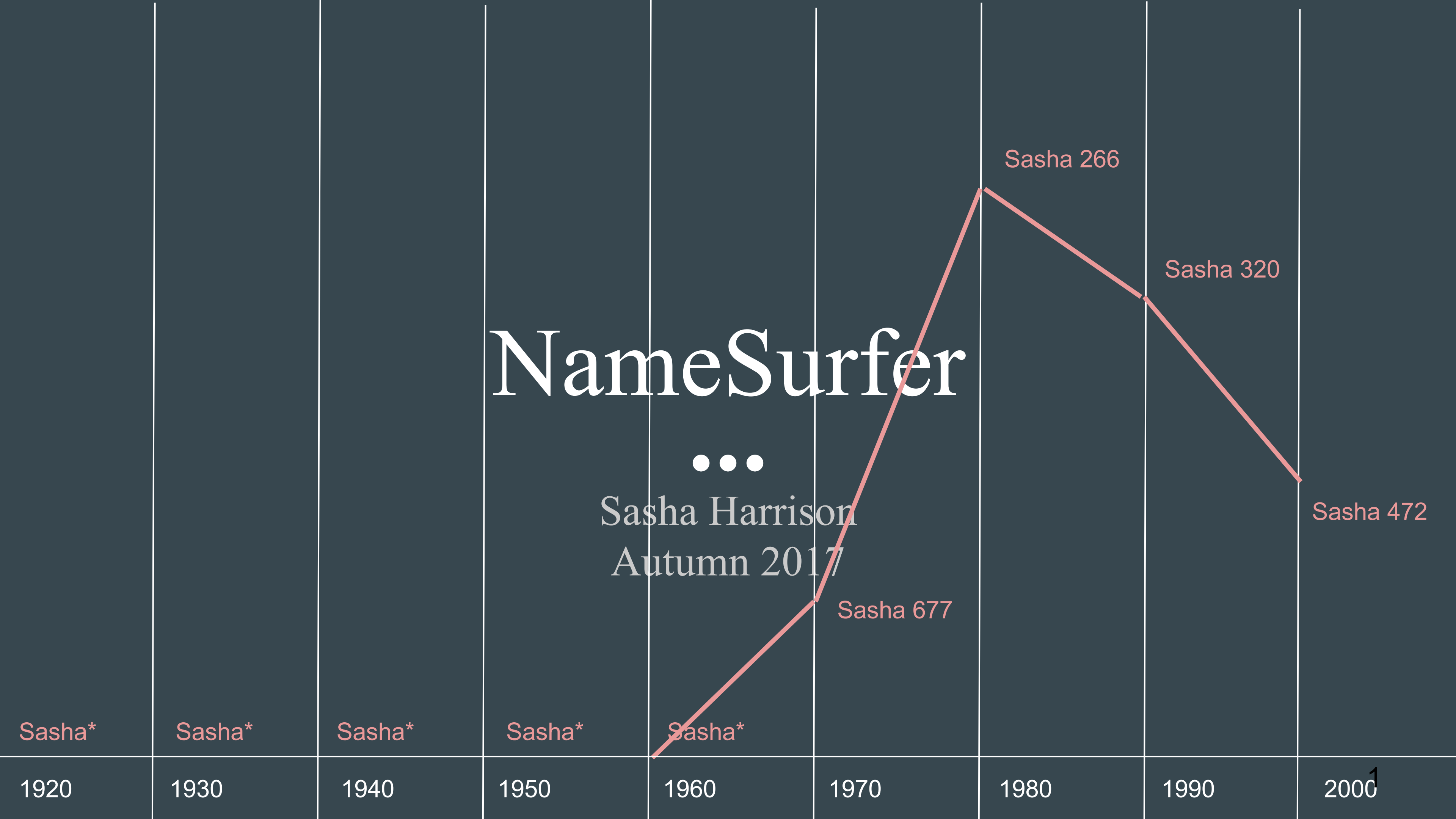
NameSurfer

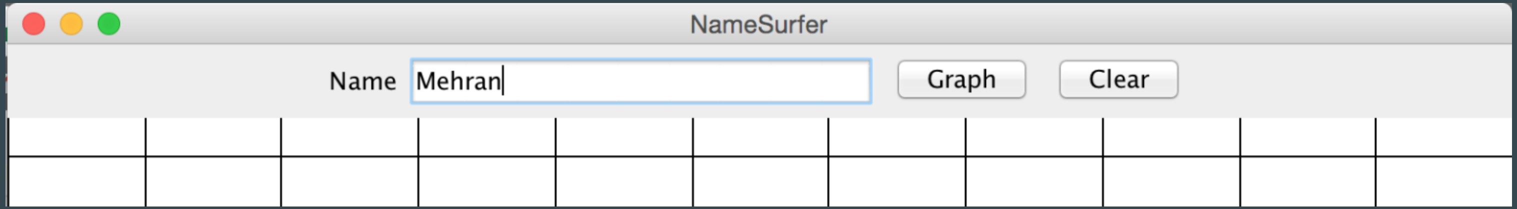
Sasha Harrison
Autumn 2017



Key Concepts

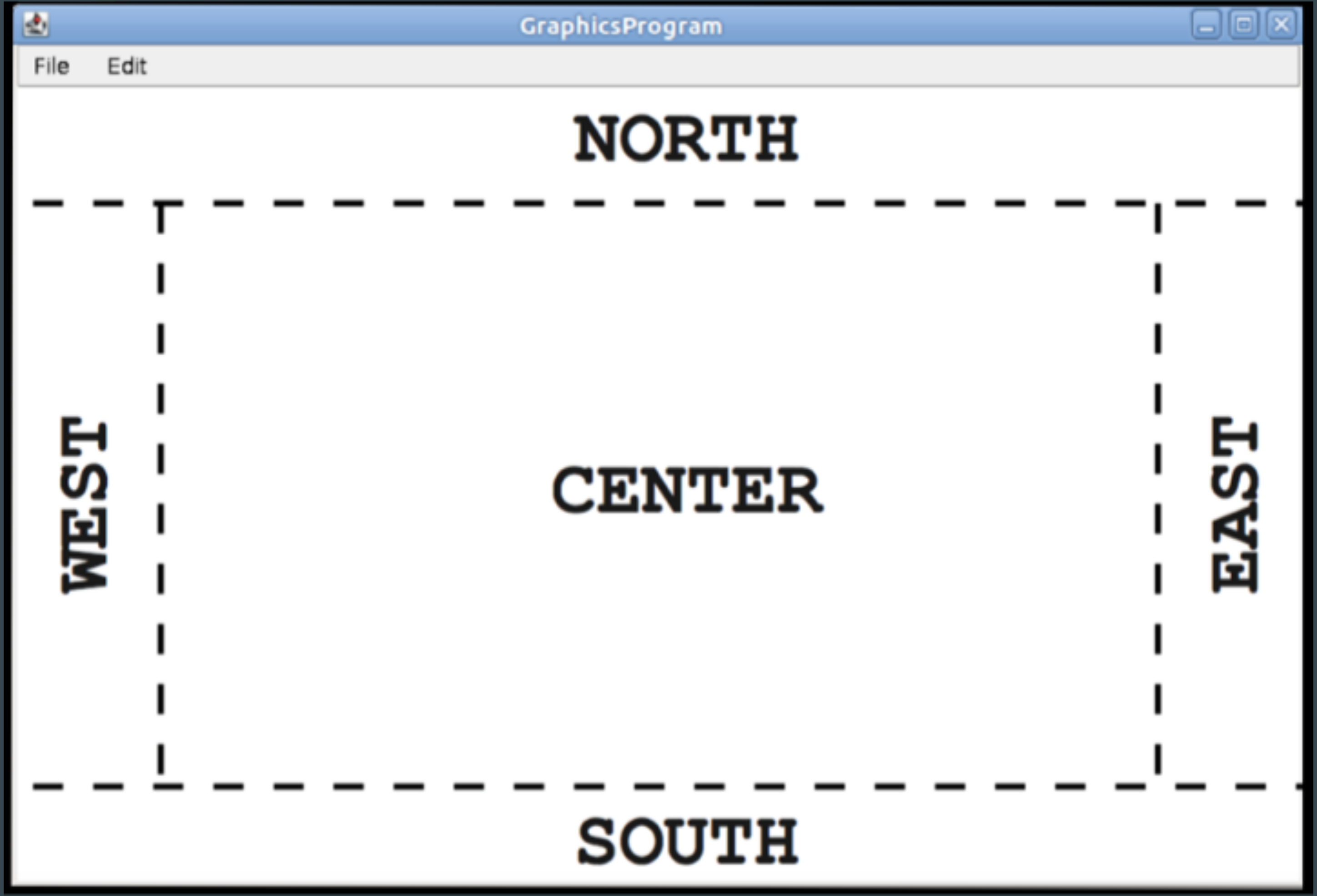
Interactors!

Main Idea: Let user interact with application by giving them choices



What does it look like in code?

```
/* JButton constructor takes String to display on
button */
JButton button = new JButton("Add");
add(button, NORTH); // Add to top of screen
// JTextField constructor takes length of text
JTextField field = new JTextField(25);
// Listen for "ENTER" in text field
field.addActionListener(this);
add(field, NORTH);
```


GraphicsProgram
File
Edit
NORTH
WEST
CENTER
EAST
SOUTH

More on Interactors

- Add them in a specific *region* on screen (usually not CENTER! That's where the canvas goes)
- addActionListeners() in your main program
- Implement the actionPerformed method to respond to action events (just like you did for mouseListeners)

