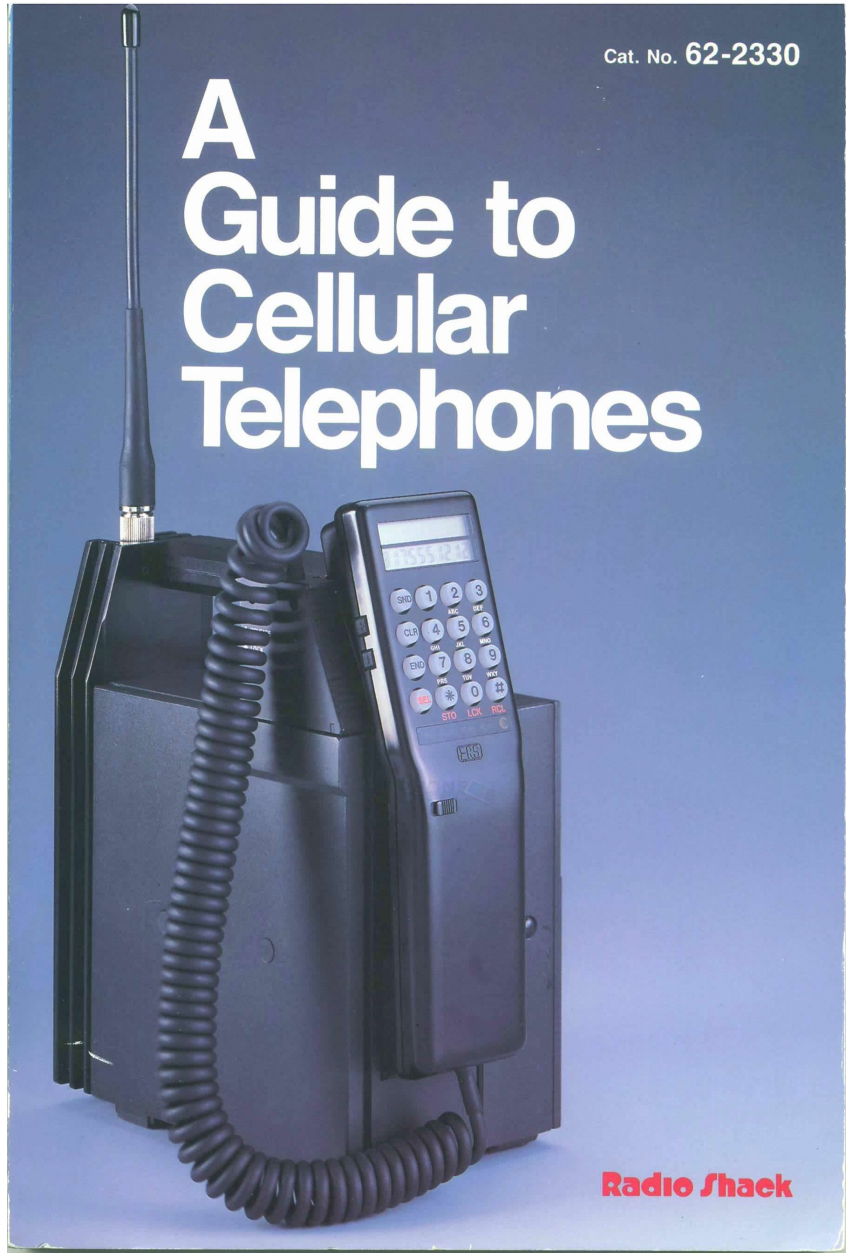
Mobile Application Development (COMP2008)

Lecture 1: Introduction

Updated: 25 July August, 2022

Discipline of Computing
School of Electrical Engineering, Computing and Mathematical Sciences (EECMS)


Cat. No. 62-2330
A Guide to Cellular Telephones
Radio Shack

Outline

Demo

- Introduction to Android Studio
- A simple Calculator

Please go to iLecture video recording to see the demo .

Everything Changes. Or Not.

- Everyone thinks that software development changes rapidly.
- This is (mostly) a fiction.
  - It's exciting to *believe* that things change rapidly.
  - . . . and that excitement/belief drives sales.
- There have been a few big innovations.
- But the key principles haven't changed in decades.

Are Mobile Devices Special?

- Perhaps they are just another "form factor".
  - One of many ways to bundle together computing hardware.
- *But...* what's different (from other computers) is their closeness to the user.
- Apps have access to:
  - You, almosts all the time, to send notifications.
  - The environment around you, through cameras, microphones, etc.
  - Highly structured personal data – your contacts, calendar, all your commmunications, your physical location, etc.
  - i.e. virtually your whole life.
- And yet, we have to work with a tiny UI.
  - Functionally, much like desktop GUIs.
  - But we must be much more careful with usability.

