**Capstone II - Tech Talk Overview**

**Grant Saylor, Kyle Smith, Anthony Tran, Jiayi Xu**

**2/10/2021**

**Topic:**

***Total Runtime: 15-20 minutes***

**Presenters:**

* ***Anthony***
* ***Grant***
* ***Kyle***
* ***Jiayi***

**Transcript:**

***<Grant Saylor>***

**1 min – 5 mins:**

***SLIDE 1 – The Art of the Architecture:***

Hi everyone, this is Grant speaking. Today myself, Kyle, Anthony and Jiayi will talk about the architecture of our Android application, primarily asynchronous tasks and how we connect it with other services to create a fully featured app. First up is Kyle to talk about the concept of asynchronous tasks.

***<Kyle>***

**5 mins – 10 mins:**

***SLIDE 2: Standard Programming***

Hi everybody, I'm Kyle, and I'm going to be introducing the concept of Asynchronous tasks.

(CLICK)

 In typical programming, the code you write happens in sequence.  You may have loops, or call other methods where the currently executing line jumps around, but you can always track the order of execution.

***SLIDE 3: API Call***

However, if you are wanting to integrate a different service, or API to your program, you are no longer in complete control of the chain of execution.  Let's say you want to store some data from an outside service into an arrayList.  When you send a query to an API, it will receive your request and do it's own thing, dictated by it's own code, and eventually give you a reply.

***SLIDE 4: How to Handle the API Call***

How do you handle this?  You could have your program halt execution and simply wait for the response.  This raises the issue of idle time in your program, it essentially needs a loading screen while it waits on the API call.  This obviously isn't ideal.  Now consider needing multiple API calls,

(CLICK)

and how long your program will spend waiting for them each to complete.  Not a great plan.

***SLIDE 5: How to Handle the API Call***

Perhaps instead, you could initialize your arrayList with the API call, and place any code that interacts with the object later on so you can give time for the API call to finish.  But if you decide to interact with that arrayList before it has been filled by the API call, your code will throw some kind of error.  This is also a problem because you don't know how long that particular API needs in order to fulfil your request.

(CLICK)

You don't know how long you should space out the rest of your code so that you give the API enough time to give you the data before interacting with that object, and so you don't know how to structure your code around this problem.  Once again, if you have multiple API calls, you need to structure your code around them all, making sure not to read them before they're ready.  It can quickly become very messy.

***SLIDE 6: Asynchronous Tasks***

The solution you need is something called an Asynchronous Task.  This is a concept of using multiple threads of programming to accommodate things that take an indeterminate amount of time.  In that example with the arrayList, if you were to surround the API call and all interactions with the arrayList in an asynchronous task method, you are basically saying "I need you to accomplish these tasks for me, I understand that they won't be ready right away, and they may take time to complete".

(CLICK)

With the code in the asynchronous method, the main thread will create a new thread with the behavior of waiting for things to finish.  So inside this method, you might make your API call, store the data into the arrayList, sort the arrayList, and then perform some if/else statements on the contents.  Each command will not execute until the prerequisite tasks are completed.  In this way, it avoids the issue of interacting with objects before they are initialized or ready to be used.

***SLIDE 7: Asynchronous Tasks – The Many Versions***

Many methods of accomplishing things asynchronously have been written, they all operate in their own way, but the overall goal is the same.  They all handle working with data that takes time to complete its task, and it allows the programmer to multitask, by executing other lines of code while waiting on the ones in an asynchronous method.

***<Anthony Tran>***

**10 mins – 15 mins:**

***SLIDE 8: Why we use Asynchronous Tasks***

Hi my name is Anthony and I’m going to explain why using Asynchronous tasks are very helpful to any programmer. (click) We believe that understanding Async tasks implementation is a beneficial component for program development. In android applications, asynchronous tasks allow the application to run specific instructions in the background, which will then synchronize with the application’s main thread once those instructions are finished. This action allows many convenient ease of use for many applications because this bypasses the standard programming limitation of executing in sequence, while this allows code to run parallel. (clicks a lot until interacts with internet is up) An example of asynchronous task would be a network operation to connect an application to the internet and executing instructions using said internet During this instance, the user can be active on the activity screen of the application, while in the background , the application will do whatever instruction the app needs so that it can update the UI of the main thread when it finishes its instruction.

