# `ArrayList`s

## An Interesting Article

“Big Data's Impact in the World”

http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html?src=me&ref=general&pagewanted=all

# Announcements

- *Hangman* due in one week (Wednesday, February 22).
- YEAH hours right after lecture today: 4:15 – 5:15PM in 370-370.

# Reading a File

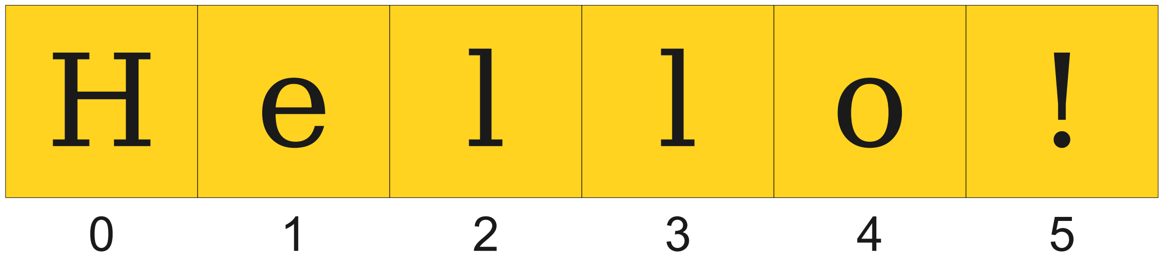
```
try {
     BufferedReader br = /* … open the file … */
     while (true) {
          String line = br.readLine();
          if (line == null) break;

          /* … process line … */
     }
     br.close();
} catch (IOException e) {
     /* … handle error … */
}
```

We can only remember one line of the file at a time!

# Remembering Lots of Data

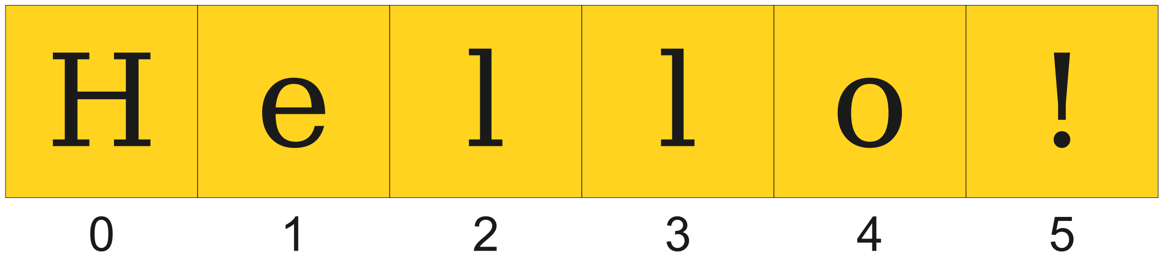
- Declare multiple variables.
  - Makes code really hard to read.
  - Have to know how much space in advance.
  - Can't treat variables uniformly.
- Store it in the canvas.
  - Only works for **`GObject`**s.
  - Can't easily retrieve them (**`getElementAt`** requires locations)
- Store it as a **`String`**.
  - Impractical for non-text information.

# Looking Closer at Strings



- A string stores a **sequence** of multiple characters.
    - Can access characters by index by calling **`charAt`**.
- Every character type is the same.
    - Namely, type **`char`**.

# Looking Closer at Strings



A string stores a **sequence** of multiple characters.

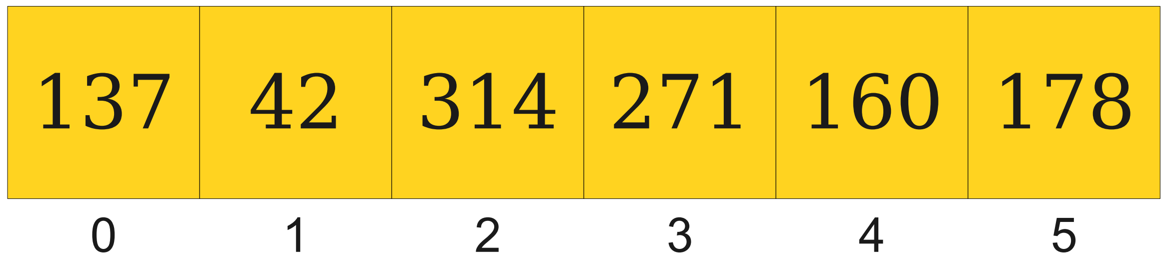
- Can access characters by index by calling **`charAt`**.

Every character type is the same.

- Namely, type **`char`**.

What if we don't want to store **`char`**s?

# Introducing **`ArrayList`**



- An **`ArrayList`** stores a **sequence** of multiple objects.
  - Can access objects by index by calling **`get`**.
- All stored objects have the same type.
  - You get to choose the type!

## **`String`**s and **`ArrayList`**s

- Both **`String`** and **`ArrayList`** store zero-indexed sequences.
  - **`String`**s store **`char`**s.
  - **`ArrayList`**s store objects.
- **`ArrayList`**s, unlike **`String`**s, are mutable.
  - You can insert, remove, and replace elements.