The Pre-Smartphone Era

- Mobile devices took a long time to get where they are now.
- The "Osbourne 1" in 1981 weighed 10.7kg.
  - Less portable than almost all computers today.
- The early days saw many non-networked mobile devices:
  - Standalone MP3 players.
  - Personal Digital Assistants (PDAs).
  - Required a cable to transfer data to/from your desktop PC.
  - Software was extremely limited, often to whatever was factory installed.

The Pre-Smartphone Era: Symbian

- Symbian is (was) a mobile phone OS.
  - Long outdated and now-discontinued.
- In its day, largely in the 2000's, it ran on the most widely-available phones, particularly Nokia's models.
- It provided Internet access via a simple web browser.
  - Networking and computing resources were limited at the time, though, so the result was quite basic.
- It allowed some degree of 3rd-party app development.
  - But writing and distributing apps wasn't made easy.
  - You had to use the "Organiser Programming Language" (OPL) – a bit like BASIC.
  - "Organiser" was later changed to "Open", but that neither helped nor made any sense.

~~Organiser~~ Open Programming Language

Take a moment to appreciate a language that (hopefully) you'll never have to use:

```
PROC test2:
    LOCAL a%
    a%=10
    WHILE a%>0
        PRINT "A=";a%
        a%=a%-1
    ENDWH
    PRINT "Finished"
    GET
ENDP
```

(Psion Computers, 1997, *OPL User Guide: Basics*, p. 22.)

The First iPhone

- The iPhone was the first “smartphone”.
  - Someone is bound to disagree, since the idea of a “smartphone” is a bit subjective.
  - But the first iPhone did change the way we thought about phones.
- Extensibility was the key.
  - Developers could easily write apps.
  - Users could easily install them.
  - More than its predecessors, the iPhone was a generalised computing device.
- The hardware also helped.
  - Memory and processing power was on the increase.
  - A capacitive touch screen interface replaced the traditional tiny phone keypad.

iOS

- The first iPhone ran "iPhone OS", later renamed to "iOS".
- iOS is different from MacOS (Apple's desktop OS).
- But the two are related:
  - iOS and MacOS are both based on Darwin.
  - . . . which is based on NextStep.
  - . . . which is based on BSD (Berkeley Software Distribution).
  - . . . which is a UNIX variant, *like* Linux, but developed independently.
- The initial language of choice was Objective C:
  - An OO extension to C, but *not* C++.
  - Showing its age.
- The "Swift" language is now taking over.

Android (vs iOS)

- Though Apple's iOS blazed the trail, Google's Android has long since caught up.
  - The market share of iOS vs Android differs a lot across: (a) countries, (b) years, AND (c) people who do the measuring!
  - In some cases, iOS controls 80% or more of the market. In other cases, Android controls 80% or more.
- But Android is not (quite) an iOS clone.
  - Different languages and APIs.

Android

- Android is based directly on Linux.
- Development is typically done in Java.
  - C++ and Kotlin are also officially supported.
  - Other languages are *possible*, but rely on third-party support.
- More "open" than iOS.
  - Google drives Android's development, but it's open source.
  - Any mobile device maker can use it, and almost all do (other than Apple).
  - Google's revenue comes from advertising, so openness is to its advantage.

Windows (Dead!!)

- Microsoft has long has its own mobile OS.
  - Or rather a lineage of them.
  - Windows CE.
  - Windows Mobile.
  - Windows RT.
  - Windows 10 S.
- It attempted to break the iOS-Android strangehold.
  - For a time, it was possible to buy smartphones loaded with Windows.
  - Never quite reached "critical mass" – not enough app development to give consumers a reason to join up.
  - Seems to be abandoned, for now.
- Microsoft now appears to be rebuilding its software *on top* of Android and iOS, instead of competing with them.

The Web

- Web applications are (sometimes) under-appreciated way to develop software for mobile devices.
- The user doesn't need to install or update anything.
- Unmatched platform independence. A well-designed web app will run on Android *and* iOS *and* Windows *and* Linux *and* MacOS, etc.
- You can reproduce most of the mobile app "look-and-feel" in a web app.

The Web: On the Other Hand...

- You may not get quite the same performance.
- A web app won't quite be a "first-class citizen" on mobile devices.
  - It won't have a launch icon.
  - It won't be able to interact with other apps quite as easily.
- You need a web-server, and you need Internet access just to run the app.
- Web development is *still* the "Wild West" – there are millions of different ways to do it, and change is very rapid.