NameSurfer!

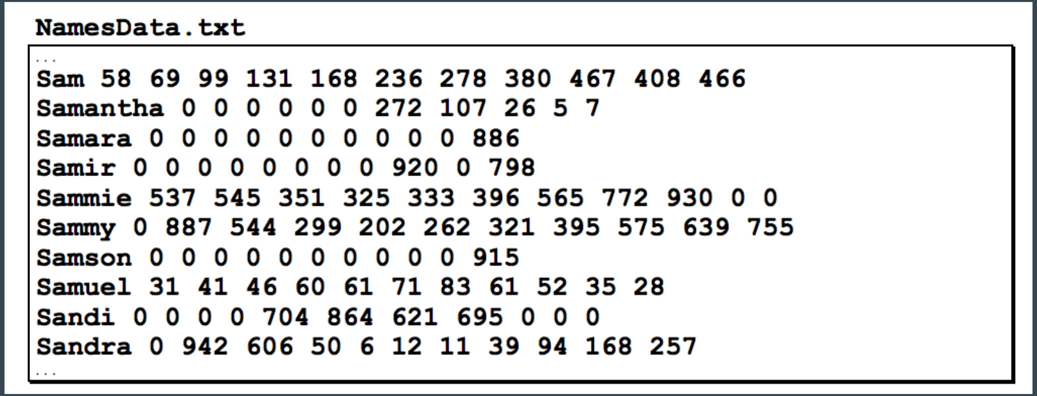
- Due at 1:30PM on Wednesday, Nov. 29 (after break)
- Practice with arrays, ArrayLists, HashMaps
- Practice with multiple classes/code files
- Practice with INTERACTORS!

How does it work?

Broad overview:

Using Census data on 1000 most popular baby names in the U.S. in the last century, take a name and graph its popularity from 1900 to 2000.

1 means most popular, 1000 means least popular



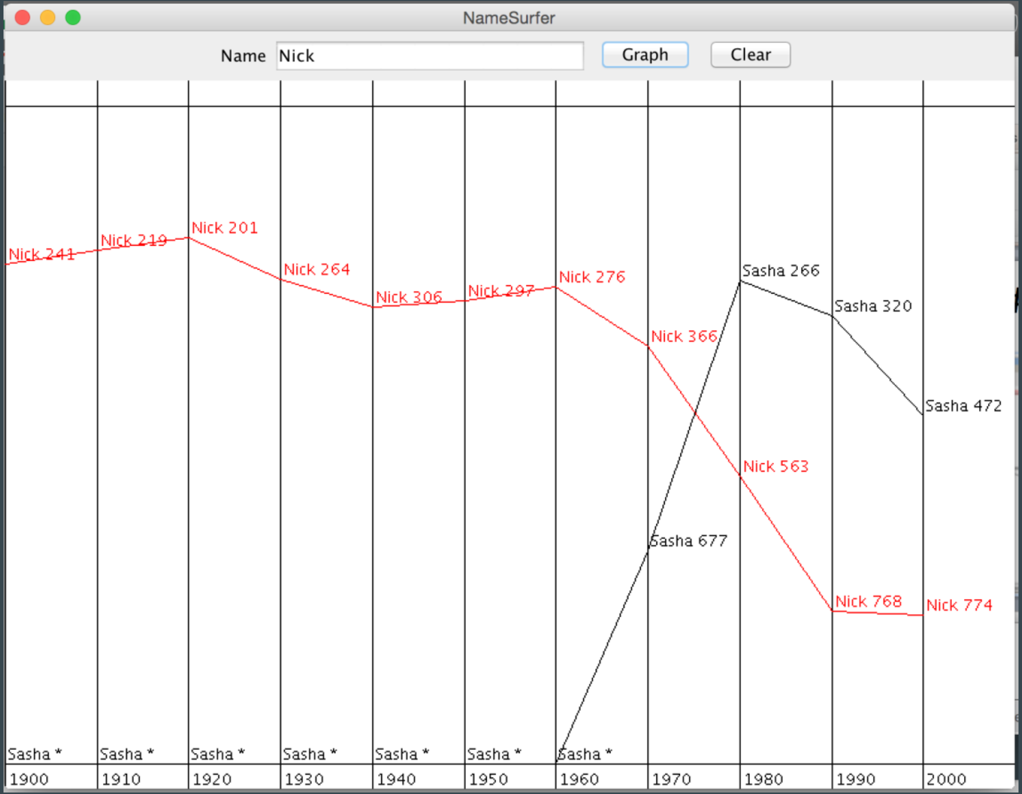
NamesData.txt

```
...
Sam 58 69 99 131 168 236 278 380 467 408 466
Samantha 0 0 0 0 0 0 272 107 26 5 7
Samara 0 0 0 0 0 0 0 0 0 0 886
Samir 0 0 0 0 0 0 0 0 920 0 798
Sammie 537 545 351 325 333 396 565 772 930 0 0
Sammy 0 887 544 299 202 262 321 395 575 639 755
Samson 0 0 0 0 0 0 0 0 0 0 915
Samuel 31 41 46 60 61 71 83 61 52 35 28
Sandi 0 0 0 0 704 864 621 695 0 0 0
Sandra 0 942 606 50 6 12 11 39 94 168 257
...
```

