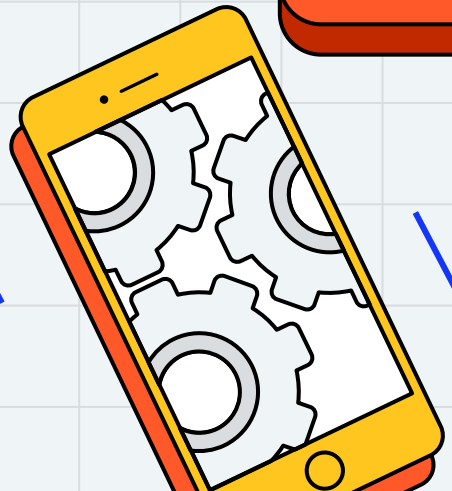
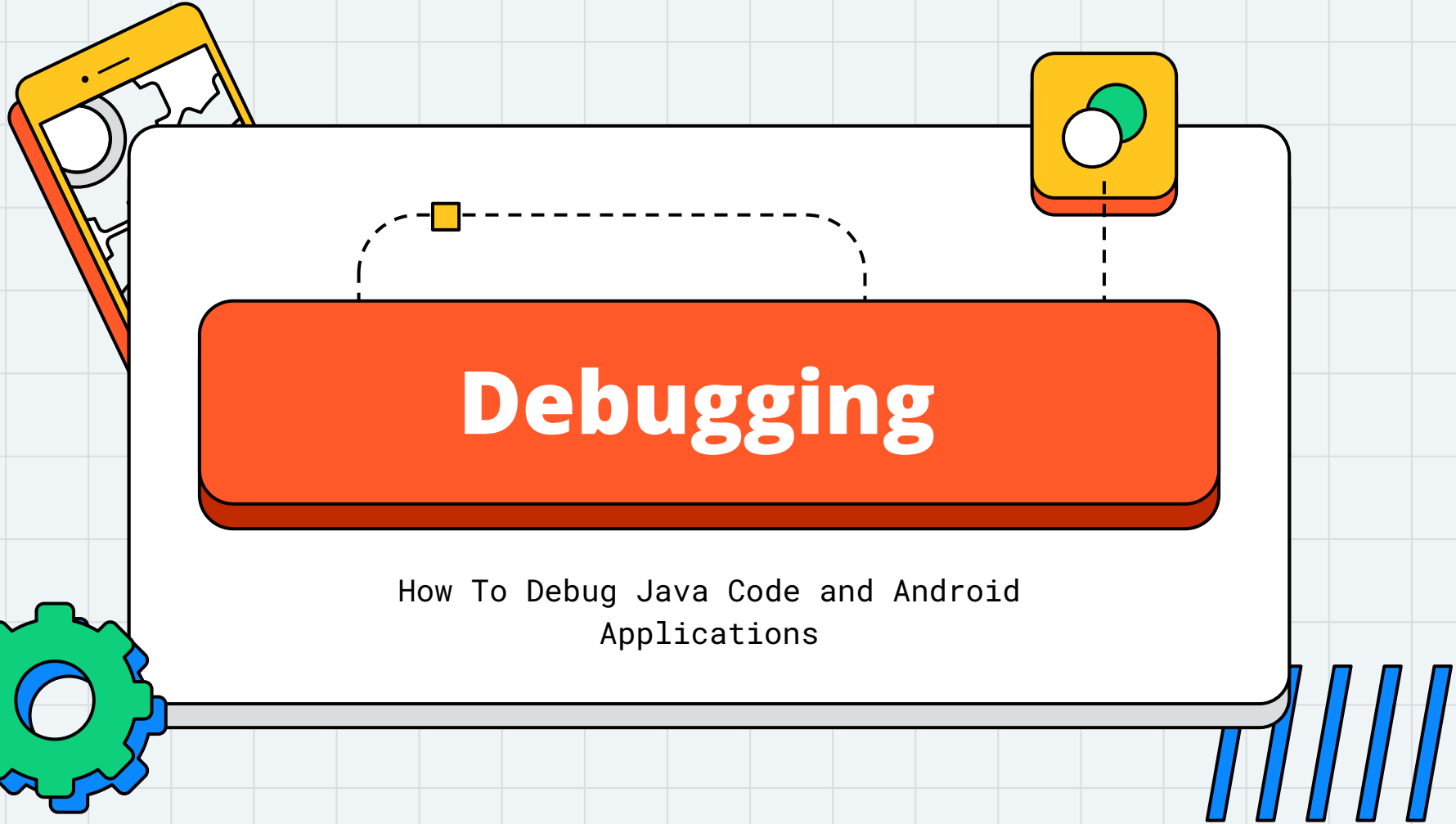
Application Programming

