CS683 Mobile Application Development

Course Overview

Yuting Zhang
BU CSMET

Welcome to CS683

- Course website: onlinecampus.bu.edu
- Google Drive
- Github classroom

  - Syllabus,
  - Lecture slides
  - Online lecture notes
  - Lab exercises
  - Project assignments
  - Discussion board
  - References

References

- Official Android Developer website: https://developer.android.com
- Textbooks:

O'REILLY
Third Edition
Head First
Android Development
A Learner's Guide to Building Android Apps with Kotlin
Dawn Griffiths & David Griffiths
A Brain-Friendly Guide
HOW TO BUILD ANDROID APPS WITH KOTLIN
A hands-on guide to developing, testing, and publishing your first apps with Android
Packt>
Alex Forrester | Eran Boudjnah
Alexandru Dumbravan | Jomar Tigcal
3RD EDITION
Android Programming
THE BIG NERD RANCH GUIDE
Bill Phillips, Chris Stewart, and Kristin Marsicano
Android Studio 4.0 Development Essentials
Kotlin Edition
o 4.0
nt
Java Edition

References

- Android development official website:
  - https://developer.android.com/
- Textbook code github entry:
  - https://github.com/dogriffiths/HeadFirstAndroid3rdEd
  - https://github.com/PacktPublishing/How-to-Build-Android-Apps-with-Kotlin/

Maximize Your Learning Outcomes

- Course Features:
  - A programming oriented course
    - Reading and writing code
  - A software engineering approach
    - Develop high quality software
  - Project centered
    - A self-defined project is splitted into 6 project assignments
    - A predefined small project is splitted into 5 labs
  - Class participation
    - discussions, coding exercises
    - Student Presentations
- Your success is my success!

Questions?

CS683 Mobile Application Development with Android

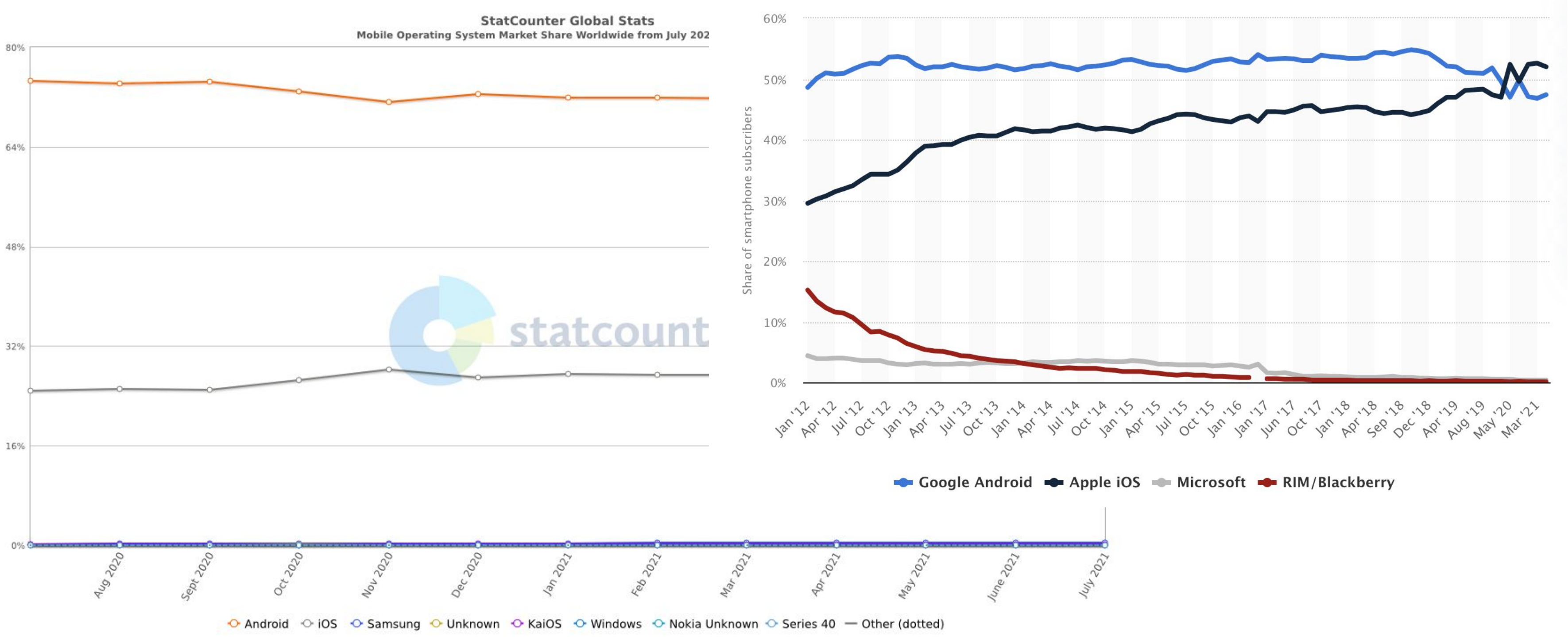
Introduction to Mobile Computing and Android