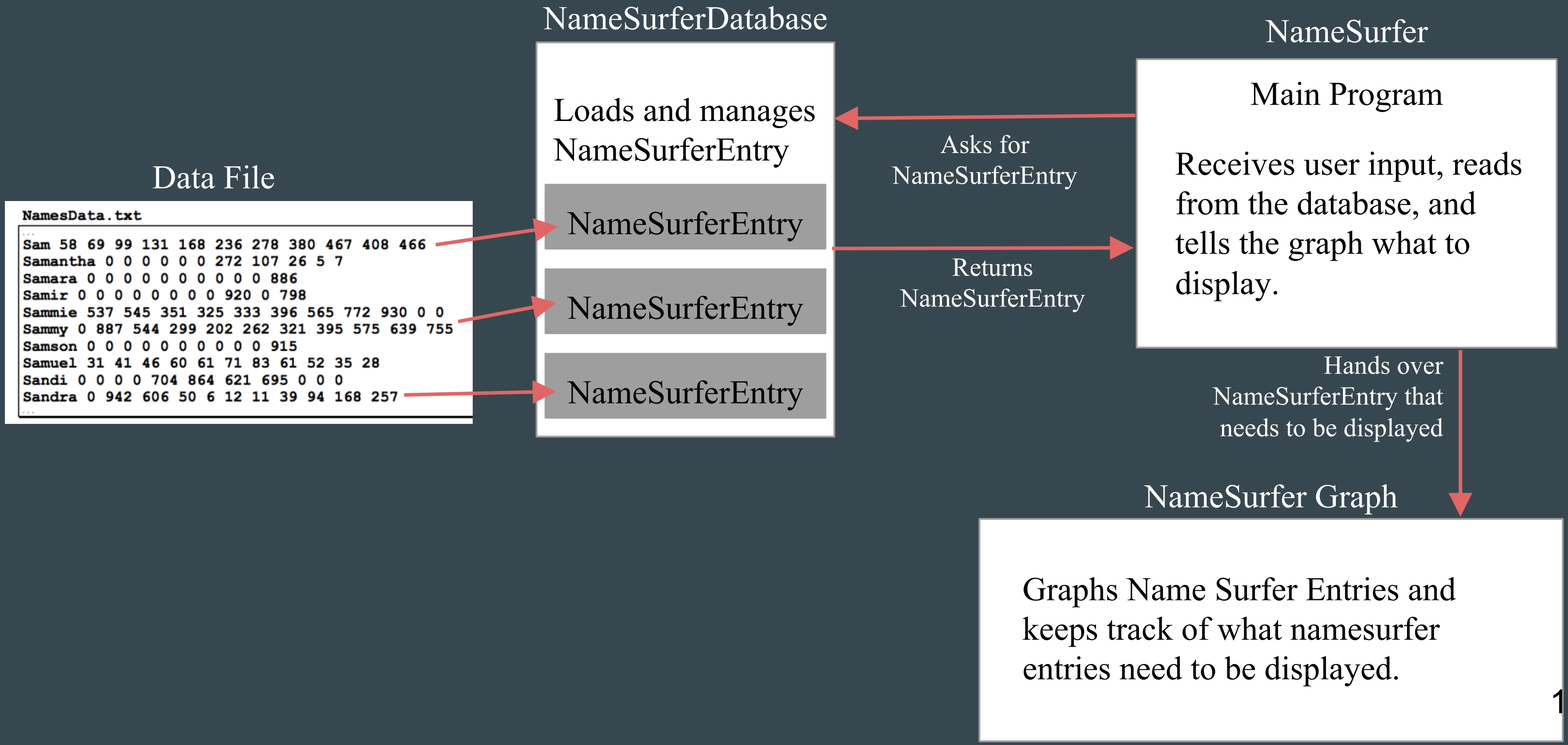
How does it work?

Specifics:

- Can graph many names,
  - each in different color
- Name not in top 1000 displayed with * at bottom of screen
- not case-sensitive
- If the name doesn't exist in the data file, don't do anything



