Coroutine Part 4 notes

00:00 Lets begin by making our own ViewModel. This is an interface that our view model will implement. We will just copy what our presenter had.   
  
00:29 Next lets make our ViewModel class and we will call it LoveCalculuatorViewModel

00:48 We then copy everything from our presenter into the new ViewModel class

01:00 And then we do a rename so that everyones happer

01:08 But why are we using a ViewModel? ViewModels are typically tied to the activities lifecycle. This means that we don’t have to manually manage Jobs, and the ViewModel exposes a viewModelScope that can be used

01:32 after we have extended our ViewModel we can delete all the manual job code we added earlier,

01:50 we no longer need to stop the job when the user presses back because the viewmodel is tied to the lifecycle of the app and will cancel the jobs accordingly. We can now delete the stop method from our interface.

02:06 The viewmodelscope now lets us use the coroutine builder to launch our code, however, we need to set its context by giving our exceptionHandler

02:22 We can safely remove the onStop override from out MainActivity

02:32 Finally we need make our activity use our view model instead of the presenter,. Koin allows us to use the by ViewModel. If are not using Koin you will have to use a viewModelProvider

03:00 lets see if everything works as it should

03:09 John and Jane have seen better days.   
  
Footnote. This view model is as testable as with the presenter and is available in the repository