Version 1.0

2018-JuL-08



Interview Feedback

Homework

Presenter: Arturo Sanchez

Mobile Apps Company

3301 Windy Ridge Parkway, Suite 400, Atlanta, GA 30339

# **Interview Feedback**

## Introduction

### Elevator

My name is Arturo Sanchez Chavarria, I am 28 years old and I have been working as an Analyst Developer since 2012.

I like to walk in the park, listen to music, play video games, and watch some movies.

I consider myself as a friendly, responsible, cheerful, optimistic and hardworking person.

I really like to be a developer and working as a team.

### Previous Experience

I have mostly worked with .NET technologies using SQL Server as the main Database server, maintaining, updating and creating new applications for the companies that I have been assign.

I have worked with Visual Studio 2010 up to 2017 using C# as primary language (also used VB .NET about 1 year).

In Visual Studio, I have created Web Applications (WebForms, MVC, and Classic ASP), Windows Applications (WinForms, WPF) and Web Services (WCF, ASMX).

In the information storage, I have mostly used SQL Server (2000 up to 2014), but also worked with Progress (10.1 b), XML, SQL LITE.

I have been a code auditor as one of my responsibilities in previous jobs so I need to work with the best practices in code implementation (OOP), that also includes validating design in databases, web services, and interfaces.

I have been involved in almost all of the steps in the creation of an application:

* Creating work plan
* Requirements survey
* Clients interviews
* Creating prototypes
* Designing the application
  + database solution
  + interfaces design (if applies)
  + Web Services Design (if applies)
* Developing the application
* Unit testing
* Publishing the application
  + Initial loads (Database)
* Post-implementation support.

I have also been involved in optimizing the performance of certain existing process as part of the developing phase.

In Android, I have worked with the platform architecture, supporting multiple languages, units of measure, components, context, intents, life cycle of an activity, intents, serializable and Parcelable classes.

## Questions Asked

### What are some containers? Types?

Containers is a bucket description for Views that wrap dynamic content. They are more specialized than Layouts and can but does not have to extend a Layout. Some extend ViewGroup indirectly e.g ListView, some do not e.g. VideoView. The Container label is used in Android Studio but is not part of the class name.

* ListView
* RecycleView

### ListView vs RecyclerView?

RecyclerView was created as a ListView improvement, so, you can create an attached list with ListView control, but using RecyclerView is easier as it:

* Reuses cells while scrolling up/down
  + This is possible with implementing View Holder in the ListView adapter, but it was an optional thing, while in the RecycleView it is the default way of writing adapter.
* Decouples list from its container
  + You can put list items easily at run time in the different containers (linearLayout, gridLayout) with setting LayoutManager.
* Animates common list actions
  + Animations are decoupled and delegated to ItemAnimator.

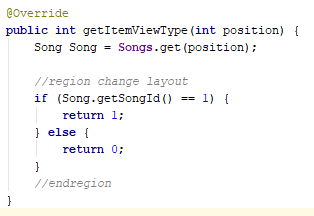
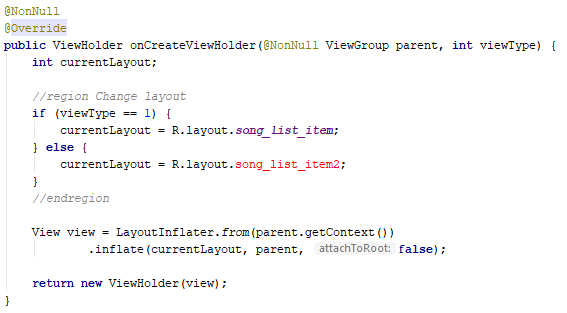
While ListView can crash your application if set too many items in it.

RecyclerView is a more flexible control for handling "list data" that follows patterns of delegation of concerns and leaves for itself only one task - recycling items.