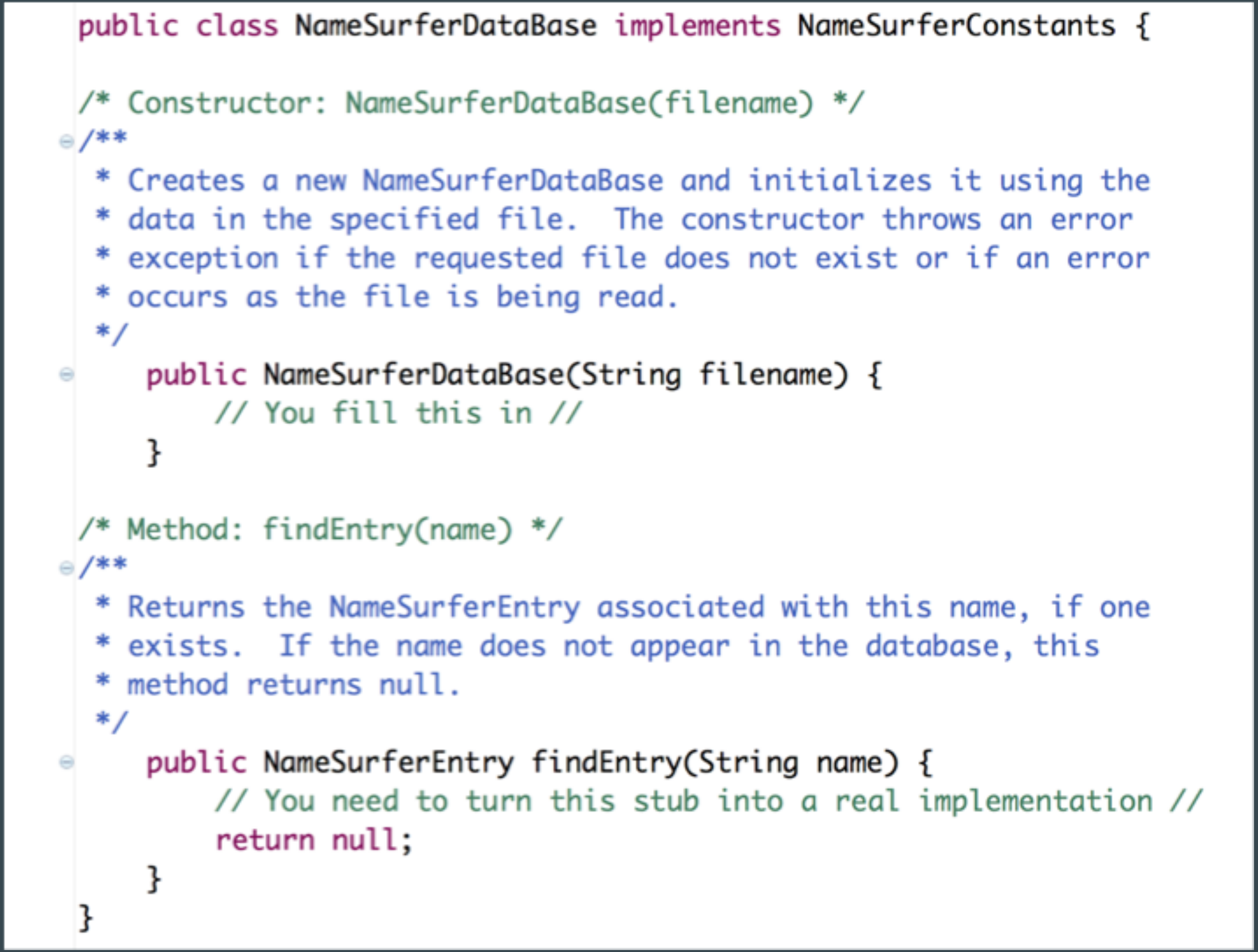
Overview of Assignment Structure


NameSurferDatabase
Loads and manages NameSurferEntry
NameSurferEntry
NameSurferEntry
NameSurferEntry
NameSurfer
Main Program
Receives user input, reads from the database, and tells the graph what to display.
Asks for NameSurferEntry
Returns NameSurferEntry
Data File
NamesData.txt
Hands over NameSurferEntry that needs to be displayed
NameSurfer Graph
Graphs Name Surfer Entries and keeps track of what namesurfer entries need to be displayed.

```
...
Sam 58 69 99 131 168 236 278 380 467 408 466
Samantha 0 0 0 0 0 0 272 107 26 5 7
Samara 0 0 0 0 0 0 0 0 0 0 886
Samir 0 0 0 0 0 0 0 0 920 0 798
Sammie 537 545 351 325 333 396 565 772 930 0 0
Sammy 0 887 544 299 202 262 321 395 575 639 755
Samson 0 0 0 0 0 0 0 0 0 0 915
Samuel 31 41 46 60 61 71 83 61 52 35 28
Sandi 0 0 0 0 704 864 621 695 0 0 0
Sandra 0 942 606 50 6 12 11 39 94 168 257
...
```

NameSurferDatabase

- Collection of NameSurferEntry objects
- Responsible for reading in text file and creating NameSurfer entry for each line in the text file
- Responsible for storing all entries, and being able to look up entries by String *name* (appropriate data structure? – array, ArrayList, HashMap?)

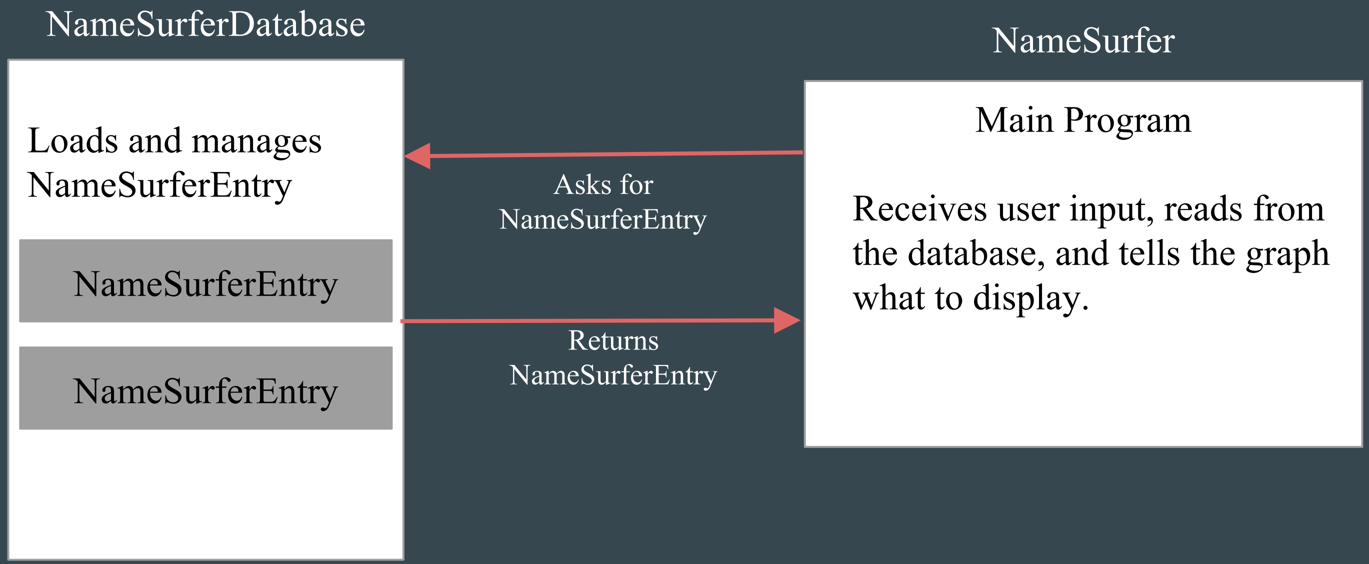


```
public class NameSurferDataBase implements NameSurferConstants {

/* Constructor: NameSurferDataBase(filename) */
/**
 * Creates a new NameSurferDataBase and initializes it using the
 * data in the specified file.  The constructor throws an error
 * exception if the requested file does not exist or if an error
 * occurs as the file is being read.
 */
    public NameSurferDataBase(String filename) {
        // You fill this in //
    }

/* Method: findEntry(name) */
/**
 * Returns the NameSurferEntry associated with this name, if one
 * exists.  If the name does not appear in the database, this
 * method returns null.
 */
    public NameSurferEntry findEntry(String name) {
        // You need to turn this stub into a real implementation //
        return null;
    }
}
```