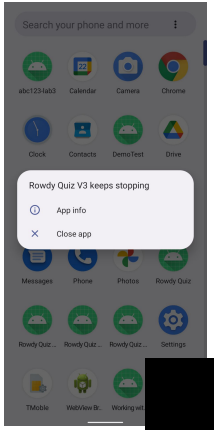


Hend Alkittawi


Debugging
How To Debug Java Code and Android Applications

DEBUGGING ANDROID APPLICATIONS

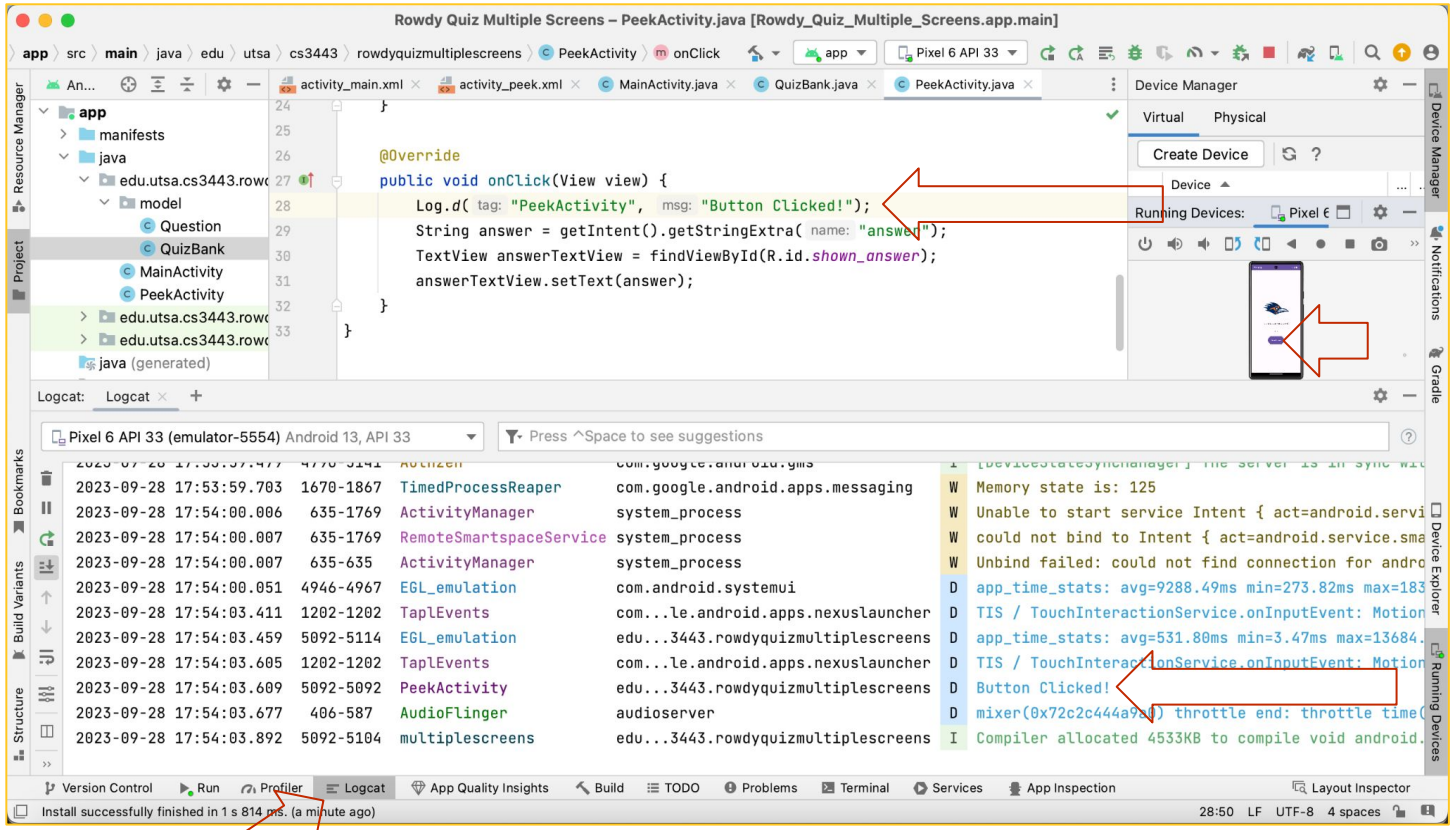
- Some of the problems you may encounter while building android applications. The application may crash at runtime or not function as expected!
- As an IDE, Android Studio has some tools to help developers debug their code. These tools are:
  - The Logcat
  - Android Lint
  - The Debugger



