Building Java Programs

ArrayList

Reading: 10.1

Easier than Arrays

- Java's arrays are really useful for many programming tasks
  - And they can hold elements of any type
- But you must choose the size when you make the array
  - Cannot "grow or shrink" as the program runs
  - Could make a new array
  - Or use the ArrayList library, which does this for you
    - Complicated internal behavior, simple external
- The first library we have seen that "works for any type"
  - But you must specify the type with some special syntax

ArrayList basics

```
import java.util.*;

ArrayList<type> name = new ArrayList<type>();
```

- "Angle-bracket syntax" the new thing: type of elements
- Initially holds 0 elements
- () because we're calling a 0-argument constructor

- `new` returns a new `ArrayList` object, of course.
  - Use its methods to access and mutate the contents…

ArrayList methods

The primary methods for manipulating an `ArrayList<E>`:

- `add(E value)` appends value at end of list
- `add(int index, E value)` inserts given value at given index, shifting subsequent values right
- `clear()` removes all elements of the list
- `get(int index)` returns the value at given index
- `remove(int index)` removes and returns value at given index, shifting subsequent values left
- `set(int index, E value)` replaces value at given index with given value
- `size()` returns the number of elements in list

Notice types for `add` and `set` depend on type of ArrayList

- Won't learn in 142 how to define our own classes that do this
- But we can use ArrayList, written by others

Mini-Exercise

- Make a new ArrayList named `creatures` and add the strings `"octopus"` and `"squid"` to it.
- Cheat sheet:

```
ArrayList<type> name = new ArrayList<type>();
```

The primary methods for manipulating an `ArrayList<E>`:

- `add(E value)` appends value at end of list
- `add(int index, E value)` inserts given value at given index, shifting subsequent values right
- `clear()` removes all elements of the list
- `get(int index)` returns the value at given index
- `remove(int index)` removes and returns value at given index, shifting subsequent values left
- `set(int index, E value)` replaces value at given index with given value
- `size()` returns the number of elements in list

Mini-Exercise -solution

- Make a new ArrayList named `creatures` and add the strings `"octopus"` and `"squid"` to it.

```
ArrayList<String> creatures = new ArrayList<String>();
creatures.add("octopus");
creatures.add("squid");
```

An annoying limitation

- Can make an ArrayList for any type ***of object*** you want
  - But `int`, `double`, `boolean`, and `char` are not types *of objects*
  - (`String` is)

- You can make:

```
ArrayList<Integer>   ArrayList<Boolean>
ArrayList<Double>    ArrayList<Character>
```

- Java will automatically convert between numbers/booleans and objects of these *wrapper classes* for you
  - A convenient, but fairly confusing and new feature to Java

- For `ArrayList`, all you need to know is that you have to write `Integer`, etc. in the angle brackets, not `int`, etc.

Practice with ArrayList

- (See separate handout for several "exam-style" ArrayList questions that show how an ArrayList can be used)

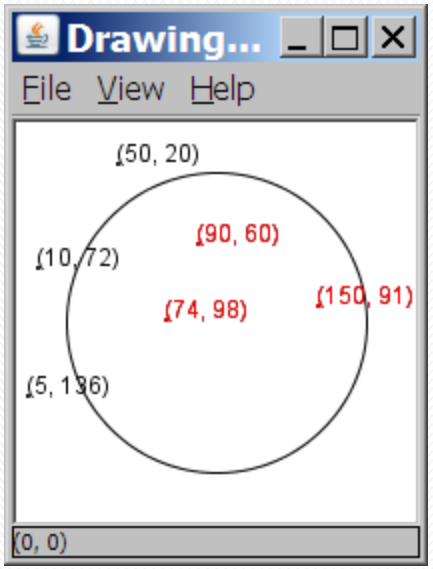
- Particularly notice what is easier than with arrays:
  - add and remove "in the middle"
  - having the size change

- Exercise on your own: Rewrite one or both of our sorting algorithms using ArrayList

Recall: earthquake problem

- Given a file of cities' (x, y) coordinates, which begins with the number of cities:

```
6
50 20
90 60
10 72
74 98
5 136
150 91
```



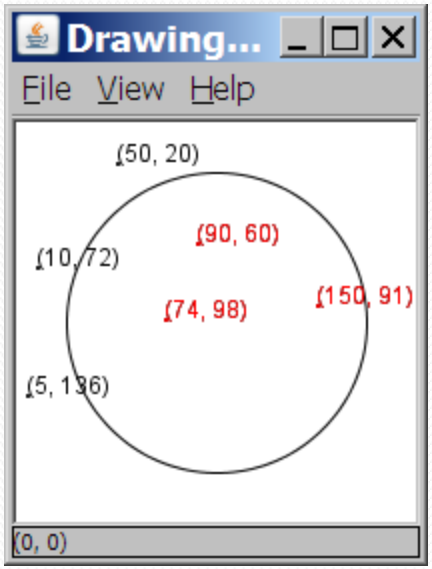
- Write a program to draw the cities on a `DrawingPanel`, then color the cities red that are within the radius of effect of the earthquake:

```
Epicenter x/y? 100 100
Radius of effect? 75
```

We made it a little too easy

- Given a file of cities' (x, y) coordinates, which begins with the number of cities:

```
6
50 20
90 60
10 72
74 98
5 136
150 91
```



- What if we didn't know the number of cities in advance?
  - Yes, "use a while loop", but first we have to make the array!

Solutions

1. Read the file twice
   - First time just to count how many cities
   - Okay, but won't work if the problem reads cities from the console until the user types, "STOP"
2. Resize the array as necessary
   - A common idiom worth knowing
   - Doubling each time is a very elegant policy
     - Never waste more than half the space
     - And you won't make very many arrays (2^{25} > 32 million)
3. Use an ArrayList
   - Just call `add` each time
   - Probably the ArrayList library is using something like resizing, *but that's somebody else's problem*
     - *Don't reinvent the wheel*