

Building Java Programs

Chapter 10

Lecture 10-1: `ArrayList`

reading: 10.1

Welcome to CSE 143!

I'm H el ene Martin

<http://cs.washington.edu/143>

CSE 143

- Goal: learn tools for automating complex tasks efficiently
 - Abstraction (client vs. implementation)
 - Data structures
 - Algorithms
- Prerequisite: can automate basic tasks using a programming language (logic, control flow, decomposition)
- For EVERYONE, not just CSE majors
- Learn by doing
- Lots of support (undergraduate TAs, IPL, message board)

Programming

- CS: "efficiently implementing automated abstractions" ¹
- Building things is empowering
 - Small number of fundamentals can solve lots of problems
 - When a program works, it's obvious
 - Welding, chain saws, safety glasses not required
- A LOT of complexity to master: exciting and scary
- Java is our tool in 14x but lessons transfer broadly

Being Successful

- Determination, hard work, focus
- Investing time (~15 hours a week)
 - Starting early
 - Developing problem-solving strategies
 - Developing a consistent style
- Knowing when to ask for help
 - Go to the IPL
 - Talk to me after class, during office hours
- Studying together
 - Homework is individual but studying in groups pays off

Logistics

- Get to know <http://cs.washington.edu/143>
- 2 sections a week
 - Turn in ONE set of problems each week for credit
- Grading described on syllabus
 - 50% projects, 20% midterm, 30% final (exams open book)
- Weekly programming projects
 - Academic honesty is serious
 - 40 point scale
 - 5 "free late days"; -2 for subsequent days late

Words exercise

- Write code to read a file and display its words in reverse order.
- A solution that uses an array:

```
String[] allWords = new String[1000];
```

```
int wordCount = 0;
```

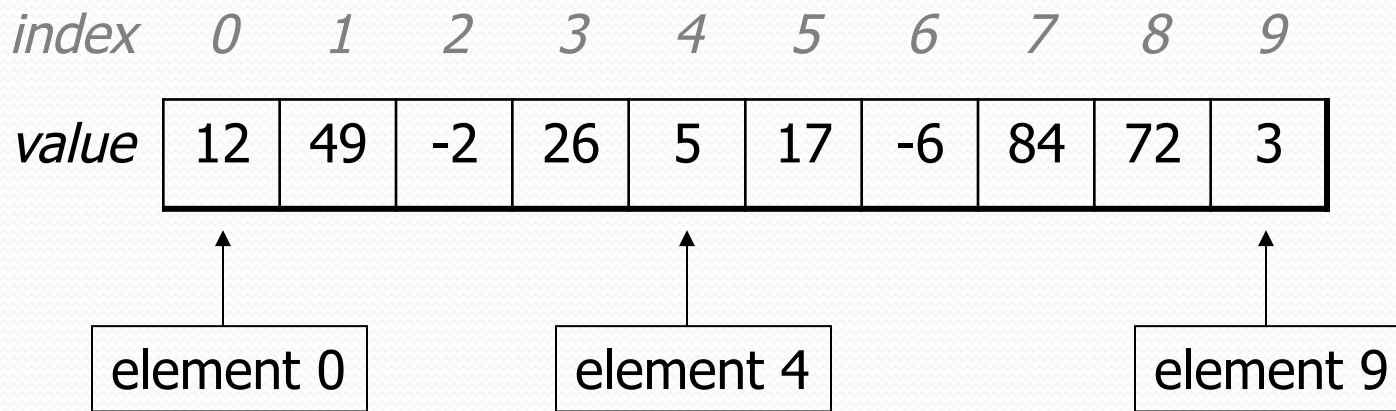
```
Scanner input = new Scanner(new File("words.txt"));
```

```
while (input.hasNext()) {  
    String word = input.next();  
    allWords[wordCount] = word;  
    wordCount++;  
}
```

- What's wrong with this?

Recall: Arrays (7.1)

- **array:** object that stores many values of the same type.
 - **element:** One value in an array.
 - **index:** 0-based integer to access an element from an array.
 - **length:** Number of elements in the array.



length = 10

Array Limitations

- Fixed-size
- Adding or removing from middle is hard
- Not much built-in functionality (need Arrays class)

List Abstraction

- Like an array that resizes to fit its contents.
- When a list is created, it is initially empty.

```
[]
```

- Use `add` methods to add to different locations in list

```
[hello, ABC, goodbye, okay]
```