THE LOGCAT

- When an **application is running**, the **Logcat tool window** provides access to **diagnostic messages.**
- The log output displays messages which are created using the **Log class**
  - The syntax to create a log diagnostic message is `Log.[logLevelInitial]("TAG", "Message")`
  - Some of the available log levels:
    - **I**nformation, **V**erbose, **D**ebug, **E**rror
  - the log level and the tag helps with **filtering messages**
  - Example: `Log.d("PeekActivity", "Button Clicked!")`

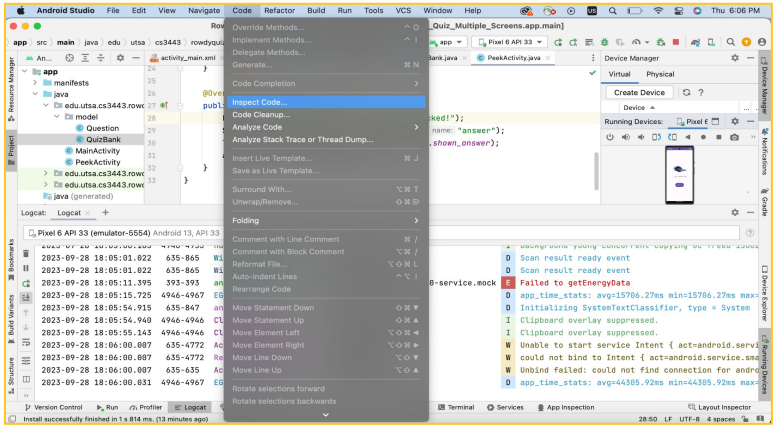
THE LOGCAT

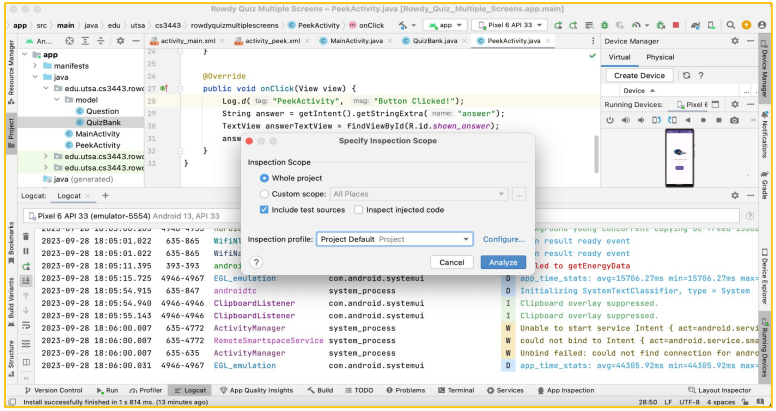

Rowdy Quiz Multiple Screens – PeekActivity.java [Rowdy_Quiz_Multiple_Screens.app.main]
app src main java edu utsa cs3443 rowdyquizmultiplescreens PeekActivity onClick
Pixel 6 API 33
activity_main.xml activity_peek.xml MainActivity.java QuizBank.java PeekActivity.java
app
manifests
java
edu.utsa.cs3443.rowd
model
Question
QuizBank
MainActivity
PeekActivity
edu.utsa.cs3443.rowd
edu.utsa.cs3443.rowd
java (generated)
24 }
25
26 @Override
27 public void onClick(View view) {
28 Log.d(tag: "PeekActivity", msg: "Button Clicked!");
29 String answer = getIntent().getStringExtra(name: "answer");
30 TextView answerTextView = findViewById(R.id.shown_answer);
31 answerTextView.setText(answer);
32 }
33 }
Device Manager
Virtual
Physical
Create Device
Device
Running Devices:
Pixel 6
Logcat:
Logcat
Pixel 6 API 33 (emulator-5554) Android 13, API 33
Press ^Space to see suggestions
2023-09-28 17:53:59.703 1670-1867 TimedProcessReaper com.google.android.apps.messaging W Memory state is: 125
2023-09-28 17:54:00.006 635-1769 ActivityManager system_process W Unable to start service Intent { act=android.servi
2023-09-28 17:54:00.007 635-1769 RemoteSmartspaceService system_process W could not bind to Intent { act=android.service.sma
2023-09-28 17:54:00.007 635-635 ActivityManager system_process W Unbind failed: could not find connection for andro
2023-09-28 17:54:00.051 4946-4967 EGL_emulation com.android.systemui D app_time_stats: avg=9288.49ms min=273.82ms max=183
2023-09-28 17:54:03.411 1202-1202 TaplEvents com...le.android.apps.nexuslauncher D TIS / TouchInteractionService.onInputEvent: Motion
2023-09-28 17:54:03.459 5092-5114 EGL_emulation edu...3443.rowdyquizmultiplescreens D app_time_stats: avg=531.80ms min=3.47ms max=13684.
2023-09-28 17:54:03.605 1202-1202 TaplEvents com...le.android.apps.nexuslauncher D TIS / TouchInteractionService.onInputEvent: Motion
2023-09-28 17:54:03.609 5092-5092 PeekActivity edu...3443.rowdyquizmultiplescreens D Button Clicked!
2023-09-28 17:54:03.677 406-587 AudioFlinger audioserver D mixer(0x72c2c444a9a0) throttle end: throttle time(
2023-09-28 17:54:03.892 5092-5104 multiplescreens edu...3443.rowdyquizmultiplescreens I Compiler allocated 4533KB to compile void android.
Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector
Install successfully finished in 1 s 814 ms. (a minute ago)
28:50 LF UTF-8 4 spaces
Resource Manager
Project
Bookmarks
Build Variants
Structure
Device Manager
Notifications
Gradle
Device Explorer
Running Devices