Compatibility Layers – Hybrid Mobile Apps

- Generally, Android apps don't run on iOS, and vice versa.
- *Unless...* you use a third-party compatibility layer.
- Apache Cordova (cordova.apache.org)
  - Write Android+iOS apps using web technologies: HTML5, CSS, JavaScript.
  - Foundation layer for "Ionic" (ionicframework.com) and other frameworks.
- Xamarin (visualstudio.microsoft.com/xamarin)
  - Write Android+iOS+Windows apps using C# and .NET.
- Pro: your app has a single unified codebase.
  - Not separate ones for iOS and Android.
- Con: slower, bigger and (slightly) more fault prone.
  - Unavoidable due to increased complexity.

Mobile Development: What *Is* Different?

- On one hand, mobile development is just programming.
- On the other hand, there are several differences from what you (may) have encountered so far.
- The OS is different.
  - Welcome to Android and iOS
- The dev environment is separate from the testing/production environment.
- Distribution is more tightly controlled.
- Security, usability, and power consumption are much more important than usual.
- You can do more interesting things!

Mobile OS's and Frameworks

- iOS and Android are both based on UNIX (BSD/Linux).
- But it won't look much like it!
- You'll have to get used to the idea of a "framework".
  - Frameworks are like libraries/APIs, but they help define the structure of your application.
- You won't be writing "`main()`" methods anymore.
- Instead, you'll be writing event handlers.
  - In particular, you inherit from framework classes and override their methods.
- It can feel like frameworks take away a lot of control.
  - Ultimately that control is still there – there's just a lot of highly sophisticated "defaults".

Android Activities

```
public class MyActivity extends AppCompatActivity
{
    @Override
    protected void onCreate(Bundle savedState)
    {
        super.onCreate(savedState);

        // This is where your code starts!
    }
}
```

The Write-Compile-Run Cycle

- So far, most of your Java development will look like this:

```
[user@pc]$ javac MyCode.java
[user@pc]$ java MyCode
```

- You can't (really) write and test your mobile app on the same device.
  - Android and iOS are good at *running* mobile apps, but not for *writing* them in the first place.
- You *write* your code on Linux, MacOS or Windows.
  - Most likely using Android Studio or Xcode.
- You *run* your code on an emulator/simulator.
  - Essentially a virtual machine running Android or iOS.
  - Or on a real physical device.

Distribution

- iOS and Android tightly control the means of distribution.
- You can't just tell people to download your app from a website.
  - . . . Or sell it to them on a 3.5" floppy disk.
  - Well, you *can* but. . . ugh.
- Vast majority of users expect to find apps in Google Play or the App Store.
  - Very tightly controlled on iOS in particular.
  - But most people even on Android aren't going to look elsewhere.
- There is an approval process.
  - You cannot "just upload" your app to these locations.

Security

- Mobile devices are easily lost or stolen.
- They deal with more hostile environments.
  - Wifi hotspots, bluetooth and NFC are all opportunities for attackers to compromise your device.
  - These issues generally don't exist (directly) for desktop/server computers.
- They tie together a lot of personal data.
  - Your phone knows *a lot* about you – your accounts, passwords, contacts, calendar appointments, your physical location, etc.
- Nobody pays for security. (And so we pay for insecurity!)
  - Phone manufacturers make money from sales, not providing security fixes.
  - Even critical security fixes can take take a long time to become available on some devices, and may *never* be available on others.
  - Low-end devices are particularly problematic.

Usability

- The mobile experience is quite different from the desktop one.
- The screen is generally smaller.
  - Human eyesight is a limiting factor.
  - Human finger size is another one!
- Users want to use your app with just their thumb while simultaneously eating a kebab, walking down the street, listening to music, and talking to friends.

Power Usage

- Your users will hate you for draining their batteries.
- CPU usage and network traffic are the main culprits.
- Mobile CPUs can be powerful, but you *should not* have long-running algorithms.
  - You might be able to "hand off" an expensive task to a server machine, and just retrieve the result.
- But don't connect to a server too frequently either.
  - The wireless transmitter consumes a lot of power too.

Functionality

- Mobile apps can use a variety of sensors:
  - Cameras, accelerometers.
- Mobile apps have access to structured personal information:
  - Contacts.
  - Calendar events.
  - Physical location.
  - While these have security issues (as mentioned), they are also an opportunity.

Android App Architecture

- Each app runs as a separate process (like any program).
- But app *also* run under separate UIDs (user IDs).
  - Traditionally, a UNIX user only has *one* UID.
  - Separating UIDs helps protect apps from each other.
- Android apps are made of (mostly) "activities".
  - Activities can be started and stopped independently of one another.
  - *But* an app's activities, when running, are all part of the same program.

