Unit 7: ArrayList
Introduction to ArrayLists

Adapted from:

1) Building Java Programs: A Back to Basics Approach
by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum

https://longbaonguyen.github.io

Problem with Arrays

Ask the user to enter a list of words, save the words in an array.

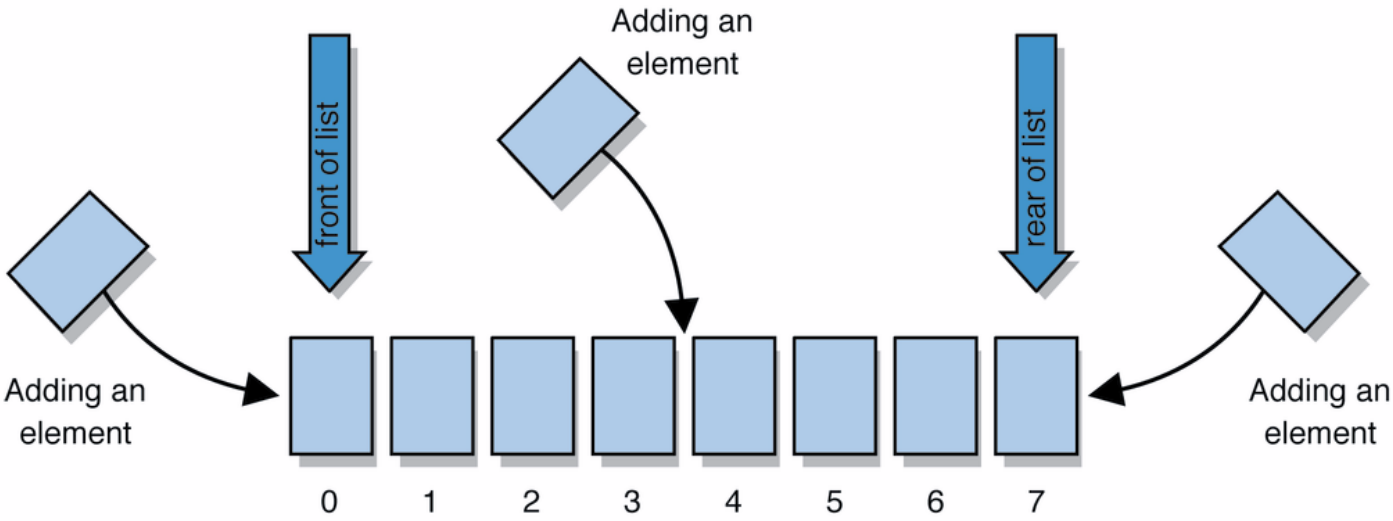
```
String[] allWords = new String[1000];
int wordCount = 0;
```

- Problem: You don't know how many words the user will enter.
  - Hard to create an array of the appropriate size.

- Luckily, there are other ways to store data besides in an array.

Lists

- **list**: a collection storing an ordered sequence of elements
  - each element is accessible by a 0-based **index**
  - a list has a **size** (number of elements that have been added)
  - elements can be added to the front, back, or elsewhere
  - in Java, a list can be represented as an **`ArrayList`** object



Idea of a list

- Rather than creating an array of boxes, create an object that represents a "list" of items. (initially an empty list.)

```
[]
```

- You can add items to the list.
  - The default behavior is to add to the end of the 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.
  - Think of an "array list" as an automatically resizing array object.
  - Internally, the list is implemented using an array and a size field.

`ArrayList` methods (10.1)

`boolean add(E obj)`	appends obj at end of list, returns true
`void add(int index, E obj)`	inserts given obj just before the given index, shifting subsequent values to the right
`E get(int index)`	returns the element at given index
`E remove(int index)`	removes/returns value at given index, shifting subsequent values to the left
`void remove(E obj)`	remove by object(Not on AP, but useful for some of our labs.
`E set(int index, E obj)`	replaces value at given index with given value, returns element that was previously at index.
`int size()`	returns the number of elements in list
`toString()`	returns a string representation of the list such as `"[3, 42, -7, 15]"`

Note: E refers to the type of objects in the arraylist.