Yuting Zhang
BU METCS

Mobile Devices

- Tablets, mobile phones (feature phones, smartphones), wearables, PDAs, etc.
- Enhanced hardware and software capabilities, and advanced communication capabilities
- From single-purpose (communication) to general-purpose (process, storage, communication)
- Store and transmit personal and corporation information, use for online transactions, etc
- Features: mobility, connectivity, battery-powered

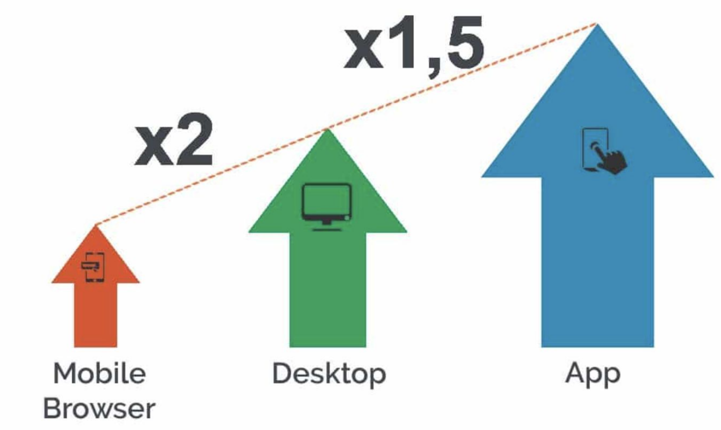
Statistics: Mobile OS Market Share



© Statista 2021

Poll

- Do you have any Android device or iOS device?
- How many hours do you spend daily on your mobile devices (such as smartphones)?
- Which types of apps do you use in a daily basis? Which types of apps do you use most?
- Name a good app that you like and a bad app that you dislike, and state why?



https://jmango360.com/mobile-app-vs-mobile-website-statistics/

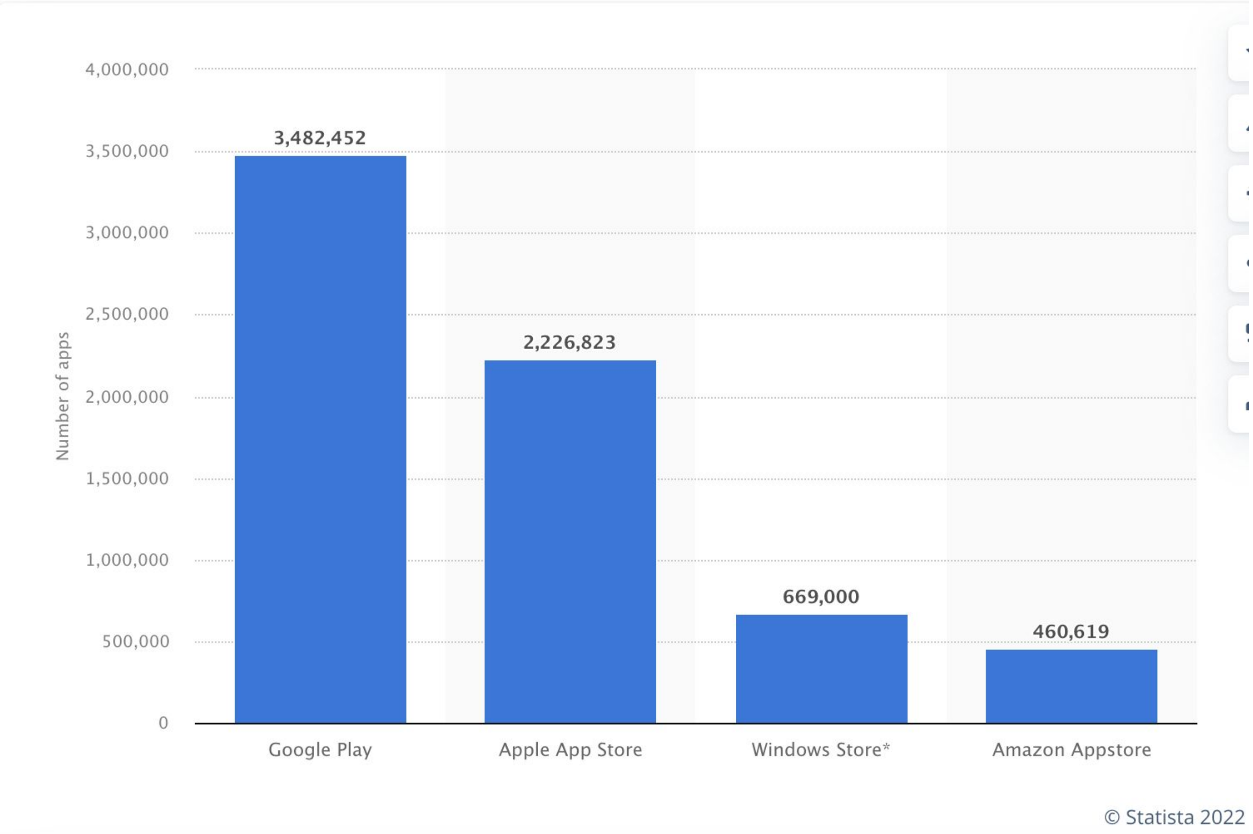
Applications

- Apps shipped with the mobile platform
- Apps installed by the manufacturer
- Apps installed by the wireless carrier
- Additional pre installed apps
- Apps installed by the user from the official market or the third market

Mobile Application Categories

- Apps
  - Books, Business, Education, Entertainment, Finance, Health&Fitness, Medical, Music & Demo, News & Magazine, Parenting, Social, Tools, Weather...
- Game
  - Action, Adventure, Arcade, Board, Card, Casino, Casual, Educational, Music, Puzzle, Racing, Role Playing, Simulation, Sports, Strategy, Trivia, Word.