Reading the Data File

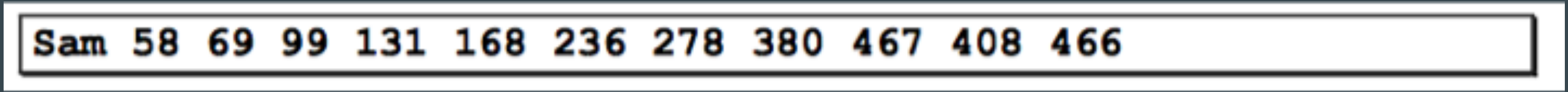
Use a BufferedReader (you already did it in Hangman!)

```
String line  = rd.readline();
NameSurferEntry entry = new NameSurferEntry(line);
// Somehow store the NameSurferEntry so it can be retrieved later
```

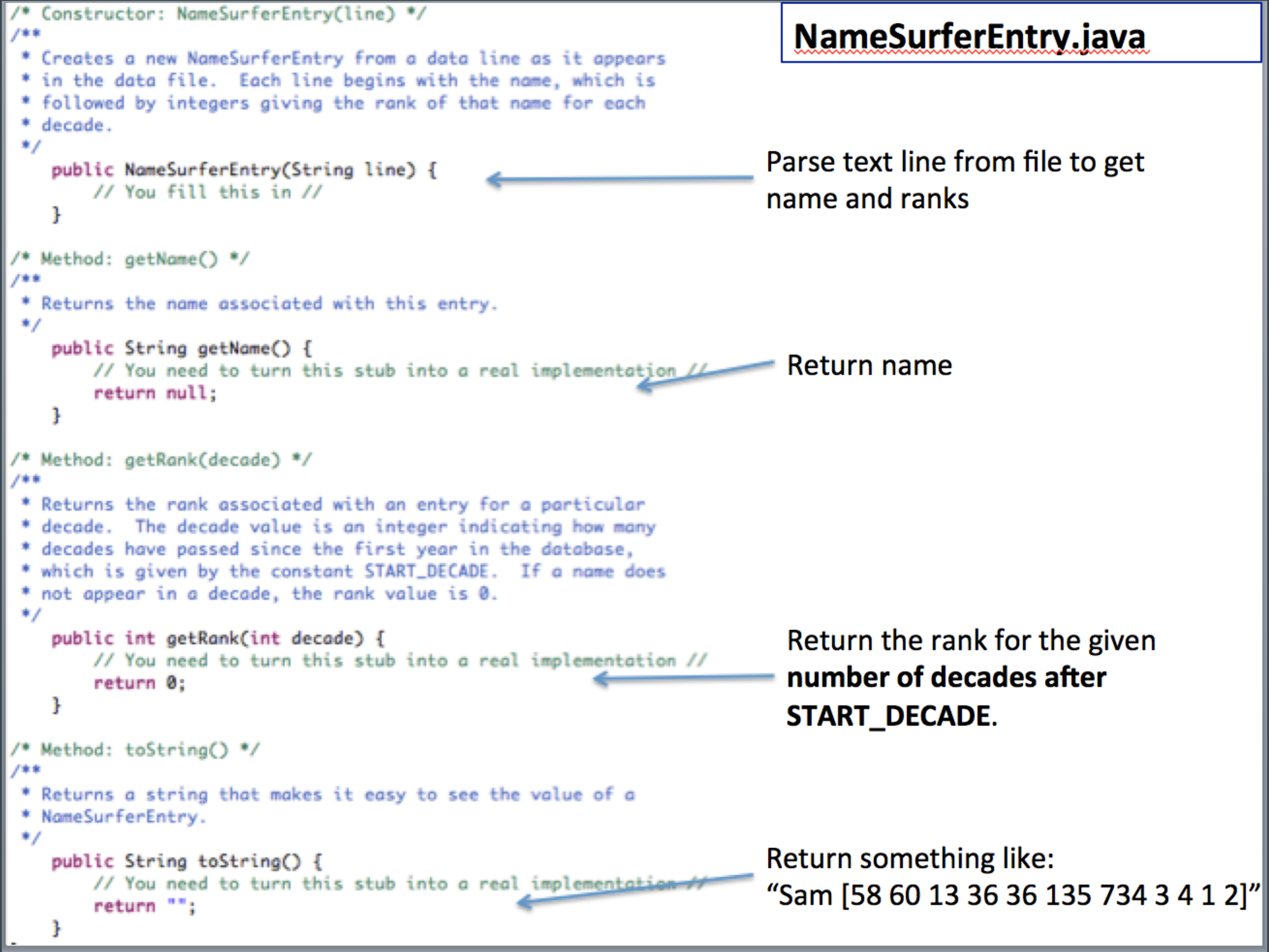

NameSurferDatabase
Loads and manages NameSurferEntry
NameSurferEntry
NameSurferEntry
Asks for NameSurferEntry
Returns NameSurferEntry
NameSurfer
Main Program
Receives user input, reads from the database, and tells the graph what to display.

NameSurferEntry

- Contains data for *one name/one line in text file*
- Stores name and popularity ranks for 1900-2000



```
Sam 58 69 99 131 168 236 278 380 467 408 466
```



NameSurferEntry.java

```
/* Constructor: NameSurferEntry(line) */
/**
 * Creates a new NameSurferEntry from a data line as it appears
 * in the data file.  Each line begins with the name, which is
 * followed by integers giving the rank of that name for each
 * decade.
 */
    public NameSurferEntry(String line) {
        // You fill this in //
    }

/* Method: getName() */
/**
 * Returns the name associated with this entry.
 */
    public String getName() {
        // You need to turn this stub into a real implementation //
        return null;
    }

/* Method: getRank(decade) */
/**
 * Returns the rank associated with an entry for a particular
 * decade.  The decade value is an integer indicating how many
 * decades have passed since the first year in the database,
 * which is given by the constant START_DECADE.  If a name does
 * not appear in a decade, the rank value is 0.
 */
    public int getRank(int decade) {
        // You need to turn this stub into a real implementation //
        return 0;
    }

/* Method: toString() */
/**
 * Returns a string that makes it easy to see the value of a
 * NameSurferEntry.
 */
    public String toString() {
        // You need to turn this stub into a real implementation //
        return "";
    }
```

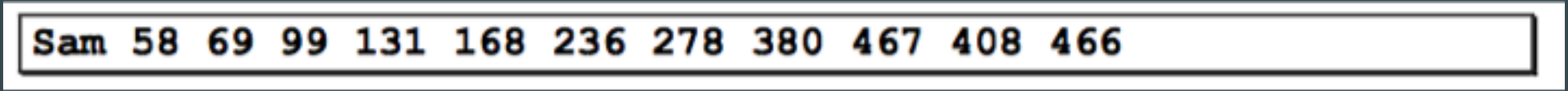
Parse text line from file to get name and ranks