ArrayList class is part of the java.util package.
To use ArrayList: import java.util.*;

Type Parameters (Generics)

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

- When constructing an `ArrayList`, you must specify the type of elements it will contain between < and >.
  - This is called a *type parameter* or a *generic* class.
  - Allows the same `ArrayList` class to store lists of different types.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Marty Stepp");
names.add("Stuart Reges");
//notice the difference between arraylist & array
String[] names= new String[10];
```

ArrayList vs. array

- construction

```
String[] names = new String[5];
ArrayList<String> list = new ArrayList<String>();
```

- storing a value

```
names[0] = "Jessica";
list.add("Jessica");
```

- retrieving a value

```
String s = names[0];
String s = list.get(0);
```

ArrayList vs. array

```
ArrayList<String> list = new ArrayList<String>();

list.add("Michael");
list.add("Jessica");
list.add("Lee"); //{"Michael", "Jessica", "Lee"}
list.add(1,"Sarah");//{"Michael", "Sarah","Jessica", "Lee"}
String store=list.set(2,"Mary")
//{Michael,Sarah,Mary,Lee}, store="Jessica"

String store2=list.get(3);//store2="Lee"

String store3=list.remove(1);
//{Michael,Mary,Lee}, store3="Sarah"
```

Traversing an Arraylist

It is common to use `for` loops to access arraylist elements. This is called **traversing the arraylist.**

The `list` arrayList below is an arrayList of Strings from the previous slides.

```
for (int i = 0; i < list.size(); i++) {
    System.out.println(list.get(i));
}
```

Contrast this with an array also called list.

```
for (int i = 0; i < list.length; i++) {
    System.out.println(list[i]);
}
```

Traversing an Arraylist

We can use a for each loop traverse the arraylist. The `list` arrayList below is an arrayList of Strings from the previous slides.

```
for (String name: list) {
    System.out.println(name);
}
```

Note this syntax is the same for both arrays and arraylists!

Do not use the enhanced for each loop if you want to add or remove elements when traversing a list because it will throw a **ConcurrentModificationException** error. Since for each loops do not use an index, you cannot do this special case of incrementing only if it is changed.

So if you are going to add or remove items or you need the index, use a regular for loop or a while loop.

Remove Plurals

Removing an element from an arraylist requires care as elements are shifted left.

```
// Suppose allWords is an arraylist of words entered by
// user. Remove all plural words from allWords.

for (int i = 0; i < allWords.size(); i++) {
    String word = allWords.get(i);
    if (word.substring(word.length() - 1).equals("s")) {
        allWords.remove(i);

    }
}
```

This is an error! Once a word is removed, words are shifted to the left, this will cause the next word to be skipped!

Remove Plurals

One way to fix the previous error is add a i-- correction.

```
// Suppose allWords is an arraylist of words entered by
// user. Remove all plural words from allWords.

for (int i = 0; i < allWords.size(); i++) {
    String word = allWords.get(i);
    if (word.substring(word.length - 1).equals("s")) {
        allWords.remove(i);
        i--; //corrected
    }
}
```

Can you think of another way of fixing this error?

Remove Plurals

Or alternatively, we can traverse the array backwards; this eliminates the need to do i--. Do you see why the following works?

```
// Suppose allWords is an arraylist of words entered by
// user. Remove all plural words from allWords.

for (int i = allWords.size() - 1; i >= 0; i--) {
    String word = allWords.get(i);
    if (word.substring(word.length - 1).equals("s")) {
        allWords.remove(i);
        // i-- not needed!
    }
}
```

ArrayList as parameter

A method can accept an arraylist as a parameter.

```
public static void name(ArrayList<Type> name) {
```

Write a method that count all of the letters in all of the Strings from an ArrayList of Strings.

```
public static int countLetters(ArrayList<String> list) {
    int count = 0;
    for (int i = 0; i < list.size(); i++) {
        count += list.get(i).length();
    }
    return count;
}
```

Returns an ArrayList

You can also return a list:

```
public static ArrayList<Type> methodName(params)
```

Write a method that accepts an array of String and return an arraylist copy of the array.

```
public static ArrayList<String> copy(String[] array)
{
   ArrayList<String> result = new ArrayList<String>();

   for (int i = 0; i < array.length; i++) {
            result.add(array[i]);
      }
   return result;
}
```

`ArrayList` of primitives?

- The type you specify when creating an `ArrayList` must be an object type; it cannot be a primitive type.

```
// illegal -- int cannot be a type parameter
ArrayList<int> list = new ArrayList<int>();
```

- But we can still use `ArrayList` with primitive types by using *wrapper* classes in their place.

