YEAH hours | **NameSurfer**

Interactors!

Make them, add them, use them

create & add them in the "`init`" method

call `addActionListeners()` to tell your program to listen for action events

example to create a JButton:

```
JButton graph = new JButton("Graph");
```

Once ActionListeners are activated, your program calls

```
public void actionPerformed(ActionEvent e) {...}
```

whenever an action event happens. You just have to change the code ***inside*** that method to make sure your program responds the way you want it to.

FOR EXAMPLE, if I had a console program and I wanted to print "cats" every time someone triggered an action event, this is what I'd do:

```
public void actionPerformed(ActionEvent e) {
    println("cats");
}
```

If an action event is triggered, it could have come from the Graph button, the Clear button, or the text field. How do I know where it came from?

Two ways:

- `e.getActionCommand()`
  - returns a string containing the name of the source
    - so, for example with our Graph JButton above, you would get "Graph" in response to e.getActionCommand() if "Graph" was pressed
- `e.getSource()`
  - returns a reference to the interactor that caused the event
    - so, if "graph" was an instance variable representing your graph button, you could see if e.getSource() == graph

Maps!

Like dictionaries: store **keys** and **values**

Constructor:

```
HashMap<String, String> map = new HashMap<String, String>();
```

Add something:

```
map.add("cats", "cats.jpg");
```

See if a map contains something:

```
boolean mapContainsDogs = map.containsKey("dogs");
```

Get a value from a map:

```
String catsImagefile = map.get("cats");
```

NameSurfer: Classes

NameSurfer
Puts everything together

NameSurferConstants
Get constants from here--nothing to implement

NameSurferEntry
Stores info for a name - the name's string and the values for all the years

NameSurferDataBase
Holds all the data
Reads in the data from a text file and stores it as a NameSurferEntry, locates the data associated with a name

NameSurferGraph
Subclass of GCanvas that displays the graph using GLines and GLabels

Milestones

Milestone 1: Assemble GUI Interactors
1. Set up a JTextField, Graph, and Clear buttons.
2. Temporarily change "Program" to "ConsoleProgram"
3. Test "Graph"'s output (for example, have the program print out the text from the textField every time graph is pushed) using the ConsoleProgram

Milestone 2: Implement the NameSurferEntry class
Object stores two things: name and a list of the 12 values indicating the ranks

Constructed with a line from the NamesData.txt file e.g.:

Sam 58 69 99 131 168 236 278 380 467 408 466 997

```
String line = rd.readLine();
NameSurferEntry entry = new NameSurferEntry(line);
```

Helpful hints:

- you can convert Strings to Integers using `Integer.parseInt(String s)`
- look at String.split or the StringTokenizer class to parse the strings

- Think about what instance variables you want to use in your NameSurferEntry object to store the name and ranks data to make getName() and getRank easy to implement

Milestone 3: Implement that NameSurferDatabase class

- Constructor: takes in the name of a datafile, reads in that datafile and stores it (somehow--which data structure would be the best?!)
    - remember how to read in a data file from Hangman?
- findEntry: take in a name, look it up in the database, return the NameSurferEntry object associated with it

Milestone 4: Create the Background Grid for the NameSurferGraph class

- Make NameSurferGraph extend Program again
- Make a NameSurferGraph object in your NameSurfer class

```
private NameSurferGraph;
graph = new NameSurferGraph();
add(graph);
```

- add the lines of the graph and the decade labels in the update method (don't forget proper decomp!)

Milestone 5: Complete the Implementation of NameSurferGraph

- Maintain a list of of values that are currently on display
    - `addEntry` adds a new NameSurferEntry to this list
    - `clear()` deletes all the entries
    - **Neither addEntry or clear actually changes the display!**
- the `update` method is the one that updates the display:
    - 1) delete all GObjects from the canvas
    - 2) draw everything again
- Make sure the lines on the graph cycle through different colors
- Mark rank 0 as a *