## Importing **`ArrayList`**

- To use **`ArrayList`**, you must

```
import java.util.*;
```

- Do **not** do the following!

```
import acmx.export.java.util.*;
```

# Simple **`ArrayList`** Operations

- You can append an element to an **`ArrayList`** by calling

```
arrayList.add(value)
```

- You can get the *n*th element of an **`ArrayList`** by calling

```
arrayList.get(n)
```

- You can see how many elements are in an **`ArrayList`** by calling

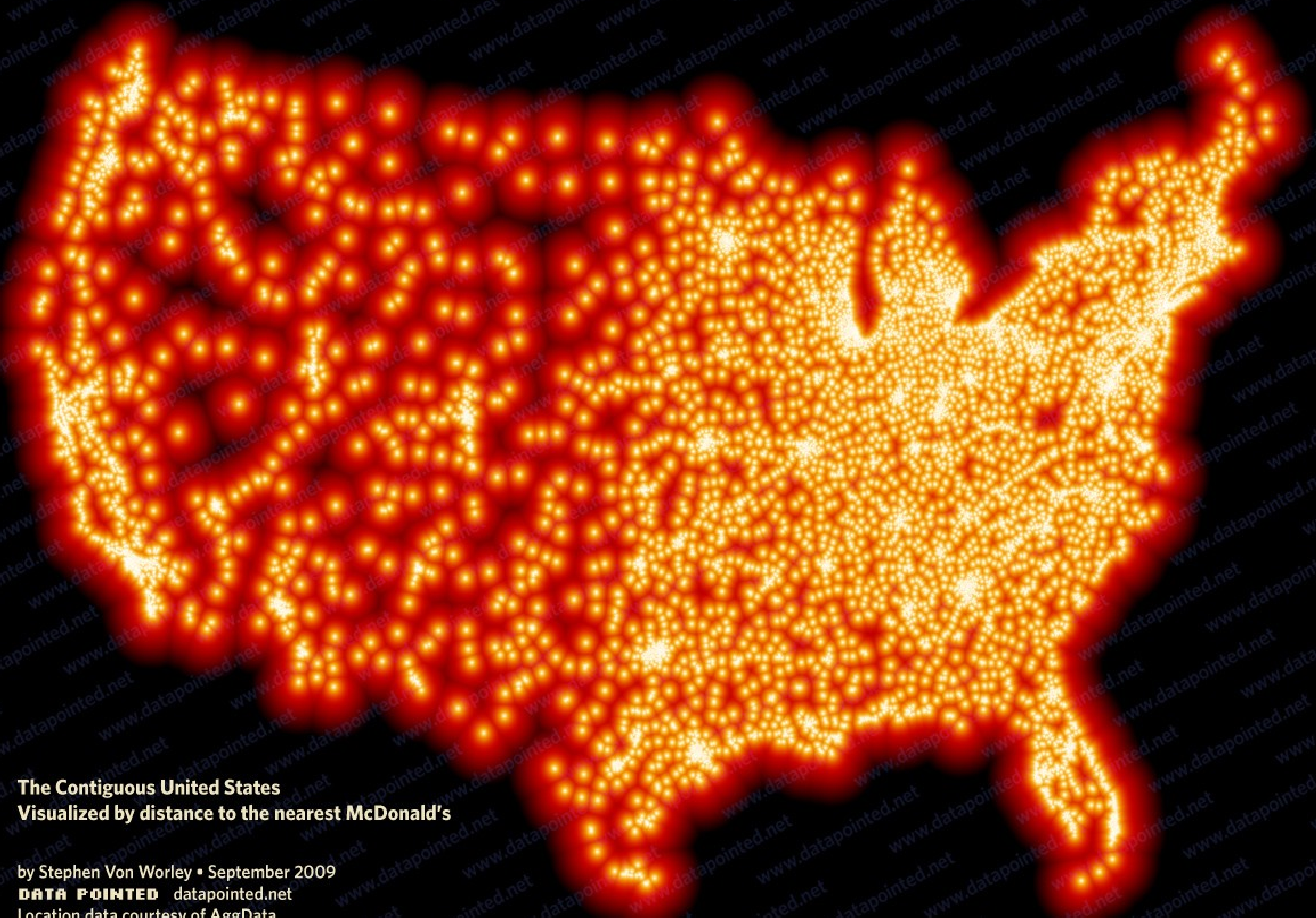
```
arrayList.size()
```

# Wrapper Types

- `ArrayList` cannot directly store primitive types.
- Java provides **wrapper types** that "wrap" a primitive type inside an object.

| | |
|---|---|
| `int` | `Integer` |
| `double` | `Double` |
| `char` | `Character` |
| `boolean` | `Boolean` |

## Putting it all Together



**The Contiguous United States**
**Visualized by distance to the nearest McDonald's**

by Stephen Von Worley • September 2009
**DATA POINTED** datapointed.net
Location data courtesy of AggData
http://www.aggdata.com/