Android Basics: Multiscreen Apps

Arrays, Lists, Loops & Custom Classes

- How to some a list of words in the app (Arrays, ArrayLists)
- Display a list of words
- Display a list of English/Miwok word pairs
- Add the words from all the remaining categories

Arrays

<strong>All variables need a data type.</strong>

<strong>You can have an array of any data type:</strong>

- int
- double
- boolean
- char
- long
- float
- short
- byte
- Objects

```
// Create array
int [] showSizesAvailable = new int[3];

// Initialize elements in an array
shoeSizesAvailable[0] = 5;
shoeSizesAvailable[1] = 7;
```

```
shoeSizesAvailable[2] = 10;

// Access elements in an array
shoeSizesAvailable[0] = 5;
shoeSizesAvailable[1] = 7;
shoeSizesAvailable[2] = 10;

// Get the array length
shoeSizesAvailable.length // -> value of 3
```

<html>
<strong><h3>Array vs ArrayList</h3></strong>
<table>
<tr>
<th></th>
<th><strong>Array</strong></th>
<th><strong>ArrayList</strong></th>
</tr>
<tr>
<td>Is it a class?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can it change size once created?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Does it use methods to access and modify elements?</td>
<td>No</td>
<td>Yes</td>
</tr>
</table>
</html>

ArrayLists

1. Documentation: List | Android Developers
2. Documentation: AbstractList | Android Developers
3. Documentation: ArrayList | Android Developers

<strong>Concrete Class: ArrayList</strong>

```
public class ArrayList {

  int size;
  Object[] elements;

  boolean add(E e) {
      elements[size++] = e;
  }

  E remove(int index){
      elements[--size] = null;
  }

  E get(int index) {
      return(E) elements[index];
  }

...
}
```

<strong>Abstract Class: AbstractList</strong>

```
public abstract class AbstractList {

  boolean add(E e) {
      add(size(), e);
      return true;
}
```

```
  E get(int index);

...
}
```

<strong>Interface: List</strong>

```
// No implementation of its methods
public interface List {

  boolean add(E object);

  E remove(int index);

  E get(int index);
}
```

Type Parameters

<table>
    <tr>
        <th><strong><u>Symbol</u></strong></th>
        <th><strong><u>Type</u></strong></th>
    </tr>
    <tr>
        <td>E</td>
        <td>Element</td>
    </tr>
    <tr>
        <td>K</td>
        <td>Key</td>
    </tr>
    <tr>

```
        <td>N</td>
        <td>Number</td>
    </tr>
    <tr>
        <td>T</td>
        <td>Type</td>
    </tr>
    <tr>
        <td>V</td>
        <td>Value</td>
    </tr>
    <tr>
        <td>S, U, V, etc.</td>
        <td>2nd, 3rd, 4th types (when there is more than one parameter)</td>
    </tr>
</table>
```

How to Create and Access Elements in an ArrayList

```
<dl>
    <dt>
        Create an ArrayList
    </dt>
    <dd>
        <pre><code>
            ArrayList&lt;String&gt; musicLibrary = new ArrayList&lt;String&gt;();
        </code></pre>
    </dd>
<br>
    <dt>
        Add elements in an ArrayList
    </dt>
    <dd>
        <pre><code>
            musicLibrary.add(“Yellow Submarine”);<br>
            musicLibrary.add(“Thriller”);<br>
            // Adds an element at a specific index<br>
```

```
                musicLibrary.add(0, “Blue Suede Shoes”);
            </code></pre>
        </dd>
<br>
        <dt>
            Access elements in an ArrayList
        </dt>
        <dd>
            <pre><code>
                musicLibrary.get(0);<br>
                musicLibrary.get(1);<br>
                musicLibrary.get(2);
            </code></pre>
        </dd>
<br>
        <dt>
            Remove elements from an ArrayList
        </dt>
        <dd>
            <pre><code>
                // Remove the element at the specific index<br>
                musicLibrary.remove(2);
            </code></pre>
        </dd>
<br>
        <dt>
            Get ArrayList length or size
        </dt>
        <dd>
            <pre><code>
                musicLibrary.size();
            </code></pre>
        </dd>
</dl>
```

Loops

While Loops

The code within the loop will be executed until the condition in parentheses is `false`

```
while(<condition>){
  <instructions>
}
```