Return name

Return the rank for the given **number of decades after START_DECADE.**

Return something like:
“Sam [58 60 13 36 36 135 734 3 4 1 2]”

Parsing Strategy - StringTokenizer!



```
Sam 58 69 99 131 168 236 278 380 467 408 466
```

```
StringTokenizer tokenizer = new  StringTokenizer(line);
while(tokenizer.hasMoreTokens()) {
    String token = tokenizer.nextToken();
    ...
// First time: token = “Sam”
// Second time: token = “58” (as a String!!)
// Third time: token = “69”, etc.
// Use Integer.parseInt(token) to convert from a string to an int
```

NameSurfer Graph

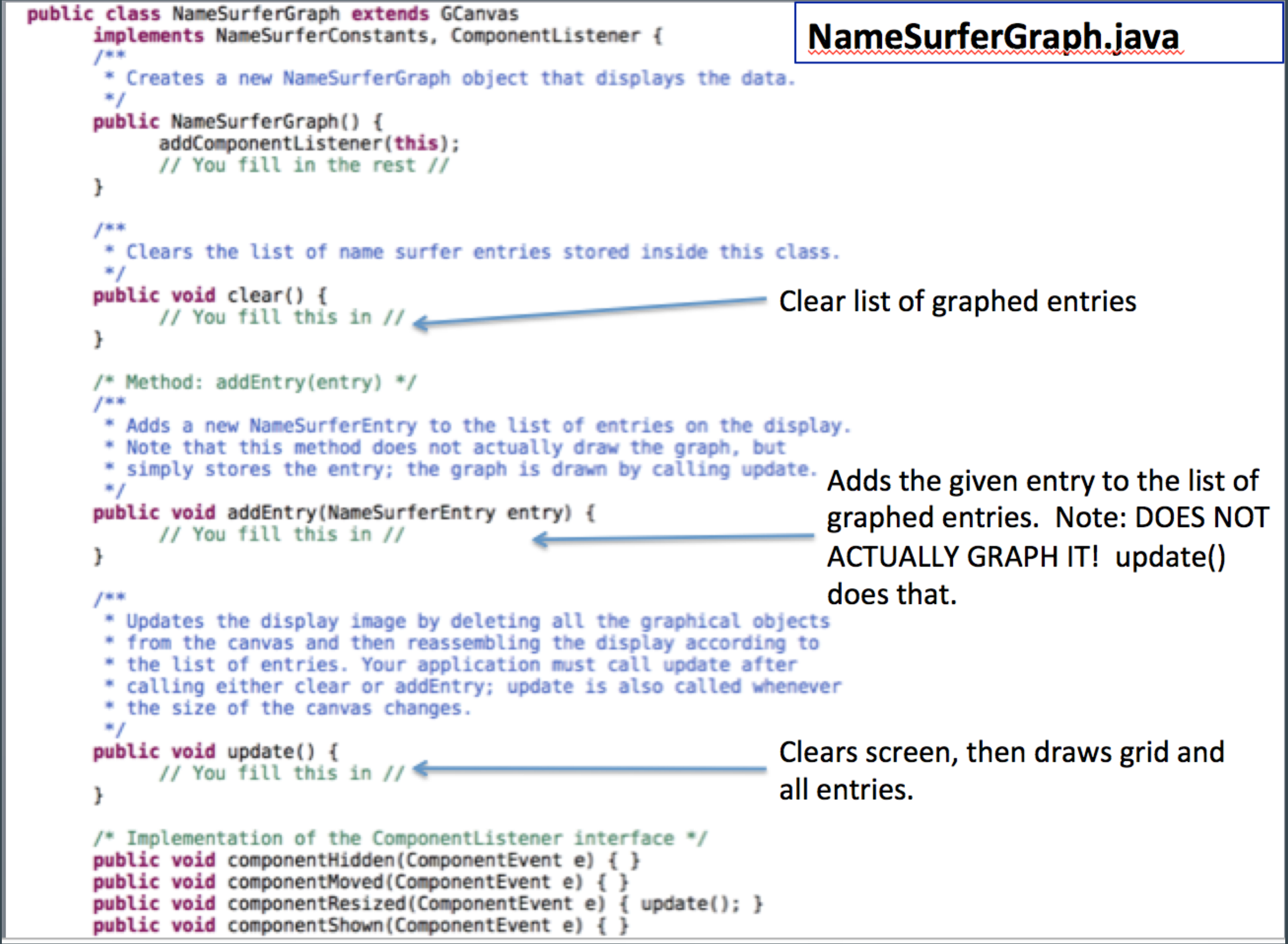
Similar purpose to HangmanCanvas class in last assignment

Handles setting up display and drawing names requested by the user

You should have all of the information you need to draw a name from the corresponding NameSurferEntry

- Resizes when window resizes! (automatic – update() is called whenever window resized)
- Stores all entries currently being graphed so graph can be redrawn when the window is resized
- Rank 0 -> use * instead of 0 in graph label. Rank 0 is at bottom of graph!!

NameSurferGraph.java



```
public class NameSurferGraph extends GCanvas
	implements NameSurferConstants, ComponentListener {
	/**
	 * Creates a new NameSurferGraph object that displays the data.
	 */
	public NameSurferGraph() {
		addComponentListener(this);
		// You fill in the rest //
	}

	/**
	 * Clears the list of name surfer entries stored inside this class.
	 */
	public void clear() {
		// You fill this in //
	}

	/* Method: addEntry(entry) */
	/**
	 * Adds a new NameSurferEntry to the list of entries on the display.
	 * Note that this method does not actually draw the graph, but
	 * simply stores the entry; the graph is drawn by calling update.
	 */
	public void addEntry(NameSurferEntry entry) {
		// You fill this in //
	}

	/**
	 * Updates the display image by deleting all the graphical objects
	 * from the canvas and then reassembling the display according to
	 * the list of entries. Your application must call update after
	 * calling either clear or addEntry; update is also called whenever
	 * the size of the canvas changes.
	 */
	public void update() {
		// You fill this in //
	}

	/* Implementation of the ComponentListener interface */
	public void componentHidden(ComponentEvent e) { }
	public void componentMoved(ComponentEvent e) { }
	public void componentResized(ComponentEvent e) { update(); }
	public void componentShown(ComponentEvent e) { }
```