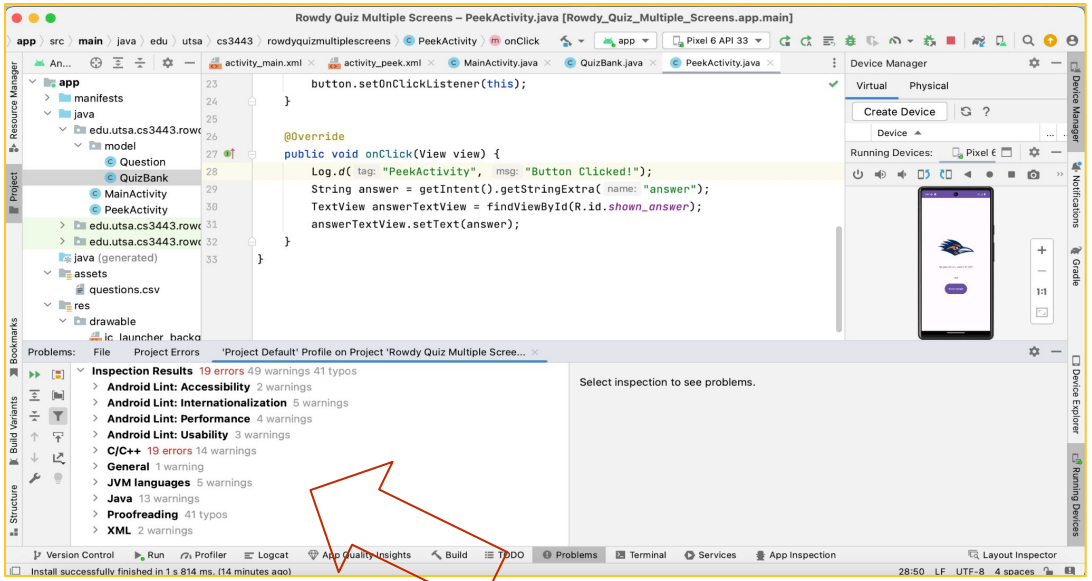
ANDROID LINT

- Android Lint is a static analyzer for Android code
    - a program that examines your code to find defects without running it
- Android Lint is aware of the Android framework components and and can **find problems that the compiler cannot find**.
- For example, it is good for finding issues in XML files.
- To run Android Lint: Code > Inspect Code the click Analyze