Apps in Major App Stores



© Statista 2020

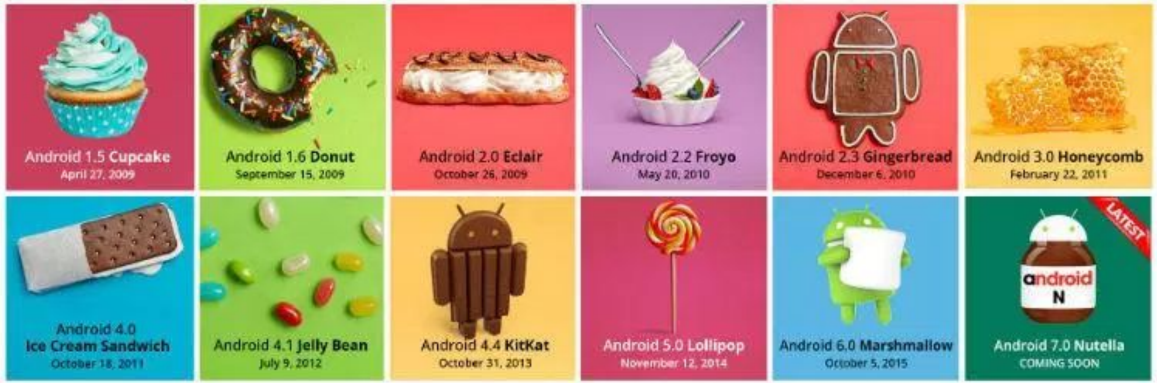
https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

Android History

- Android Inc. was founded by Andy Rubin, Chris White, Nick Sears and Rich Miner in Oct 2003 in Palo Alto, CA.
- Google acquired Android Inc in Aug, 2005.
- OHA (The Open Handset Alliance) was announced in Nov 2007.
  - It is a consortium of technology companies including google, HTC, samsung, T-mobile, Qualcomm, etc.
  - The goal is to develop open standards for mobile devices.
- Google released the early version of Android SDK (Software Development Kit) in 2007 (one week later).
- Android Open Source Project (AOSP) was officially released in Oct 2008.
- The first publicly available Android smartphone, the T-Mobile G1 was also released in Oct. 2008.

Android Versions

- Check this page: https://developer.android.com/about/dashboards/index.html
- The latest version?
- Mapping between version numbers, code names, API levels?
  API 23 -> version ? -> code name?   Android O -> version? -> API ?
  Android 10 - > API ?



Android 10

Openness?

- AOSP (Android Open Source Project): https://source.android.com
- Stated goal: innovation and openness
- Linux kernel uses GPLv2, and most of the android platform source code uses Apache 2.0 license.
- Customer devices do contain some closed source software components, e.g. boot loaders, firmware, DRM, and apps.
- Source code may not be made available at the release time. e.g. Ice-cream Sandwich source code is released a month after the release date.
- Discussion :Openness vs. Security (how do they relate to each other)?

Issues

- Fragmentation
- Compatibility
  - Android compatibility definition document: http://source.android.com/compatibility
  - Android Compatibility Test Suite (CTS): https://android.googlesource.com/platform/cts/
- Update Issues
  - App update: can be deployed directly by developer.
  - OS update though firmware/OTA update: more complicated, slower
  - Almost no back-porting (apply current fix to the older version)

Mobile Security Challenges

- Mobility: easily stolen or physically tampered
- Strong personalization: usually, the owner of device is also its unique user
- Strong connectivity (always connected through wifi, cellular network, bluetooth ..)
- Technology convergence: a single device combines different technologies, different routes to perform attacks
- Reduced capabilities and limited resource (such as CPU, memory, battery limit the sophisticate security mechanism)

C. R. Mulliner, “Security of Smart Phones,” Master’s thesis, University of California, Santa Barbara, 2006.

Android Stakeholders

- Google:
  - Performs legal administration.
  - Owns and manages the Android brand.
  - Relates to the software and hardware infrastructure; runs google play; supports AOSP (gmail, calendar, contacts, ….)
  - Oversees the development of the core Android platform.
  - Fosters AOSP as an open source project. Provides SDK, API documentation, source code, style guidance and more.

Android Stakeholders

- Hardware Vendors
    - CPU manufacturers (ARM, Intel (Android on Intel Architecture Project), MIPS)
    - System-on-chip (SOC) Manufacturers
    - Device Manufacturers: ODM (original design manufacturers) and OEM (original Equipment manufacturers)
- Carriers: customization, QA testing, etc
- Developers: (google or third party) platform developers, application developers, custom ROMs developers
- Users: general users, power users, researchers

Questions?