User Interface

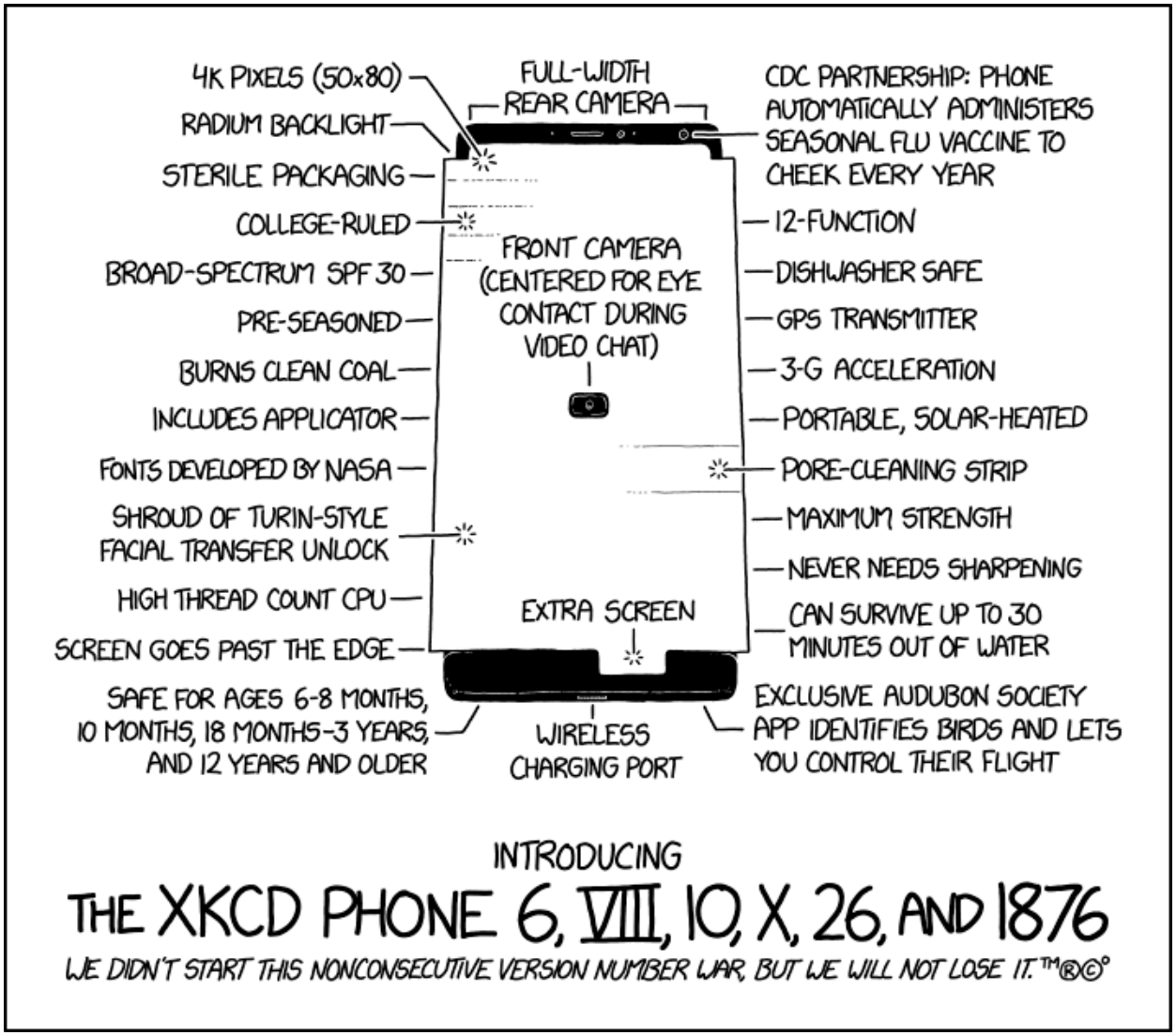
- You'll design your UI dragging/dropping GUI elements.
  - This is how most GUI design happens in practice (on both desktop and mobile platforms).
- Your UI design is stored in a series of XML files.
- These are auto-magically converted into a working UI when your app runs.
- You connect your UI to your activity(ies) by defining event handlers.

More As We Go

- The first worksheet covers a lot of nitty-gritty detail.
- And we'll explore more of it in the near future!

Something for you to explore!!

1. Create your Github Account
2. Link your android studio with Github
3. Create your Git Repository and Android project
4. Make sure your project is being pushed to Github



(https://xkcd.com/1889/)