### How to implement the adapter class in RecyclerView?

1. Create a Class that will be your ViewHolder
   1. Extend the ViewHolder Class from RecyclerView.ViewHolder
   2. In the constructor of the class set the controls from the layout for the item
2. Create a Class that will be your Adapter
   1. Extend the class from RecyclerView.Adapter<YourViewHolder>
   2. Create a List from the type of the object that you will receive in the adapter
   3. Set the List in the constructor
   4. Override the onCreateViewHolder method to inflate the layout
   5. Override the onBindViewHolder method to set the values from the current item to the controls in the layout.

### How to use different viewTypes in RecyclerView?

1. In your Adapter class override getItemViewType the method to change the return value depending on the properties of your object.
   1. 
2. In your Adapter class onCreateViewHolder method evaluate the *viewType* value to change the layout.
   1. 

### What is a Fragment? Ways to create?

You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

Ways to create a fragment:

* Static
  + Create a Fragment in the layout of the activity that you want the fragment to be.
  + Set an id for the fragment
  + Set the name of the fragment to be the namespace from your fragment class
* Dynamic
  + Create an instance from your fragment in the activity that you want it to be.
  + Add your fragment instance to your container using the FragmentManager and a Transaction.

### What is the lifecycle of a fragment?

1. onAttach()
2. onCreate()
3. onCreateView()
4. onActivityCreated()
5. onStart()
6. onResume()
7. onPause()
8. onStop()
9. onDestroyView()
10. onDestroy()
11. onDetach()

### What is a FragmentTransaction?

Is the ability to add, remove, replace, and perform other actions with Fragments in response to user interaction. Each set of changes that you commit to the activity is called a transaction.

### Can a fragment exist on their own?

No. A fragment must always be hosted in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.

### How to communicate among fragments?

* Creating a method for obtaining values at the creation of the instance of the fragment.
* Creating an interface in the fragment and implementing the interface in the Activity that will call the fragment.
* Using a broadcastreceiver
* Using a third-party component like EventBus

### Does a Fragment needs a UI?

No, you can also use a fragment to provide a background behavior for the activity without presenting additional UI.

### How to handle backstack with Fragments? Remove?

Adding

* When adding the fragment to you container in the Activity, specify the addToBackStack method from the FragmentTransaction.

Remove

* In the FragmentManager instance call the popBackStackImmediate method to remove the fragment.

### How to retain fragment instance?

Control whether a fragment instance is retained across Activity re-creation (such as from a configuration change). This can only be used with fragments not in the back stack. If set, the fragment lifecycle will be slightly different when an activity is recreated:

* onDestroy() will not be called (but onDetach() still will be, because the fragment is being detached from its current activity).
* onCreate(Bundle) will not be called since the fragment is not being re-created.
* onAttach(Activity) and onActivityCreated(Bundle) will still be called.

### What is a service? Types of services?

An android component that can perform long-running operations in the background. It continues to run even if the user switches applications

Types of services:

* Bound Service
  + A bound service is a service that has some other component (typically an Activity) bound to it. A bound service provides an interface that allows the bound component and the service to interact with each other. Once there are no more clients bound to the service, Android will shut the service down.
* IntentService
  + An IntentService is a specialized subclass of the Service class that simplifies service creation and usage. An IntentService is meant to handle individual autonomous calls. Unlike a service, which can concurrently handle multiple calls, an IntentService is more like a work queue processor – work is queued up and an IntentService processes each job one at a time on a single worker thread. Typically, an IntentService is not bound to an Activity or a Fragment.
* Started Service
  + A started service is a service that has been started by some other Android component (such as an Activity) and is run continuously in the background until something explicitly tells the service to stop. Unlike a bound service, a started service does not have any clients directly bound to it. For this reason, it is important to design started services so that they may be gracefully restarted as necessary.

### Service vs IntentService?

IntentService is a subclass of Service, written in Java code. Hence, anything an IntentService does, a Service could do, by including the relevant bits of code that IntentService uses.

The three primary features of an IntentService are:

* the background thread
* the automatic queuing of Intents delivered to onStartCommand(), so if one Intent is being processed by onHandleIntent() on the background thread, other commands queue up waiting their turn
* the automatic shutdown of the IntentService, via a call to stopSelf(), once the queue is empty

Any and all of that could be implemented by a Service without extending IntentService.