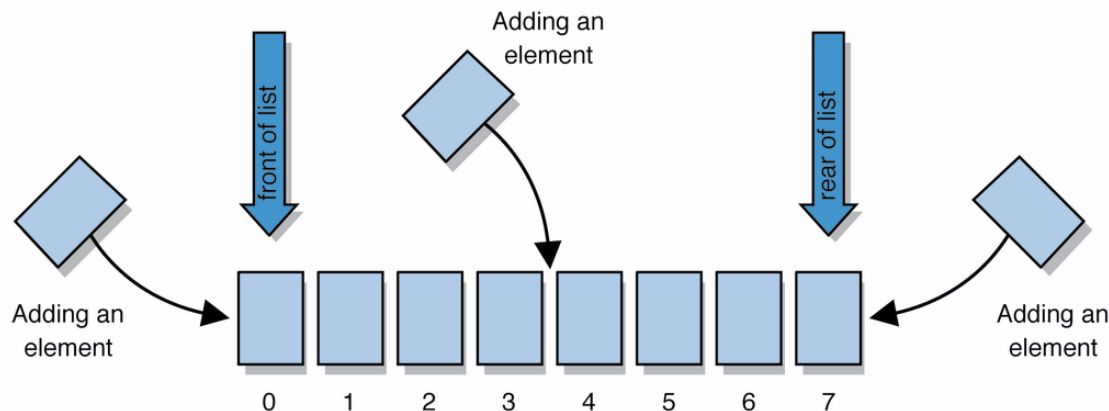
- The list object keeps track of the element values that have been added to it, their order, indexes, and its total size.
- You can add, remove, get, set, ... any index at any time.

Collections and lists

- **collection**: an object that stores data ("**elements**")

```
import java.util.*; // to use Java's collections
```

- **list**: a collection of elements with 0-based **indexes**
 - elements can be added to the front, back, or elsewhere
 - a list has a **size** (number of elements that have been added)
 - in Java, a list can be represented as an **ArrayList** object



Type parameters (generics)

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

- When constructing an `ArrayList`, you must specify the type of its elements in `< >`
 - This is called a *type parameter*; `ArrayList` is a *generic* class.
 - Allows the `ArrayList` class to store lists of different types.
 - Arrays use a similar idea with **Type** []

```
ArrayList<String> names = new ArrayList<String>();  
names.add("Marty Stepp");  
names.add("Stuart Reges");
```

ArrayList methods (10.1)*

<code>add (value)</code>	appends value at end of list
<code>add (index, value)</code>	inserts given value just before the given index, shifting subsequent values to the right
<code>clear ()</code>	removes all elements of the list
<code>indexOf (value)</code>	returns first index where given value is found in list (-1 if not found)
<code>get (index)</code>	returns the value at given index
<code>remove (index)</code>	removes/returns value at given index, shifting subsequent values to the left
<code>set (index, value)</code>	replaces value at given index with given value
<code>size ()</code>	returns the number of elements in list
<code>toString ()</code>	returns a string representation of the list such as "[3, 42, -7, 15]"

* (a partial list; see 10.1 for other methods)

ArrayList vs. array

```
String[] names = new String[5];           // construct
names[0] = "Jessica";                    // store
String s = names[0];                      // retrieve
for (int i = 0; i < names.length; i++) {
    if (names[i].startsWith("B")) { ... }
}
```

```
ArrayList<String> list = new ArrayList<String>();
list.add("Jessica");                       // store
String s = list.get(0);                     // retrieve
for (int i = 0; i < list.size(); i++) {
    if (list.get(i).startsWith("B")) { ... }
}
```

ArrayList as param/return

```
public static void name(ArrayList<Type> name) { // param
public static ArrayList<Type> name(params) // return
```

- Example:

```
// Returns count of plural words in the given list.
public static int countPlural(ArrayList<String> list) {
    int count = 0;
    for (int i = 0; i < list.size(); i++) {
        String str = list.get(i);
        if (str.endsWith("s")) {
            count++;
        }
    }
    return count;
}
```

Words exercise, revisited

- Write a program that reads a file and displays the words of that file as a list.
 - Then display the words in reverse order.
 - Then display them with all plurals (ending in "s") capitalized.
 - Then display them with all plural words removed.

Exercise solution (partial)

```
ArrayList<String> allWords = new ArrayList<String>();
Scanner input = new Scanner(new File("words.txt"));
while (input.hasNext()) {
    String word = input.next();
    allWords.add(word);
}

// display in reverse order
for (int i = allWords.size() - 1; i >= 0; i--) {
    System.out.println(allWords.get(i));
}

// remove all plural words
for (int i = 0; i < allWords.size(); i++) {
    String word = allWords.get(i);
    if (word.endsWith("s")) {
        allWords.remove(i);
        i--;
    }
}
}
```