***SLIDE 9: Transition to Grant***

There are many diverse ways of implementing async tasks in kotlin and java. For our group Libra Works, we incorporated two different types of Async tasks into our virtual library app: JetBrain’s Anko doAsync and FireBase’s Event Listener. I’m going to hand it off to Grant for one of our implementations of async tasks

***<Grant Saylor>***

***Slide 10 – Using an Event Handler/Listener in Our App***

Hi, this is Grant again. I’d like to preview how we use an event listener in our app, Virtual Library. This behaves like an asynchronous task. When our map is being set up, we call the Google Maps API to fill a map fragment (like a map class) with the Google Map view. If we left it like this, the Google Map would just show streets and nothing else, the user wouldn’t be able to manipulate the map or anything at all.

In our code, once a callback to Google Play services is fulfilled, the “OnMapReady” function is triggered, this will allow the map to be manipulated and filled out with markers. You might be more familiar with markers being referred to as “pins”, these are the visual representation of libraries on our map, the colorful book icon. Inside of the OnMapReady method is the event listener we’ve been referring to.

***Slide 11 – Using an Event Handler/Listener in Our App***

Using the event listener we can do various things like allowing users to add additional markers to the map and regenerate the map with existing libraries. Methods are in place to listen for markers being placed around the map, to push and pull the associated data to our database and to add it all to an array to regenerate it for users at a later time.

The event listener is always ready to go and automatically triggers once the map is ready, which to the user should look pretty instant.

To make it a bit clearer, during the loading of the map, all the information to be put onto the map is also being loaded, but it is just waiting to be placed onto the map until the map is ready.

Testing our map without the event listener, or if you placed the event listener in a place where it could trigger before the map is ready would cause a crash. The data is there, but there is nothing to put the data into yet, which is why we wait for the map to finish generating!

Further in the event listener there are calls to a regeneration method that in turn calls the database to retrieve latitude and longitude values amongst others to place the markers onto the map as you see in the screenshots. Every user will see the same markers on their map and if a user places a marker onto the map, the event listener will listen and populate that for every other user as well.

***Slide 12 – Using an Event Handler/Listener in Our App***

There are tons of uses for event listener and async tasks in an android app. In our own app we use it to talk with our Firebase database, whether that is the map screen as you’re all now familiar with or with other screens in our app, such as searching for a book inside our database. All of this is data that we can’t necessarily sequentially program because we don’t have hardcoded values for them, so we do it asynchronously in conjunction with Firebase to prevent a crash scenario. While all of this may seem a bit nested in our code, to the user it all appears seamless and they would never know the intricate dance going on behind the scenes.

I hope this was a useful insight in how asynchronous tasks can make or break your Android application.

I’ll be handing it over to Jiayi so she can close us out with an explanation of Anko and how we use it in our app.

***<Jiayi Xu>***

**15 mins – 20 mins:**

***SLIDE 13:***

This is Jiayi Xu. Getting third-party resources from our website can save us a lot of time developing software without having to generate or store information separately. To read the JSON file on the site and use some information as our resource, we can use "doAsync" for background tasks and update the UI with "UI Threads". Kyle mentioned this section before and explained how it works.

As you can see, for our app, you need to scan the barcode of the book and get the title, cover page, or other information from the Open Library website. You might consider how these are connected. Let's look at an example of our project.

***SLIDE 14:***

This is one of the cases we use in our project. "This: Anko AsyncContext <Barcode> tell us that the doAsync() utility function runs tasks in a background thread, which is box 1 The internal section is used to connect URLs and get resources, and when this process is complete, the UI thread switches back to the main thread, you can look at box 2. You can see "It: Barcode Activity" marked after the UI thread, which is our main thread.

Once connected to the fire base library, we can easily store this information in the fire base library and use it in our features. Simply obtaining information does not meet our software requirements, and displaying the cover of this book is one of the features of this mobile phone software.

***SLIDE 15:***

To read data from firebase, you must use an event listener, and as shown here, we create a function called Add Book Event Listener and use the Value event listener to read data from firebase, and we only need to read the data once from the database. For example, the last four digits of a barcode are 8106, and we want to get the cover of the book, and the event listener will help us grab the cover image code once, and then we can use this cover ID to read the image from another website.

We also use other asynchronous tasks, such as toast, intent, and so on.

This is our group presentation, thank you.

Any questions?