Lecture 16: Arraylist

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

Problem with Arrays

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

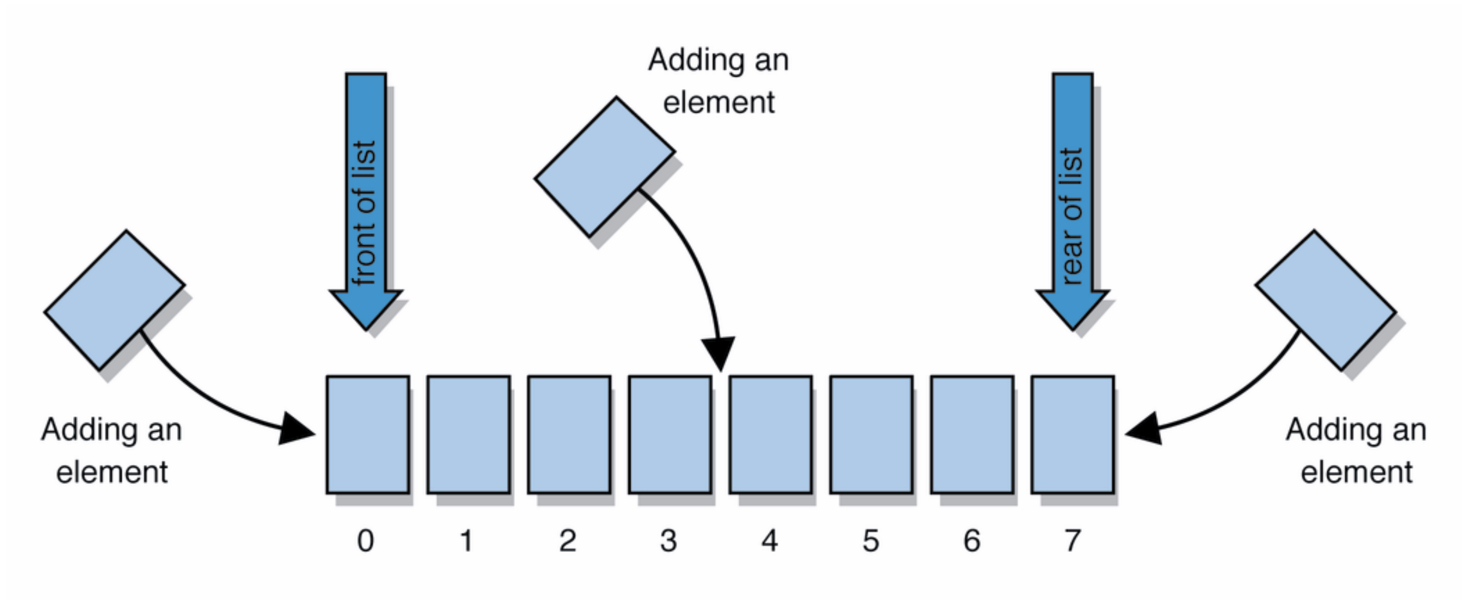
```
String[] allWords = new String[1000];
int wordCount = 0;
```

- 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)

`add(`**`value`**`)`	appends value at end of list
`add(`**`index`**`, `**`value`**`)`	inserts given value just before the given index, shifting subsequent values to the right
`E get(`**`index`**`)`	returns the value at given index
`E remove(`**`index`**`)`	removes/returns value at given index, shifting subsequent values to the left
`E set(`**`index`**`, `**`value`**`)`	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.

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"
```

ArrayList vs. array 2

- doing something to each value that starts with "B"

```
for (int i = 0; i < names.length; i++) {
    if (names[i].startsWith("B")) { ... }
}
```

```
for (int i = 0; i < list.size(); i++) {
    if (list.get(i).startsWith("B")) { ... }
}
```

Remove Plurals

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



// allWords is an arraylist of words entered by user.
// remove all plural words
for (int i = 0; i < allWords.size(); i++) {
    String word = allWords.get(i);
    if (word.endsWith("s")) {
        allWords.remove(i);

    }
}

//error, once a word is remove, words are shifted to the
//left, this will cause the next word to be skipped
```

Remove Plurals

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



// allWords is an arraylist of words entered by user.
// 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--; //corrected
    }
}
```

ArrayList as parameter

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

- Example:

```
// Removes all plural words from the given list.
public static void removePlural(ArrayList<String> list) {
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i).endsWith("s")) {
            list.remove(i);
            i--;
        }
    }
}
```

- You can also return a list:

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

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
```

Wrapper Classes

Primitive Type	Wrapper Type
`int`	`Integer`
`double`	`Double`
`boolean`	`Boolean`

Java allows wrapper object for each of its primitive types. The two that will be on the AP Exam are Integer and Double classes.