ANDROID LINT







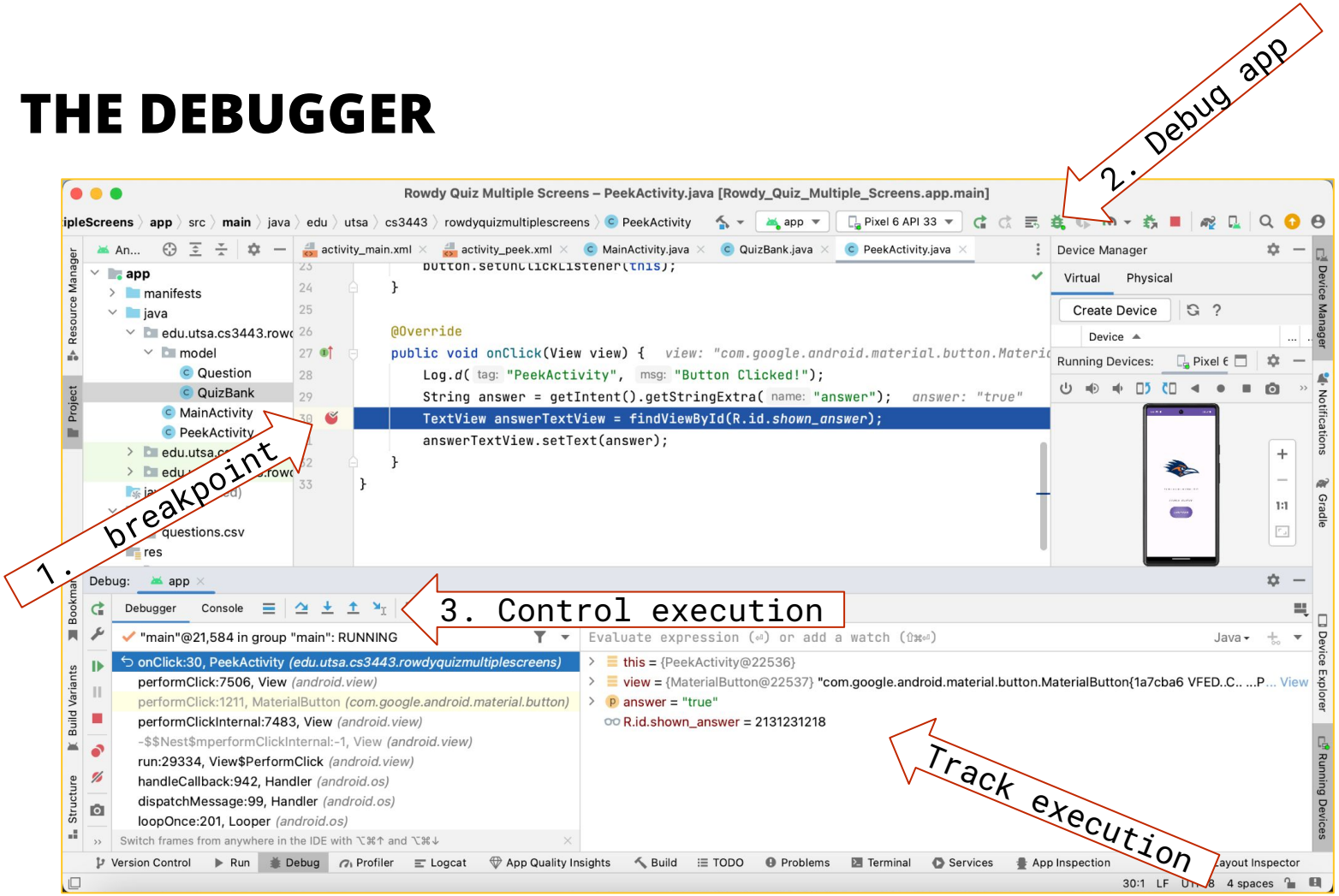
THE DEBUGGER

- The debugger is a tool mainly used to **run the application under controlled conditions** that permit the developer to track its execution and monitor changes in computer resources that may indicate malfunctioning code.
- To control the application execution, add **breakpoints** to your code lines.
- A breakpoint next to a line of code pauses the execution before the line executes and allows you to examine what happens next.
- To run the application in debug-mode, add breakpoint(s) then Run > Debug

THE DEBUGGER

- When using the debugger, the developer can control the app execution:
  - Step Into
    - A method is about to be invoked, and you want to debug into the code of that method, so the next step is to go into that method and continue debugging step-by-step.
  - Step Over