Android Applications

- Developed in Java/Kotlin and executed in VM.
- ***AndroidManifest.xml***: package and version info, components def, permission def, external lib info, shared UID info, etc.
- **Activity**: UI components
- **Service**: components running in the background.
- **Content Provider**: a structured interface to common shared data stores. Usually backed by a SQLite database or a direct file system path.
- **Broadcast receiver**: define what types of broadcast to receive, and the criteria of the sender.
- ***Intent:*** message objects for inter-component communication (ICC)
- http://developer.android.com/guide/index.html

Manifest Example

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="edu.bu.metcs.cs683example">
   <uses-permission android:name="android.permission.INTERNET" />
   <application
       android:allowBackup="true"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:roundIcon="@mipmap/ic_launcher_round"
       android:supportsRtl="true"
       android:theme="@style/Theme.MyApplication">
       <activity android:name=".Activities.MyActivity"
           android:exported="true">
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name=
                   "android.intent.category.LAUNCHER"/>
           </intent-filter>
       </activity>
```

Manifest Example

```
         <provider
               android:name=".ContentProvider.MyContentProvide"
               android:authorities=".ContentProvider.MyContentProvide"
               android:permission="My_PERMISSION"/>
           <receiver android:name="Receiver.MyReceiver"
               android:exported="true">
               <intent-filter>
                   <action android:name=
                       "android.intent.action.BATTERY_LOW"></action>
               </intent-filter>
           </receiver>
           <service android:name=".Service.MyService"/>
       </application>
   </manifest>
```

SDK and NDK

- Software Development Kit (SDK)
    - Includes software libraries and APIs, reference materials, emulator and other tools. (http://developer.android.com/sdk/index.html)
    - AVD (Android Virtual Device) & ADB (Android Debug Bridge)
        - sdk/tools/android, sdk/platform-tools/adb
    - SDK directory: (e.g. ~/Library/Android/sdk in Mac)
- Native Development Kit(NDK)
    - It allows developers to write code in C/C++ and compile directly for the CPU. (http://developer.android.com/tools/sdk/ndk/index.html)
    - C/C++ components are packaged inside the application's .apk file, and run within the VM.

ADB

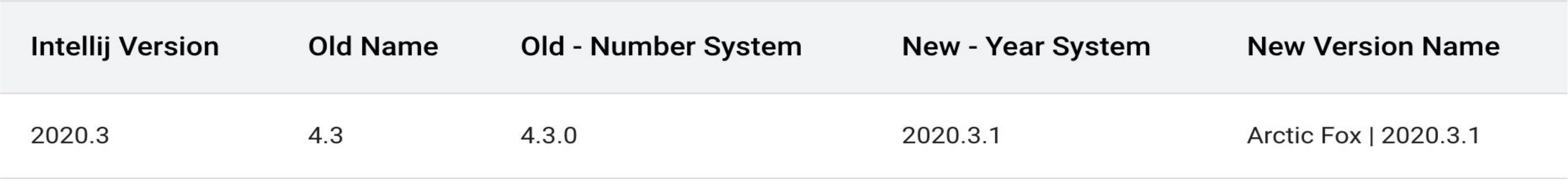
- ADB (Android Debug Bridge) (<*android_sdk*>/platform-tools/adb)
  - A command line tool to enable the communication between with the android device and the computer.
  - Need enable the USB debugging option.
  - Include three components:
    - A daemon program adbd running on the android device,