Wrapper Classes

Integer and Double are wrapper classes…not Rapper Classes.

These are Rapper Classes:

```
public class Tupac{…}
public class Biggie extends Tupac{…}
public class JayZ extends Biggie{…}
public class KendrickLamar extends Biggie{…}
```

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?

```
[10, 20, 30, 40, ..., 100]
```

ArrayList "mystery"

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (int i = 1; i <= 10; i++) {
    list.add(10 * i);   // [10, 20, 30, 40, ..., 100]
}
```

- What is the output of the following code?

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

- Answer:

```
[20, 40, 60, 80, 100]
```

Remove elements

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

// Removes all elements with even values from the given
  list.
public static void filterEvens(ArrayList<Integer> list) {
    for (int i = 0; i <list.size(); i++) {
        int n = list.get(i);
        if (n % 2 == 0) {
            list.remove(i);
        }
    }
}

// INCORRECT!!
```

Solution 1

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

// Removes all elements with even values from the given list.
public static void filterEvens(ArrayList<Integer> list) {
    for (int i = 0; i <list.size(); i++) {
        int n = list.get(i);
        if (n % 2 == 0) {
            list.remove(i);
            i--; // prevents skipping
        }
    }
}

// CORRECT!!
```

Solution 2

Or alternatively, we can traverse the array backwards; this eliminates the need to do i--.

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



// Removes all elements with even values from the given list.
public static void filterEvens(ArrayList<Integer> list) {
    for (int i = list.size() - 1; i >= 0; i--) {
        int n = list.get(i);
        if (n % 2 == 0) {
            list.remove(i);
        }
    }
}

// Correct Version, go backwards, no need to do i--.
```

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
```

index	*0*	*1*	*2*	*3*
value	Marty	Kevin	Vicki	Larry

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?

```
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]
```

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?

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

Infinite Loop!

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

Barron's AP Computer Science A with Online Tests
(Eighth Edition)

https://www.amazon.com/Barrons-Computer-Science-Online-Tests/dp/1438009194
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.

- 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.

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

Lab 2

Write the class Question as describes below. You can add more methods but must have the methods listed below.

```
public class Question{
  private String question;
  public Question(String q){}
  public String getQuestion(){}
  public void setQuestion(String q) {}
  public String toString(){
  }
}
```

Lab 2

Write the class ShortAnswer which extends Question.

```
public class ShortAnswer extends Question{
  private String correctAnswer;
  public ShortAnswer(String q, String correct){
  }

  // must call toString of Question.
  //Who is the first President of the United States?
  //Answer:
  public String toString(){
  }

  // Format should be: Answer: George Washington
  public void printAnswer(){
  }
}
```

Lab 2

Write the class MultipleChoice which extends Question.

```
public class MultipleChoice extends Question{
  private ArrayList<String> answers;
  private String correctAnswerLetter; //A-E
  public static final String LETTERS = "ABCDE";
  public MultipleChoice(String q, ArrayList<String> ans, String
correct){}
  public MultipleChoice(String q){}

  // add to end of list.
  public boolean addChoice(String choice, boolean isCorrect){}
  public String toString(){}
  public void printAnswer(){
  }

}
```

Lab 2

A MultipleChoice question must have at most five choices(A-E). The arraylist `answers` contains answer choices for the multiple choice question.

```
public boolean addChoice(String choice, boolean isCorrect){}
```

Add choice to the end of the arraylist answers only if it has less than 5 answers. Update correctAnswerLetter if isCorrect is true. Even if answers already have the correct answer, to keep things simple, if another correct answer is added, the new one is the correct answer.

Returns true if properly added. Discard choice if answers is maxed out(5) and returns false.

Lab 2

`public String toString(){}`

This method overrides toString of Question. Returns a string in test format. You must use `LETTERS` in this method. For example:

Who is the first president of the United States?

A. Abraham Lincoln

B. George Washington

C. Franklin Roosevelt

Hint: Use the "\n" escape sequence to add a new line to your String.

For example, String x = "hi"; x += "\n"; // add new line to the String.

`public void printAnswer(){}`

Prints out the answer to the question.

If the question is multiple choice:

Answer: B: George Washington

Lab 2

Write a driver class with a main method to create three Questions: 1) a Question object, 2)A ShortAnswer object and a 3)MultipleChoice object.

You should be careful with the MultipleChoice object, make sure that as you're adding answer choices, toString is printing the question properly.