Clear list of graphed entries

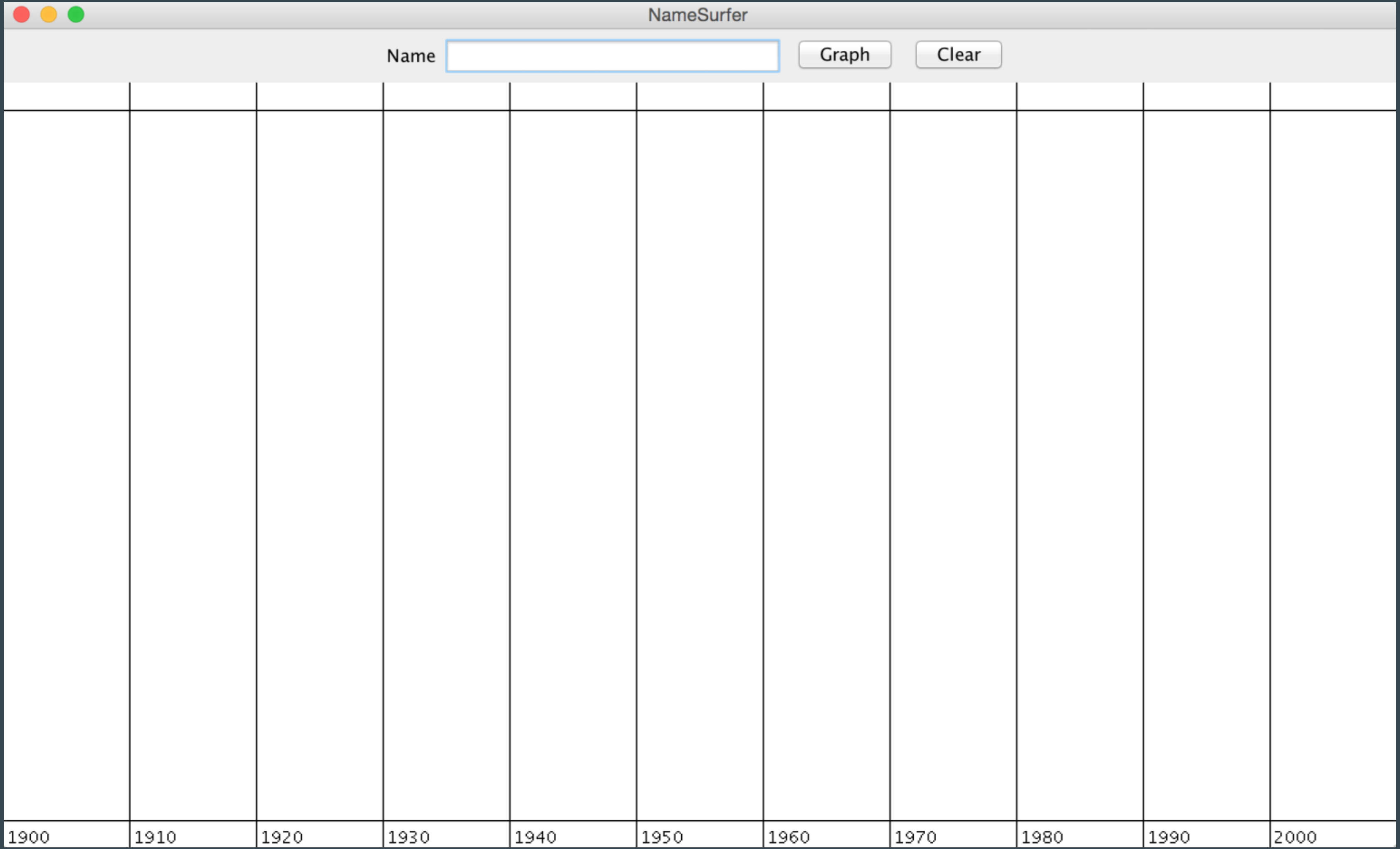
Adds the given entry to the list of graphed entries. Note: DOES NOT ACTUALLY GRAPH IT! update() does that.

Clears screen, then draws grid and all entries.

NameSurferGraph: Update()

Must also call update() when clearing or adding a new item. update() should be doing the drawing! (Why? We need to be able to reconstruct the entire graph) in NameSurfer.java (with graph as an instance variable):

```
graph = new NameSurferGraph(); // in init!
add(graph); // in init!
// later...
graph.add(entry); // graph entry!
graph.update(); // actually draws it!
```


NameSurfer
Name
Graph
Clear
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000

NameSurferGraph: Drawing

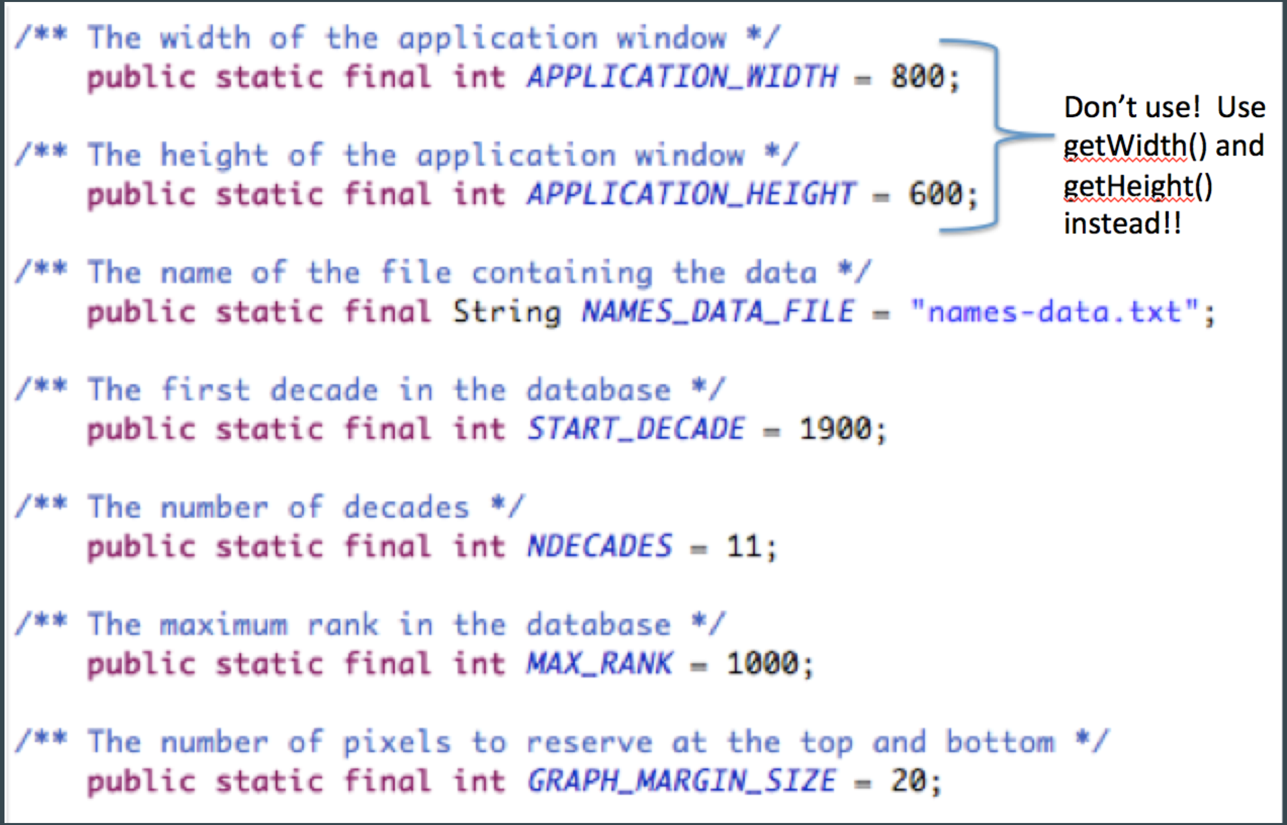
Draw lines + GLabels labeling each point

Remember, rank 0 should be graphed like MAX_RANK!