    - A service program running on the computer
    - A client program running on the computer
  - ADB commands:
    - adb shell
    - adb install/ adb uninstall
    - adb backup/ adb restore
    - Commands used within an adb shell: am command/pm command
  - https://developer.android.com/studio/command-line/adb.html

AVD

- AVD (Android Virtual Device/emulator)
    - A configuration that defines the characteristics of an Android phone, tablet, Android Wear, or Android TV device that you want to simulate in the Android Emulator. Create and manage AVDs
        - Open the AVD Manager (GUI) from Android Studio (Tools>Android>AVD Manager)
        - When an AVD is created, a folder named .android is created (by default it is in your home directory, e.g. ~/.android/avd/), which holds all configuration and data files needed to run each avd (.avd, .ini).
        - https://developer.android.com/studio/run/managing-avds.html

Android Studio

- https://developer.android.com/studio
- The official IDE for Android Application Development based on IntelliJ IDEA.
- It includes:
  - Powerful code editor,
  - Android SDK and other tools
    A fast and feature-rich emulator
  - A flexible Gradle-based build system
  - C++ and NDK support
  - A unified environment for you to develop for all Android devices
  - Instant Run to push changes to your running app
  - Code templates and GitHub integration
  - Extensive testing tools and frameworks
  - Lint tools

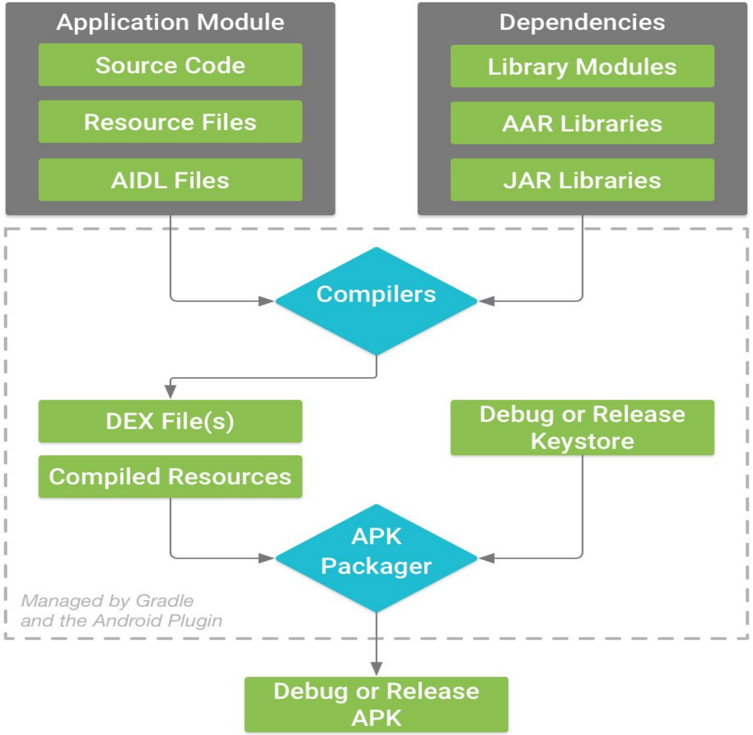


Intellij Version	Old Name	Old - Number System	New - Year System	New Version Name
2020.3	4.3	4.3.0	2020.3.1	Arctic Fox | 2020.3.1

Gradle Build System

- Gradle is an advanced build toolkit to automate and manage the build process, while allowing you to define flexible custom build configurations. The goal is faster, powerful and customizable.
  - https://gradle.org/
  - Like Ant & Maven
  - Scalable and extensible with powerful dependency management
  - Groovy build scripts
- The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications.
- Gradle and the Android plugin run independent of Android Studio.

Compilation Process

- https://developer.android.com/studio/build
- 

Lint

- It is a code scanning tool to check your Android project source files for potential bugs. It also provides you correction suggestions.
- The report includes a description message and a severity level for each problem identified.
- It can be configured in Android studio to run whenever build your app. It can also run manually from the command line.

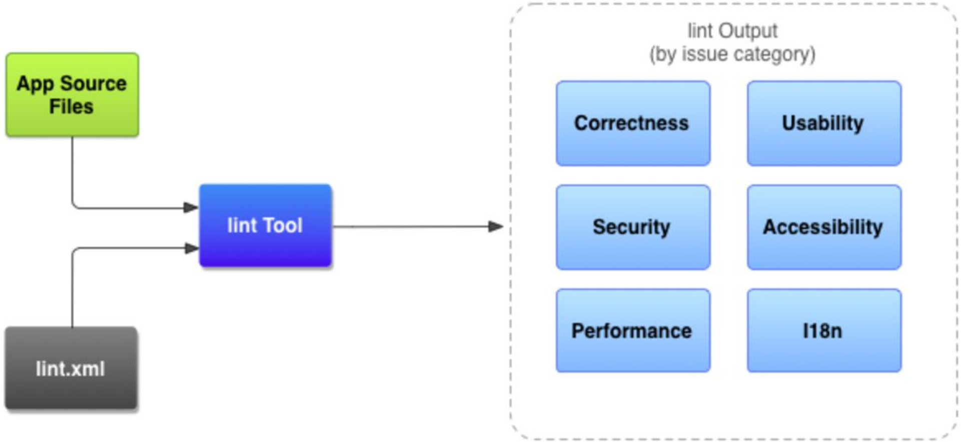
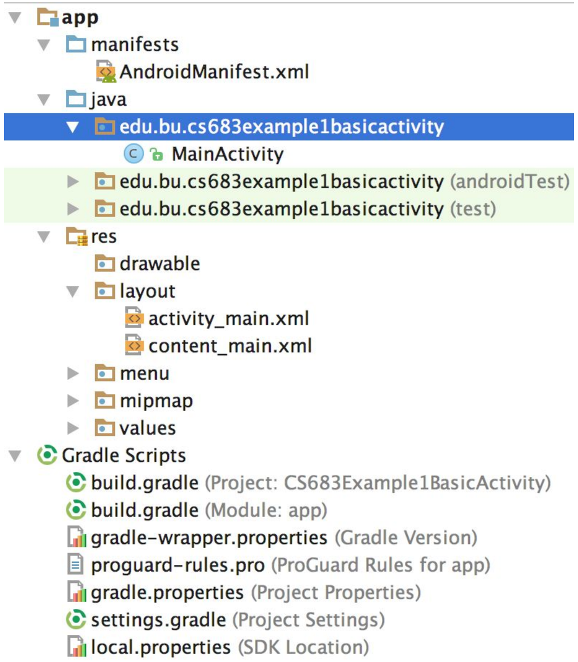


Figure 1. Code scanning workflow with the lint tool

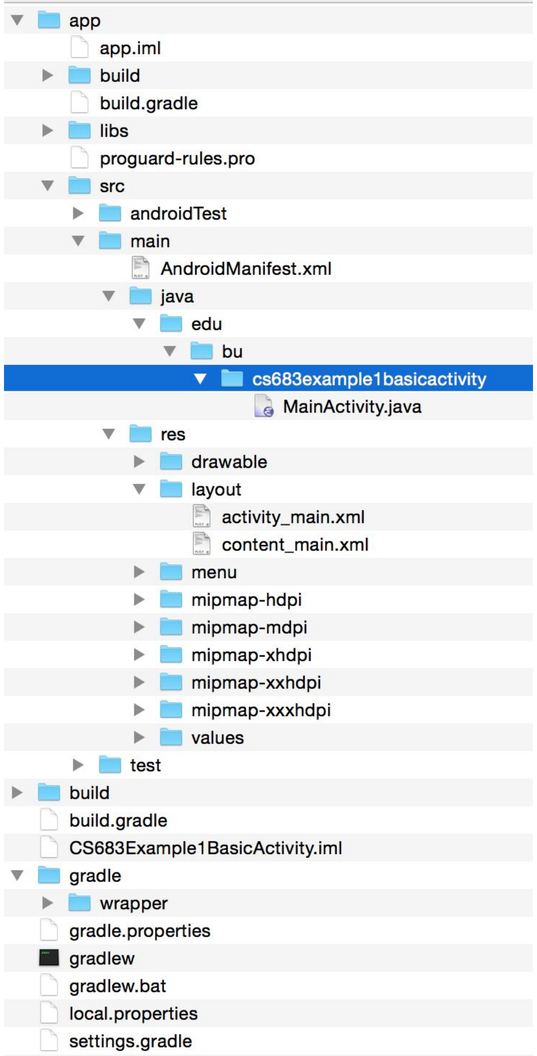
- https://developer.android.com/studio/write/lint

Project Structure

- Project Structure in AS view
- Project Folder Structure in File System view



