ArrayList

Reading a File

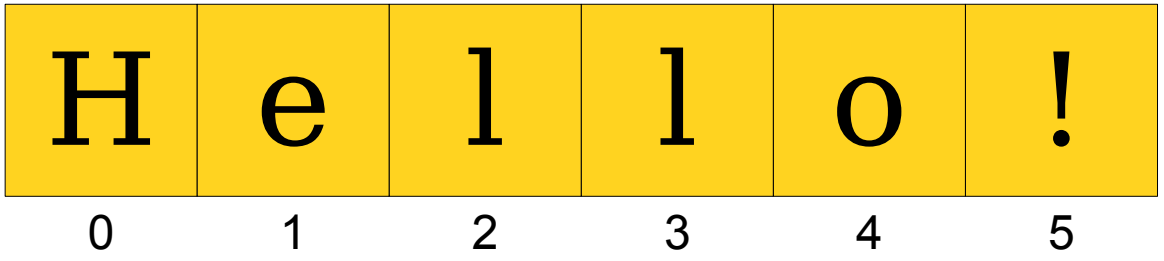
```
try {
    BufferedReader br = /* … open the file … */
    while (true) {
        String line = br.readLine();
        if (line == null) break;

        /* … 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

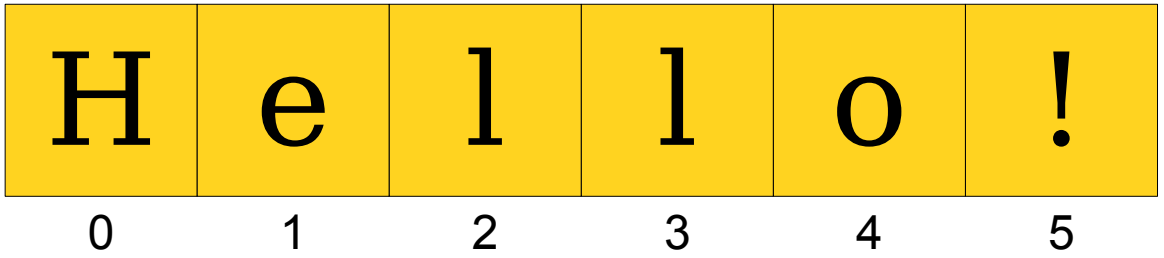
- Declare multiple variables.
  - Makes code really hard to read.
  - Have to know how much space in advance.
  - Can't treat variables uniformly.
- Store it in the canvas.
  - Only works for `GObjects`.
  - Can't easily retrieve them (`getElementAt` requires locations)
- Store it as a `String`.
  - 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 element has the same type.
  - Namely, type `char`.

Looking Closer at Strings



A string stores a sequence of multiple characters.

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

Every element has the same type.

- Namely, type `char`.

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

Introducing `ArrayList`

137	42	314	271	160	178
0	1	2	3	4	5

- 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!

`Strings` and `ArrayLists`

- Both `String` and `ArrayList` store zero-indexed sequences.
  - `Strings` store `chars`.
  - `ArrayLists` store objects.
- `ArrayLists`, unlike `Strings`, are mutable.
  - You can insert, remove, and replace elements.

Importing `ArrayList`

- To use `ArrayList`, you need to import it:

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

- ***Don't*** import 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()`**

Time-Out for Announcements!

Midterm Locations

- Midterm is ***tomorrow***, February 10 from 7PM – 10PM.
- Locations divvied up by last name:
  - `Abb - Jon:` Go to **Hewlett 200**
  - `Jun - Mari:` Go to **Hewlett 201**
  - `Marq - Mik:` Go to **Hewlett 101**
  - `Mil - Ogr:` Go to **Hewlett 102**
  - `Oke - Pat:` Go to **Hewlett 103**
  - `Pau - Tan:` Go to **Braun Auditorium**
  - `Tao - Zuc:` Go to **320-105**
- Remember to prepare your one sheet of notes!

Assignments

- Assignment 4 was due at 3:15PM today.
  - Due Wednesday with one late period or Friday with two.
- Assignment 5 (**The Java String Quartet**) out now, due next Wednesday, February 18 at 3:15PM.
  - Play around with strings, file processing, and `ArrayLists`!
  - ***Recommendation:*** Study for the midterm! Read through the assignment handout by Wednesday.

Back to CS106A!

The Range-Based `for` Loop

- You can iterate over the elements of an `ArrayList`, in order, using this syntax:

```
for (type var : list) {
    /* … process var … */
}
```

- Useful when you need to visit everything in a list in order and don't need access to the indices.

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/