MAX_RANK drawn at bottom of graph, rank 1 drawn at top. How many rank spaces are in between?

All other ranks drawn, equally spaced between top and bottom.

Remember to use constants!



```
/** The width of the application window */
    public static final int APPLICATION_WIDTH = 800;

/** The height of the application window */
    public static final int APPLICATION_HEIGHT = 600;

/** The name of the file containing the data */
    public static final String NAMES_DATA_FILE = "names-data.txt";

/** The first decade in the database */
    public static final int START_DECADE = 1900;

/** The number of decades */
    public static final int NDECADES = 11;

/** The maximum rank in the database */
    public static final int MAX_RANK = 1000;

/** The number of pixels to reserve at the top and bottom */
    public static final int GRAPH_MARGIN_SIZE = 20;
```

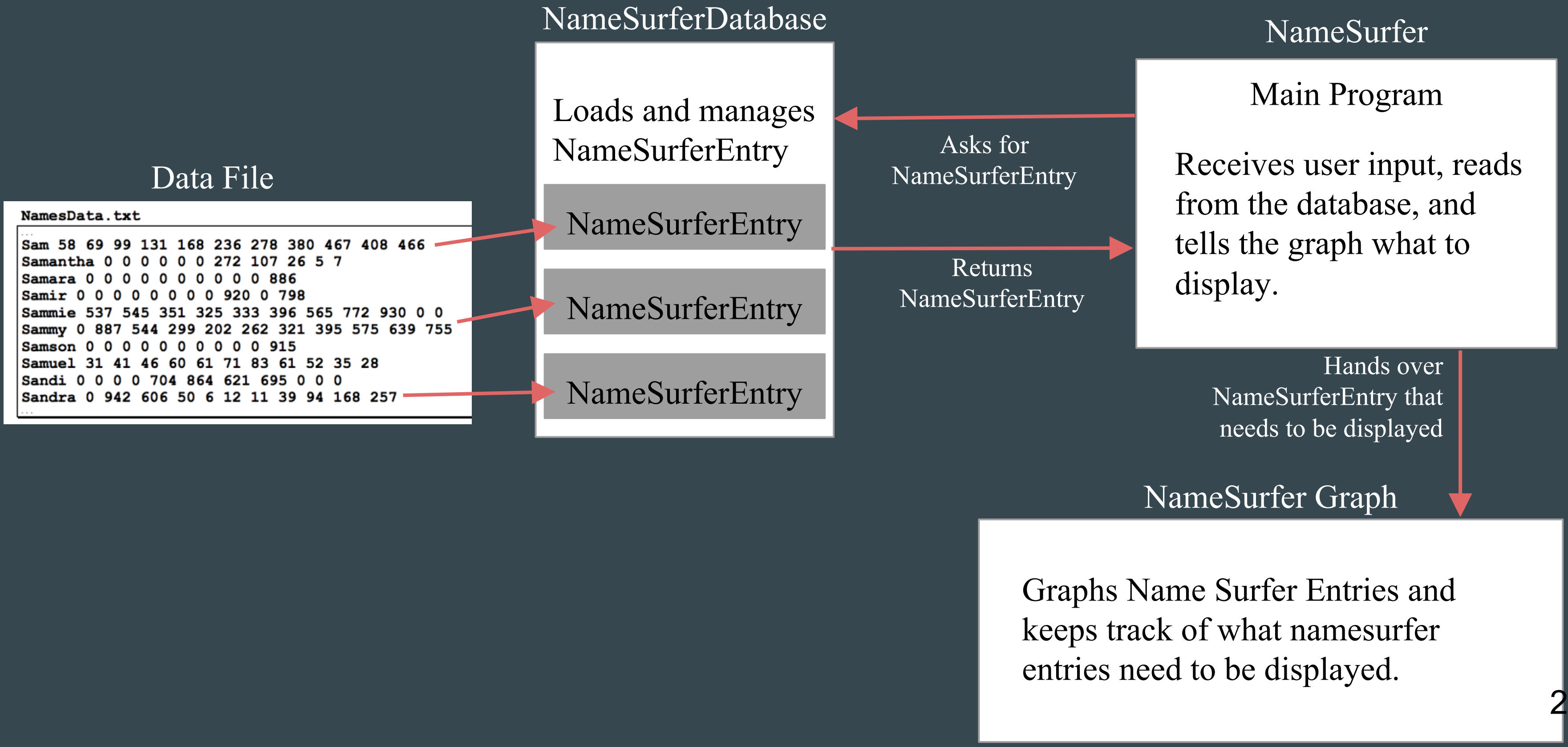
Don't use! Use getWidth() and getHeight() instead!!

NameSurfer

Main Program!

- Handles interactors and user input
- Based on user input, reads from database
  - Tells the graph what to draw
- Note: Name entered is not case sensitive

Overview of Assignment Structure



```
NamesData.txt
...
Sam 58 69 99 131 168 236 278 380 467 408 466
Samantha 0 0 0 0 0 0 272 107 26 5 7
Samara 0 0 0 0 0 0 0 0 0 0 886
Samir 0 0 0 0 0 0 0 0 920 0 798
Sammie 537 545 351 325 333 396 565 772 930 0 0
Sammy 0 887 544 299 202 262 321 395 575 639 755
Samson 0 0 0 0 0 0 0 0 0 0 915
Samuel 31 41 46 60 61 71 83 61 52 35 28
Sandi 0 0 0 0 704 864 621 695 0 0 0
Sandra 0 942 606 50 6 12 11 39 94 168 257
...
```

Loads and manages NameSurferEntry

Receives user input, reads from the database, and tells the graph what to display.

Graphs Name Surfer Entries and keeps track of what namesurfer entries need to be displayed.

Tricky Parts

- Null Pointer Exceptions - use the debugger to make sure you initialize objects
- Use milestones specified in the handout
- Do Extensions! Comment!

HAVE FUN