    - A method is about to be invoked, but you're not interested in debugging this particular invocation, so you want the debugger to execute that method completely as one entire step.
  - Step Return
    - You're done debugging this method step-by-step, and you just want the debugger to run the entire method until it returns as one entire step.
  - Resume
    - You want the debugger to resume "normal" execution instead of step-by-step

THE DEBUGGER

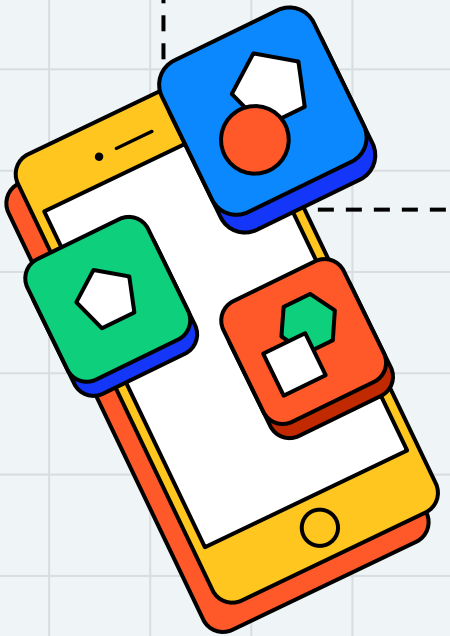


DEBUGGING ANDROID APPLICATIONS

- When errors persist, some things you can try
  - Recheck the xml in resource files
  - Clean the project build
    - Build > Clean Project
    - Android studio will rebuild the project from scratch
  - Clean the project
    - File > Invalidate Caches/Restart
    - Android Studio will perform some maintenance on the project and restart itself
  - Check online :)

CODE DEMO

- Show the different tools in Android Studio.



THANK YOU!

DO YOU HAVE ANY QUESTIONS?

hend.alkittawi@utsa.edu

By Appointment

Online