CS 112: Array Lists

Ellen Veomett

Nov 29, 2022

My Problem: Making a List of my Favorite Foods

Idea for Solution: Use an Array

- Array stores objects
- Objects in an array are stored in order

My Problem: Making a List of my Favorite Foods

Idea for Solution: Use an Array

- Array stores objects
- Objects in an array are stored in order

Issues with Arrays

- It's awkward to remove elements
- It's awkward to add in the middle while keeping all other elements
- We must know the size when it is declared
- We can't change the size (shrink or grow) later

Solution: ArrayList!

- We can increase/decrease the size of an ArrayList
- It's easy to delete in the middle or insert in the middle
- We can access by index (just like an Array)
- We get nice methods for finding, adding, deleting objects.

Note:

- ArrayLists hold a list of objects (stores references)

Basic Usage

```
import java.util.ArrayList; // So we can declare an ArrayList

public class ArrayListPractice {
    public static void main(String[] args) {
        // declare a new ArrayList that contains base type
            Strings
        ArrayList<String> myStrings = new ArrayList<String>();
        // The base type (String here) is called the "type
            parameter"
        // Note: ArrayLists must contain *objects* and not
            *primitives*
        myStrings.add("When");
        myStrings.add("is");
        myStrings.add("lunch?");
        System.out.print(myStrings); // [When, is, lunch?]
        System.out.print(myStrings.size()); // prints:
        System.out.print(myStrings.get(1)); // prints:
    }
}
```

Comparison: Arrays, ArrayLists

```
int[] intArray = new int[2];
ArrayList<Integer> intArrayList = new ArrayList<Integer>(); //
    initial size 0

intArray[0] = -3;
intArrayList.add(-3); // autoboxing

intArray[1] = 42;
intArrayList.add(1,42); // autoboxing

intArray[0] = 99;
intArrayList.set(0, 99); // autoboxing

System.out.println(intArray[1]); // 42
System.out.println(intArrayList.get(1)); // 42 (autoboxing)
```

Printing/For Each

```
System.out.println(Arrays.toString(intArray)); // [99, 42]
System.out.println(intArrayList); // [99, 42]

for (int element : intArray){ // 99 42
    System.out.print(element + " ");
}
for (Integer element : intArrayList){ // 99 42
    System.out.print(element + " ");
}
```

Printing/For Each

```
System.out.println(Arrays.toString(intArray)); // [99, 42]
System.out.println(intArrayList); // [99, 42]

for (int element : intArray){ // 99 42
    System.out.print(element + " ");
}
for (Integer element : intArrayList){ // 99 42
    System.out.print(element + " ");
}
```

Pair Programming (count off): Implement the steps of FavoriteFoods.java, now using an ArrayList

Some ArrayList Methods

Method	**Action**
myList.add(object);	Adds object to *end*
myList.get(index);	Finds object at that index
myList.isEmpty();	`true` if list is empty (otherwise `false`)
myList.set(index, object);	Replaces entry at given index with object
myList.size();	Gives number of elements in list
myList.indexOf(object);	Returns first index of object (or -1 if object not present)
myList.remove(index);	Removes entry at given index
myList.add(index, object)	Adds object so that its index is the given index. Everything else shifts right
myList.remove(object)	Adds *first* occurrance of object (if present) Note: you need to explicitly wrap Integers!

Some Nuances

```
// intArrayList.remove(99); // Will not compile!
intArrayList.remove(Integer.valueOf(99));

int[] testArray = new int[5];
ArrayList<Integer> testArrayList = new ArrayList<Integer>(5);
    // Initial size 0, initial *capacity* 5
for (int i=0; i<5; i+=2){testArray[i] = i;} // compiler OK
    with this
//for (int i=0; i<4; i+=2){testArrayList.add(i,i);} //
    compiler NOT OK with this
//testArrayList.set(0,1); // compiler NOT OK with this either
```

```
class Range {
    private int length;
    private int[] elements;
    Range(int num){
        length = Math.max(0, num); // If num < 0 length is 0
        elements = new int[length];
        for (int i = 0; i<length; i++){
            elements[i] = i;
        }
    }
    Range(int num1, int num2, int step){
        length = Math.max(0, (num2-num1)/step); // if num1 >
            num2, length is 0
        elements = new int[length];
        for (int i = 0; i< length; i ++){
            elements[i] = num1 + i * step;
        }
    }
    public String toString(){
        return Arrays.toString(elements);
    }
}
```

Pair Programming:

Refactor our Range class to use an ArrayList instead of an Array. If time:

```
/**
 * Removes all occurrences of i in the elements of this
 * @param i number which is entirely removed from elements
 */
void rmAll(Integer i){
   // TODO
}
/**
 * Returns the concatenation of two RangeAL lists
 * @param range2 is concatenated behind this
 * @return this followed by range2
 */
RangeAL concat(RangeAL range2){
   // TODO
}
```

https://www.programiz.com/java-programming/library/arraylist

Why do we need both?

- Array benefits
  - Faster retrieval from Array
  - Less space used in Array
- ArrayList benefits
  - ArrayList can be lengthened or shortened (more flexible)
  - ArrayList comes with lots of useful methods
- Note: ArrayList is a class implementation of collection interface
  - We'll see more later: LinkedList, Stack, PriorityQueue

Use Array when speed is a priority and you know what size you'll need.