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!
package course.examples.helloworld

import android.app.Activity
import android.os.Bundle

class MainActivity : Activity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
The Android Emulator

Runs virtual devices
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 Bluetooth, USB connections, NFC, 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
Advanced Features

Change network speeds
Advanced Features

Emulate incoming phone calls & SMS messages
The Android Emulator

Can interconnect multiple emulators
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:

We may never know
class TheAnswer : Activity() {
    companion object {
        private val answers = intArrayOf(42, -10, 0, 100, 1000)
        private const val answer = 42
        private const val TAG = "TheAnswer"
    }

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

        // Set up the application's user interface (content view)
        setContentView(R.layout.answer_layout)
        val value = findAnswer()
    }
}
val output = if (value != null) answer.toString() else getString(R.string.never_know_string)

// Get a reference to a TextView in the content view
val answerView = findViewById<TextView>(R.id.answer_view)
// Set desired text in answerView TextView
answerView.text = output

private fun findAnswer(): Int? {
    Log.d(TAG, "Entering findAnswer()")
    // Incorrect behavior
    return answers.firstOrNull { it == -answer }
    // Correct behavior
    // return answers.firstOrNull { it == answer }
}
The answer to life, the universe and everything is:

42
Development Tools

Android Studio provides numerous tools for monitoring application behaviors
Example Tools

Device File Explorer
Logcat
CPU Profiler
Layout Inspector
Device File Explorer

View, copy, and delete files on your device

Often used to examine and verify file creation and transfer
```kotlin
override fun onCreate(savedInstanceState: Bundle?) {

    // Required call through to Activity.onCreate()
    super.onCreate(savedInstanceState)

    // Set up the application's user interface (content view)
    setContentView(R.layout.answer_layout)

    // Get a reference to a TextView in the content view
    val answerView = findViewById<TextView>(R.id.answer_view)
    val value = findAnswer()
    val output = if (value == answer) answerView.text else "We may never know"

    // Set desired text in answerView TextView
    answerView.text = output
}

private fun findAnswer(): Int {
    Log.d(TAG, msg = "Entering findAnswer()"

    // Incorrect behavior
    return answers.firstOrNull { it == answer }

    // Correct behavior
    return answers.firstOrNull { it == answer }
}
```
package course.examples.theanswer

import ...

class TheAnswer : Activity() {

    companion object {
        private val answers = intArrayOf(42, -10, 0, 100, 1888)
        private const val answer = 42
        private const val TAG = "TheAnswer"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        // Required call through to Activity.onCreate()
        // Restore any saved instance state
        super.onCreate(savedInstanceState)
        // Set up the application's user interface (content view)
        setContentView(R.layout.answer_layout)

        // Get a reference to a TextView in the content view
        val answerView = findViewById<TextView>(R.id.answer_view)
        val value = findViewById<TextView>(R.id.value)
        val output = if (value == answer) answer.toString() else "We may never know"
        // Set desired text in answerView Textview
        answerView.text = output
    }

    private fun findAnswer(): Int? {
        Log.d(TAG, msg = "Entering findAnswer()")
        // Incorrect behavior
        return answers.firstOrNull { it == answer }
        // Correct behavior
        return answers.firstOrNull { it == answer }
    }
}
Logcat

Write and review log messages
Apps use Log class to write messages to log
Developer can search and filter log messages
SuperclassAnswer

// Set up the application's user interface (content view)
setContentView(R.layout.answer_layout)

// Get a reference to a TextView in the content view
val answerView = findViewById(R.id.answer_view)
val value = findViewById()
val output = 
  if (value == answerView) answerView.text
else "We may never know"
// Set desired text in answerView TextView
answerView.text = output

private fun findAnswer(): Int {
  Log.d(TAG, msg "Entering findAnswer()"
  // Incorrect behavior
  return answers.firstOrNull { it == answer }
  // Correct behavior
  return answers.firstOrNull { it == answer }
}
// Set up the application's user interface (content view)
setContentView(R.layout.answer_layout)

// Get a reference to a TextView in the content view
val answerView = findViewById<TextView>(R.id.answer_view)
val value = findAnswer()
val output =
    if (value == answer) answer.toString() else "We may never know"

// Set desired text in answerView TextView
answerView.text = output

private fun findAnswer(): Int {
    Log.d(TAG, msg = "Entering findAnswer()")
    // Incorrect behavior
    return answers.firstOrNull { it == -answer }
    // Correct behavior
    return answers.firstOrNull { it == answer }
}
CPU Profiler

Logs execution sequences and timing taken from a running application

Graphically displays method traces and metrics
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class TheAnswer {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

        Collections.sort(numbers);

        System.out.println(numbers.toString());
    }
}

// Correct behavior
return answers.firstOrDefault();
package course.examples.theanswer

import ...

class TheAnswer : Activity()
{
    companion object {
        private val answers = intArrayOf(42, -10, 0, 100, 1888)
        private const val answer = 42
        private const val TAG = "TheAnswer"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_theanswer)

        val answerView = findViewById<TextView>(R.id.answerView)
        val valView = findViewById<TextView>(R.id.valView)
        val valInput = findViewById<EditText>(R.id.valInput)
        val answerInput = findViewById<EditText>(R.id.answerInput)
        val outputView = findViewById<TextView>(R.id.outputView)

        if (answers.contains(intArrayOf(answerInput.text.toString().toInt()))) {
            answerView.text = answerView.text.toString() + " Correct"
        } else {
            answerView.text = answerView.text.toString() + " Incorrect"
            outputView.text = "Val: $valView
            Input: $answerInput
            Val: $valInput"
        }
    }

    private fun find(val: String, Log-dTAG: String, Ment: String): String {
        if (val == Ment) {
            return "Correct"
        } else {
            return "Incorrect"
        }
    }
}

TheAnswer()

Run/Debug Configurations

Name: app
Share through VCS
Allow parallel run

Enable advanced profiling (required for API level < 26 only)
Allows the profilers to track data such as network payloads, application events and object counts, but it might have a minor performance impact on your build speeds.

Start recording CPU activity on startup
You must select Run > Profile from the main menu and deploy your app to a device running Android 8.0 (API level 26) or higher.

Toast Java Methods

Cancel    Apply    OK
package app

import java.util.ArrayList;

class TheAnswer {
    private ArrayList<Integer> answers = new ArrayList<Integer>(Arrays.asList(12, -10, 0, 100, 1888));

    public boolean isAnswer(Integer number) {
        return answers.contains(number);
    }

    public String getAnswer() {
        return answers.get(0).toString() + " We may never know";
    }

    public static void main(String[] args) {
        TheAnswer answer = new TheAnswer();
        System.out.println("Answer: " + answer.getAnswer());
    }
}

// Correct behavior:
// return answers.firstOrDefault(12);
Layout Inspector

Shows the runtime organization of the user interface
package com.example.theclass

import ...

class TheAnswer {

    companion object {
        val answers = mutableListOf(-10, 0, 100, 1000)
    }

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

        setContentView(R.layout.activity_theclass)

        // Get a reference to a TextView in the content view
        val answerView = findViewById<TextView>(R.id.answer_view)
        val value = findAnswer()
        val output = if (value == answer) answerView.text = output
        // Set desired text in answerView TextView

        private fun findAnswer(): Int? {
            Log.d(TAG, msg) "Entering findAnswer()"
            // Incorrect behavior
            return answers.firstOrNull { it == answer }
            // Correct behavior
            return answers.firstOrNull { it != answer }
        }
    }
}
The answer to life, the universe and everything is:

42
Next

Application Fundamentals
Example Applications

HelloWorld

TheAnswer