```
// creates a list of ints
ArrayList<Integer> list = new ArrayList<Integer>();
```

- Once you construct the list, use it with primitives as normal:

```
ArrayList<Double> grades = new ArrayList<Double>();
grades.add(3.2); //autoboxing
grades.add(2.7);
double myGrade = grades.get(0);//auto-unboxing
```

ArrayList of Integers

```
ArrayList<Integer> myList = new ArrayList<Integer>();
myList.add(50);
myList.add(30);
myList.add(20);

int total = 0;
for (int i = 0; i < myList.size(); i++){
  total += myList.get(i);
}
System.out.println(total);
```

Output: 100

ArrayList "mystery"

```
ArrayList<Integer> list = new ArrayList<Integer>();

for (int i = 1; i <= 10; i++) {
    list.add(10 * i);
}
```

What is in the arraylist list?

Answer: [10, 20, 30, 40, ..., 100]

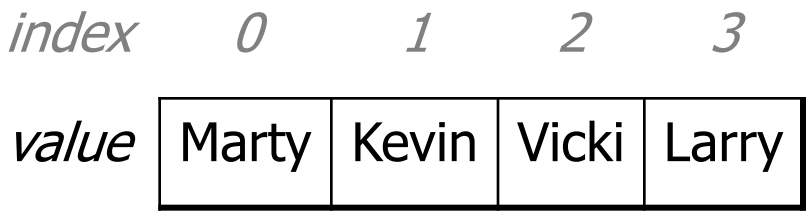
```
for (int i = 0; i < list.size(); i++) {
    list.remove(i);
}
System.out.println(list);
```

Answer: [20, 40, 60, 80, 100]

Out-of-bounds

- Legal indexes are between **0** and the **list's size() - 1**.
  - Reading or writing any index outside this range will cause an `IndexOutOfBoundsException`.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Marty");    names.add("Kevin");
names.add("Vicki");    names.add("Larry");
System.out.println(names.get(0));          // okay
System.out.println(names.get(3));          // okay
System.out.println(names.get(-1));         // exception
names.add(9, "Aimee");                     // exception
```



ArrayList "mystery" 2

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (int i = 1; i <= 5; i++) {
    list.add(2 * i);   // [2, 4, 6, 8, 10]
}
```

What is the output of the following code?

```
for (int i = 0; i < list.size(); i++) {
    list.add(i, 42);   // add 42 at index i
}
System.out.println(list);
```

Answer: Infinite Loop!

ArrayList "mystery" 3

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (int i = 1; i <= 5; i++) {
    list.add(2 * i);   // [2, 4, 6, 8, 10]
}
```

What is the output of the following code?

```
int size = list.size();
for (int i = 0; i < size; i++) {
    list.add(i, 42);   // add 42 at index i
}
System.out.println(list);
```

Answer:

```
[42, 42, 42, 42, 42, 2, 4, 6, 8, 10]
```

Objects storing collections

- An object can have an array, list, or other collection as a field.

```
public class Course {

    private ArrayList<Student> students;

     public Course() {
        students = new ArrayList<Student>();
          ...
     }
```

- Now each object stores a collection of data inside it.

Please Buy this Book

AP Computer Science A: With 6 Practice Tests (Barron's Test Prep) Ninth Edition

https://www.amazon.com/AP-Computer-Science-Practice-Barrons/dp/1438012896

Make sure it is the new **ninth edition**. It is the only edition that reflects all of the changes for May 2020 exam.

If you want to do well on the AP Exam, this would be your best investment.

It is recommended that you buy the hard copy edition. Many students like to simulate the actual AP Exam experience by writing and working directly in the book.

Lab 1

Write the following static methods.

- Write the method `smallest` which accepts an arraylist of integers and return the smallest. Return the first one if there are multiple smallest values. Your return type should be an Integer. **MUST USE a for each loop.**

- Write the method `longest` which accepts an arraylist of strings and return the longest string. Return the first one if there are multiple longest strings. **MUST USE a regular for loop.**

- Write the method `remove` which accepts an arraylist of Integers `list` and an integer variable `x`. Remove all occurrences of `x` in `list`.

Lab 2(Processing)

A random set of Ball objects move about on the screen. As the mouse touches a Ball, it is removed from the screen.

A template for this lab with comments explaining the lab is available on my website:

https://longbaonguyen.github.io/courses/apcsa/processing/ArrayListLab.zip

Lab 3(Processing)

Tank Shooting Lab: Download the template for this processing lab on my website.

Tank shoots bullets.

Crate is removed if hit by a bullet.

Score keeps track of number of crates hit.