```
▼ app
    ▼ manifests
        AndroidManifest.xml
    ▼ java
        ▼ edu.bu.cs683example1basicactivity
            MainActivity
        ▶ edu.bu.cs683example1basicactivity (androidTest)
        ▶ edu.bu.cs683example1basicactivity (test)
    ▼ res
        drawable
        ▼ layout
            activity_main.xml
            content_main.xml
        ▶ menu
        ▶ mipmap
        ▶ values
▼ Gradle Scripts
    build.gradle (Project: CS683Example1BasicActivity)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Version)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```



```
▼ app
    app.iml
  ▶ build
    build.gradle
  ▶ libs
    proguard-rules.pro
  ▼ src
      ▶ androidTest
      ▼ main
          AndroidManifest.xml
          ▼ java
              ▼ edu
                  ▼ bu
                      ▼ cs683example1basicactivity
                          MainActivity.java
          ▼ res
              ▶ drawable
              ▼ layout
                  activity_main.xml
                  content_main.xml
              ▶ menu
              ▶ mipmap-hdpi
              ▶ mipmap-mdpi
              ▶ mipmap-xhdpi
              ▶ mipmap-xxhdpi
              ▶ mipmap-xxxhdpi
              ▶ values
      ▶ test
▶ build
  build.gradle
  CS683Example1BasicActivity.iml
▼ gradle
  ▶ wrapper
  gradle.properties
  gradlew
  gradlew.bat
  local.properties
  settings.gradle
```

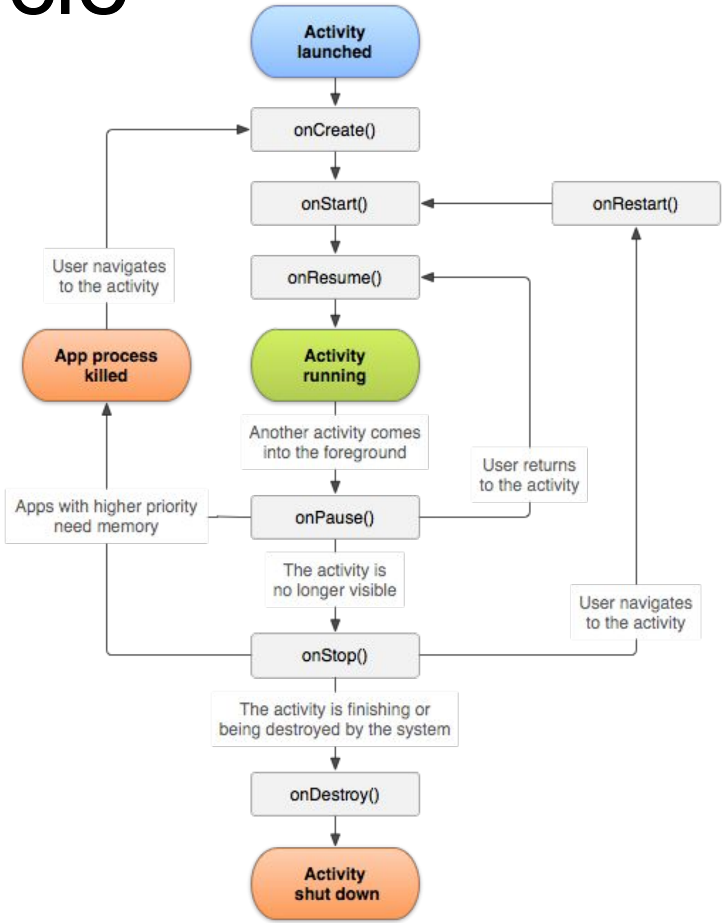
Questions?