### How to communicate with a bound service?

1. If application and service are running in same process which is mostly the case then create your own interface by extending [Binder](https://developer.android.com/reference/android/os/Binder.html) class and return an instance of it in [onBind()](https://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29).
2. If service is running in separate process then you can create an [AIDL](https://developer.android.com/guide/components/aidl.html) interface. Most applications should not use this as it may require a lot of multi-threading capabilities and may make code more complex.
3. If you need an interface to communicate across processes you can create a [Messenger](https://developer.android.com/reference/android/os/Messenger.html). It handles communication on single thread. Client can also define a messenger and pass it to service using which service can send data back to client.

### What is a Binder? How to use with Bound service?

Binder is an Android-specific interprocess communication mechanism, and remote method invocation system.

That is, one Android process can call a routine in another Android process, using binder to indentify the method to invoke and pass the arguments between processes.

* Create a Service
* Create a binder class that extends from Binder
  + Create an IBinder interface instance.
  + Create a method to get the data from the server.
    - This returns the context for the service.
* On the component that you want to use your service
  + Create a ServiceConection and override the onServiceConnected method
  + Create an instance of your bound service in the onServiceConnected method.
  + Call the method from the bound service instance to get your data from the server.

### Foreground vs background service?

* A service which is visible to the user. It must provide a notification on the status bar. Foreground services are not candidates to be killed when the system is low on memory.
* The process which runs without user interaction i.e receiving a message, incoming call, receiving mails, or setting alarms. The method used here is onStart() and onStop().

### Different ways to store data on Android? Explain each

* Files
  + Allows you to store data in a file similar to disk-based file systems on other platforms.
* Databases
  + Allows you store data in structured manner.
  + Can also be queried and the data retrieval is much more robust.
* Shared Preferences
  + An Android interface that allows apps to store key-value pairs of primitive data types

### How to create a database using SQLite?

1. Create a Data Contract class.
2. Specify the Version and name of the database.
3. Create the Database Definition Language (DDL) for creating your tables.
4. Create the Database Manipulation Language (DML) for creating your queries.
5. Create a Data Source class that extends from SQLiteOpenHelper.
6. Override the onCreate Method for creating your tables (using the Data Contract class).
7. Create new methods on Data Source class for your queries.

### What is Room? How to use it?

Database library.

Provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite.

How to use it:

1. Add dependencies to your project (build.gradle).
2. Create entity - Class with @Entity annotation.
3. Define primary key for the entity - Property with @PrimaryKey and @NonNull annotation.
4. Create Data Access Object (DAO) interface.
5. Define the signature methods in your interface with annotations.
6. Create database - Abstract class that extends from RoomDatabase with @Database (entities, version) annotation.
7. Create Helper - Class that gets the context, creates the database with the databaseBuilder, and implements the methods from the DAO.

### What is JobScheduler? Why do we need it?

Is an API for scheduling various types of jobs against the framework that will be executed in your application's own process.

We need it because it can run once a specified event occurs such as when the device is idle or when the battery is running low.

### What is difference b/w LinkedList and ArrayList?

LinkedList<E> allows for constant-time insertions or removals using iterators, but only sequential access of elements. In other words, you can walk the list forwards or backwards, but finding a position in the list takes time proportional to the size of the list. Javadoc says "operations that index into the list will traverse the list from the beginning or the end, whichever is closer", so those methods are O(n) (n/4 steps) on average, though O(1) for index = 0.

ArrayList<E>, on the other hand, allow fast random read access, so you can grab any element in constant time. But adding or removing from anywhere but the end requires shifting all the latter elements over, either to make an opening or fill the gap. Also, if you add more elements than the capacity of the underlying array, a new array (1.5 times the size) is allocated, and the old array is copied to the new one, so adding to an ArrayList is O(n) in the worst case but constant on average.