Steps:

1. What is the task to repeat?
   - ➜ <instructions>
2. How many times to repeat?
3. What is the condition?
   - ➜ Count variable < 3

```
int count = 0;

while(count < 3){
  playSound();
  count += 1;
}
```

Counter Shorthands

<table>

<tr>

    <th><strong><u>Shorthand</u></strong></th>

    <th><strong><u>Meaning</u></strong></th>

</tr>

<tr>

    <td>i++</td>

    <td>i = i + 1</td>

</tr>

<tr>

```
        <td>j--</td>
        <td>j = j - 1</td>
</tr>
<tr>
        <td>j += 3</td>
        <td>j = j +3</td>
</tr>
</table>
```

For Loop

```
for (int index=0; index < 3; index++){
  Log.v("NumbersActivity", "Index: " + index + "Value: " + words.get(index));
}
```

View Recycling

When looking at 1000 contacts,
Only 5 will appear on the screen at any time.
Instead of creating 1000 TextViews, we create 5, and change the content as the user scrolls.
This is called view recycling.

Using a ListView and an ArrayAdapter, we can create just enough views based on what we need to fill the screen.

We create a bunch of views, and have a pile of scrap views, that aren't currently being shown on the screen.

When a user scrolls down, and a View goes off of the top of the screen, it gets cycled back around to the top of the screen to be used again.

When a user scrolls up, and a View goes off the bottom of the screen, it gets cycled back around to the bottom of the screen to be used again.

DEPRECATED: ~~Documentation: Memory Monitor | Android Studio~~

Memory Monitor : a feature in Android Studio that keeps track of your application's memory usage (DEPRECATED)

Documentation: View the Java Heap and Memory Allocations with Memory Profiler | Android Studio
Memory Profiler : a feature introduced in Android Studio 3.0 that helps you identify memory leaks and memory churn.

LinearLayout does not recycle views
ListView, GridView, RecyclerView all recycle views.

ListView and ArrayAdapter

The ListView and the ArrayAdapter work together. Without the adapter, the ListView is an empty container.
The ArrayAdapter holds the actual data, and knows how to change it into a format that creates a list view.

Using an ArrayAdapter with ListView · codepath/android_guides Wiki · GitHub

Generics

```
// We use one Generic class
ArrayList<E>

// Instead of creating a class for EVERY type of possible list
StringList class
IntegerList class
AnyObject class
// etc
```

Adapters

Concrete class:

```
public ArrayAdapter{

  public boolean has StableId(){
      return false;
}

  void notiftyDataSetChanged(){
      mDataSetObservable.notifyChanged();
}

...

}
```

Abstract class:

```
public abstract class BaseAdapter{


  public boolean has StableId(){
      return false;
}

  void notifyDataSetChanged(){
      mDataSetObservable.notifyChanged();
}

...
}
```

Interface:

```
public interface ListAdapter {

  public boolean areAllItemsEnabled();

  boolean isEnabled(int position);
}
```

Encapsulation: When to Create a Custom Class

Encapsulation is evident in Java Classes. All the logic and properties of the Object is contained in the class.

A class definition contains both state and methods.

State is the properties of the object, while methods are functions that the Object can call.

Example: Soundcloud

```
Class: Song
State: album_art, song_title, number_of_listens
Methods: getSongTitle();, getAlbumName();
```

- ListView calls the ArrayAdapter's getView() method when it needs a new view.
- One of the input arguments into the method is the desired element of the list.
- Taking in this information, the adapter returns a view to the ListView, that is populated with the data from the specified index

Example:

  - So, like a **surgeon** asking for a tool from the **technician** :

1. Surgeon (ListView) asks for scissors(View).

2. The Technician(getView()), gets the request. The Technician gets the scissors.
3. The technician hands the scissors to the surgeon.

Custom ArrayAdapter Example:

> **Single AndroidFlavor Object**
> State: vNumber
> State: Image
> State: vName

> **androidFlavors Array**
> Many AndroidFlavor objects are stored in the androidFlavors array.
> (index): object
> 0: androidFlavor1
> 1: androidFlavor2
> 2: androidFlavor3

#programming/java/android/beginner/3-multiscreen-apps/2-arrays-lists-loops-custom-classes