Hello World Example

- Demo
- Choose a proper minimum SDK version
- Create an activity using a template:
  - The xml layout file: activity_main.xml
  - The java file: MainActivity.kt
  - The resource folder res/ and the R class
- The Manifest file: AndroidManifest.xml

Activity LifeCycle

- Foreground (most important)
  - onCreate() (created)
  - onStart() (started)
  - onResume() (resumed)
- Visible at Background
  - onPause() (paused)
- Invisible at Background
  - onStop() (stopped)
- Empty
  - onDestroy() (destroyed)


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Lab 1

- Create, build and run the simple “Hello World!” example
- Add the log information into each activity callback functions
- Use logcat to review log messages
- Perform different actions such as open/close the app, bring it background/foreground, start another app, press the back button, press the home button, press the overview button, etc and record how those callbacks are called in these scenarios.
- Use Git/Github together with Android studio to commit, pull and push changes
- Complete the lab report. You can choose to use the markdown file for documentation.
- https://github.com/PacktPublishing/How-to-Build-Android-Apps-with-Kotlin/blob/master/Chapter02/Exercise2.01/app/src/main/java/com/example/activitycallbacks/MainActivity.kt

Questions?

Android Hardware Platforms

- Device Types: smartphones, tablet, netbook, TV, watch, glasses, and others (automobiles, copy machine/printer, washing machines, microwave, etc.)
- CPU: mainly **ARM** family, also support x86, MIPS, and NEON
  - What is ARM?(https://developer.arm.com/architectures/cpu-architecture)
  - # of cores (dual-core/quad-core/multi-core), 32/64 bits
  - Speed: GHz (e.g. 2GHz)
- Random Access Memory (RAM): GB (e.g. 3GB, 4GB, 8GB), volatile
- NAND-flash memory: GB (32 - 256GB), nonvolatile
  - Operations: read, write, and erase. Can only be written once.
  - built-in memory
  - External SD card (slots)
- SoC (System on a Chip): CPU, GPU, memory, storage, etc.

Android Hardware Platforms

- Screen (size: 4” - 10’’) , keyboard,
- Camera, microphone/speaker
- SIM card
- Wireless communication technology:
  - Baseband Modem/Radio
  - Wifi (IEEE 802.11 a/b/g/n/ac): WEP, WPA (WLAN)
  - Bluetooth (~10m), NFC (Near Field Communication) (4cm), IrDA
- Positioning System
  - In USA, GPS (Global position system) (receiver)
  - Outside USA, Global Navigation Satellite System (GNSS) and its various flavors: GPS, GLONASS, Galileo or Beidou
- Battery: # of hours
- Other sensors: accelerometer/gyroscope …

Android OS Architecture

- Layered Architecture
  - Linux Kernel: binder, ashmem, pmem, logger, wakelocks … (http://elinux.org/Android_Kernel_Features)
  - Middle layer: native code support, hardware abstraction layer, runtime library, runtime virtual machine environment, system services, framework
  - Apps: pre-installed (by google, OEM, carrier) (/system/app) and user-installed apps (through google play, or direct downloaded and manually installed)(/data/app)
- https://developer.android.com/guide/platform/index.html

Kernel

- Various kernel versions for different device models. (https://android.googlesource.com, https://source.android.com/source/building-kernels.html)
- Linux-based kernel: provide process-based isolation (CPU, memory, process management, etc)
  - Android 1.5 is based on Linux 2.6
  - Android 7 is based on Linux 4.4
  - http://android.stackexchange.com/questions/51651/which-android-runs-which-linux-kernel

Kernel

- Additional Drivers:
  - Binder: facilitate IPC mechanism
  - ashmem (Anonymous shared memory): file-based reference-counted memory interface for low-memory environment
  - pmem: manage large, physical contiguous memory.
  - Viking Killer: OOM killer that implements Android's "kill least recently used process" logic under low memory conditions.
  - Logger: support additional logging subsystem
  - Paranoid networking: restrict certain networking features to specific group IDs
  - Additional functionality:
    - wakelocks: power management feature to keep a device from entering low power state and staying responsive
    - yaffs2: support for yaffs2 flash file system

Virtual Machine

- Dalvik: (in earlier versions)
  - Register-based VM, uses JIT (Just-In-Time) runtime
  - Use the DEX (Dalvik Executable) format (all class files combined into a DEX file loaded and interpreted by the Dalvik VM.)
  - Try to reduce the memory footprint and optimize memory usage.
- ART (Android Run Time) (starting in Android 4.4)
  - Compatible to run DEX bytecode
  - AOT(Ahead of Time) compilation: improve app performance (faster)
  - Improved garbage collection
  - Enhanced performance and battery efficiency
  - Android 7 adds a JIT compiler with code profiling to ART to improve the performance

