Recording in Progress

This class is being recorded

Please turn off your video and/or video if you do not wish to be recorded
CMSC436: Programming Handheld Systems
Android Development Environment
The Android Platform

A multi-layered software stack for building and running mobile applications
The Android Development Environment

Starts with knowledge of the Android platform
Your workbench for writing Android applications

See:

https://developer.android.com/studio/intro/
Today’s Topics

- Downloading Android SDK
- Using the Android Studio IDE
- Using the Android emulator
- Debugging Android applications
- Other tools
Prerequisites

Supported Operating Systems:
- Microsoft Windows 8/10 (64-bit)
- Mac OS X 10.14 (Mojave) or higher
- Any 64-bit Linux that supports Gnome, KDE, Unity DE
General Prerequisites

8GB RAM min

8GB memory for Android SDK, emulator system images, and caches

1280 x 800 min screen resolution
Getting Started

Download & install Android Studio

See: https://developer.android.com/studio/
Android Studio

Android platform

Android Studio IDE

Key development tools

System image for emulator
Hello World!
The Android Emulator

Runs virtual devices
// This helper method is called when user clicks the
// Show button and will start an activity to display
// a user specified location.

override fun processClick() {
    try {
        // Process text for network transmission
        val address = binding.txtSearch.text.toString()
        val intent = Intent(TravelActivity.ACTION_SEARCH)
        intent.putExtra(TravelActivity.EXTRA_ADDRESS, address)
        startActivity(intent)
    } catch (e: Exception) {
        Log.e(TAG, e.message)
    }
}
The Android Emulator

Pros

- Doesn’t require an actual phone
- Hardware is reconfigurable
- Changes are non-destructive
The Android Emulator

Cons

- Slower than an actual device
- Some features unavailable
  - e.g., no support for NFC, SD card insert/eject, etc.
- Performance / user experience can be misleading
Advanced Features

Can emulate many different device/user characteristics, such as:

- Network speed/latencies
- Battery power
- Location coordinates
- Sensor readings
Advanced Features

Ex: Change network speeds
Advanced Features

Ex: Emulate incoming phone calls & SMS messages
Advanced Features

Many more options

See:

Debugger

Tool for examining the internal state of a running application
The answer to life, the universe and everything is: 42
package course.examples.themanewer

class TheAnswer : AppCompatActivity() {

    companion object {
        private val answers = intArrayOf(1L, 8, 16, 32, 64)
        private val hint = "42"
        private val text = "TheAnswer"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)
    }

    val binding by binding.lifecycleScope

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)

        binding.button.setOnClickListener {
            val answer = binding.editText.toString().toInt() ?: 0
            if (answer == answers.first()) {
                // Handle correct answer
            } else {
                // Handle incorrect answer
            }
        }
    }
}

val layout: ActivityLayout = context?.layout

binding.button.setOnClickListener { binding.textInput?.text = hint }
Development Tools

Android Studio provides numerous tools for monitoring application behaviors
Example Tools

Device File Explorer
Logcat
Layout Inspector
CPU Profiler
Device File Explorer

View, copy, and delete files on your device

Often used to examine and verify file creation and transfer
package course.examples.thearn

class TheAnswer : Activity {
    companion object {
        private val answers = listOf(10, 20, 100, 1000)
        private const val answer = 42
        private const val text = "TheAnswer"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        // Required call through to Activity.onCreate()
        // Restore any saved instance state
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_thearn)

        val subject = findViewById<TextView>(R.id.textview)
        val output = if (answer == subject.text.toString()) 1 else 0
        outputTextView.text = output

        for (i in answers) {
            Log.d("TAG", "Trying $i")
            // Incorrect behavior
            return answers.firstOrNull { it == answer }
            // Correct behavior
            return answers.firstOrNull { it == answer }
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        Log.d("TAG", "Destroying view")
    }
}

The answer to life, the universe and everything is:

42
package course.examples.thekanswer

class Thekanswer : AppCompatActivity {
    companion object {
        private val answers = resources.getStringArray(R.array.answer)
        private const val answer = 4
        private lateinit var TVA : "Thekanswer"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        // super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main)
        val binding = AnswerFormBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.tvAnswer.text = answers[answer] + 1
        binding.tvQuestion.text = TVA
        binding.tvAnswer.text = binding.tvAnswer.text

        private fun findAnswer(): Int {
            Log.d("Theanswer", "Finding answer")
            // Incorrect behavior
            return answers[answer] + 1
        }
    }
}
Logcat

Write and review log messages
Apps use Log class to write messages to log
Developer can search and filter log messages
The answer to life, the universe and everything is:

42
The answer to life, the universe and everything is: 42
Layout Inspector

Shows the runtime organization of the user interface
The answer to life, the universe and everything is:

42
Next

Application Fundamentals
Example Applications

HelloWorld

TheAnswer