Android Framework

- Provides a set of APIs to develop apps for android devices.
- Developed in Java/kotlin and executed in VM.
- Framework managers: activity manager, view system, package manager, telephone manager, resource manager, location manager, notification manager

Android Libraries

- Android Libraries included in the Android Framework
  - android.app
  - android.content, android.database, android.provider
  - Android.text, android.view, android.widget
  - Android.util, android.os, android.hardware
  - Android.graphics, Android.opengl
  - Android.media
  - Android.net
  - ...

https://developer.android.com/reference/packages.html

Android JetPack

- Jetpack is a suite of libraries to help developers
  - follow best practices,
  - eliminate boilerplate code,
  - Reduce fragmentation
- Launched in 2018, including existing android support libraries, android architecture components, Android KTX library.
  - Fragment, ViewModel, LiveData, Room, WorkManager, Compose
- AndroidX replaces original Android Support Library to provide backward compatibility across Android releases. It is separately maintained and updated.

Questions?

Android Application Development

- A semester long self-defined project to develop a real world android application
- Several labs to develop a simple demo application
- Use a software engineering approach
- Use Github
- Learning Android APIs, programming concepts and software development

SDLC

- Software Development Life Cycle
- Activities:
    - Inception: research similar systems; define the scope, major functionality, and targeted customers of the software system to be developed
    - Planning: estimate the cost (in terms of man hours), define work items, schedule, resources, and high-level activities; perform feasibility study and also plan configuration management
    - Requirements Analysis: gather and analyze functional and nonfunctional requirements
    - Software Design: design the software architecture and design detailed classes

SDLC

- Software Development Life Cycle
- Activities
  - Implementation: coding, debugging, refactoring
  - Testing: include unit-testing, integration testing, and system testing
  - Maintenance: defect repair and quality enhancement
  - Project Management: perform throughout the life cycle of a project; include risk management, quality management, configuration management, etc.

Software Engineering Process

- Software Engineering is a collection of techniques, methodologies and tools that help with the production of
  A high quality software system developed with a given *budget* before a given *deadline* while *change* occurs
- Challenges: Dealing with both complexity and constant change
- Process: Framework for carrying out the activities of a project in an organized and disciplined manner.
  - Use an iterative and incremental process. (e.g. Agile)

SDLC Models

- Sequential: waterfall
- Iterativative: spiral, Unified process, scrum, etc.
  - Parts of or all activities are repeatedly conducted in order to cope with change and provide more feedback.
- We will use an iterative model in our projects:
  - Iteration 0: planning
  - Iteration 1 - 5: each iteration includes requirement analysis, design, implementation, and testing

Agile

- Agile Manifesto:
    - ***Individuals and interactions over processes and tools***
    - ***Working software over comprehensive documentation***
    - ***Customer collaboration over contract negotiation***
    - ***Responding to change over following a plan***
- Agile models:
    - Iterative and incremental
    - Short iterations to get quick feedback and thus more adaptive
- Agile Frameworks:
    - Scrum, Extreme programming, etc
- Agile Practices:
    - User stories, constantly refactoring, TDD, continuous integration,

Requirement Analysis

- High level requirements
    - Essential
    - Desirable
    - Optional
- User Stories
    - Used in most agile framework to describe high-level functional requirements
    - Concise, specific, written with and/or by customers
    - Roles + functionalities/features + value
    - The INVEST principles:
        - Independent, negotiable, valuable, estimable, small, testable,

User Stories

- Title, a short description (may use some template)
  - As a (role), I want to (some feature), so that (value)
  - View all Projects: As a user, I want to view all projects in the system, so that I can find some projects that I am interested.
- Acceptance tests
  - Define the “definition of Done” (the criteria to meet to be accepted)
  - Given (setup/precondition), when (input/action), then (expected result)
  - Given the app is successfully installed, when I clicked on the app icon, then I can see all project titles listed on the screen.

User Stories

- Tasks and estimation
    - List all tasks to implement the user stories
        - Define a project class
        - Define a database table Project, and populate some initial data into the table or define a static variable to hold a list of project data
        - Implement a ViewProject Screen using ListView or RecycleView
    - The complexity is usually estimated in points
        - Linear scale: 1,2,3,4
        - Exponential scale: 1,2,4,8

Requirement Analysis

- Lo-fi mock screens:
  - Hand draw, or wireframe tools,
  - XML layout in AS

Software Design

- Application domain -> Solution domain
- A representation/model of the software to be built based on the requirements.
- High-level (architecture) design
  - Architecture style: client server, MVC, layered, 3(4)-tied ...
- Detailed design
  - Class model
- Design Goals
  - sufficiency, usability, efficiency (high performance, throughput, memory, response time), reliability, robustness, security, reusability, flexibility, ...

Software Design Principles

- Separate Variants from Invariants .
- A class should have only one reason to change. (single-responsibility principle)
- Classes should be open for extension but closed for modification (open-closed principle)
- Limit the number of classes to interact
- Liskov Substitution Principle: subclasses should be substitutable for their base classes.

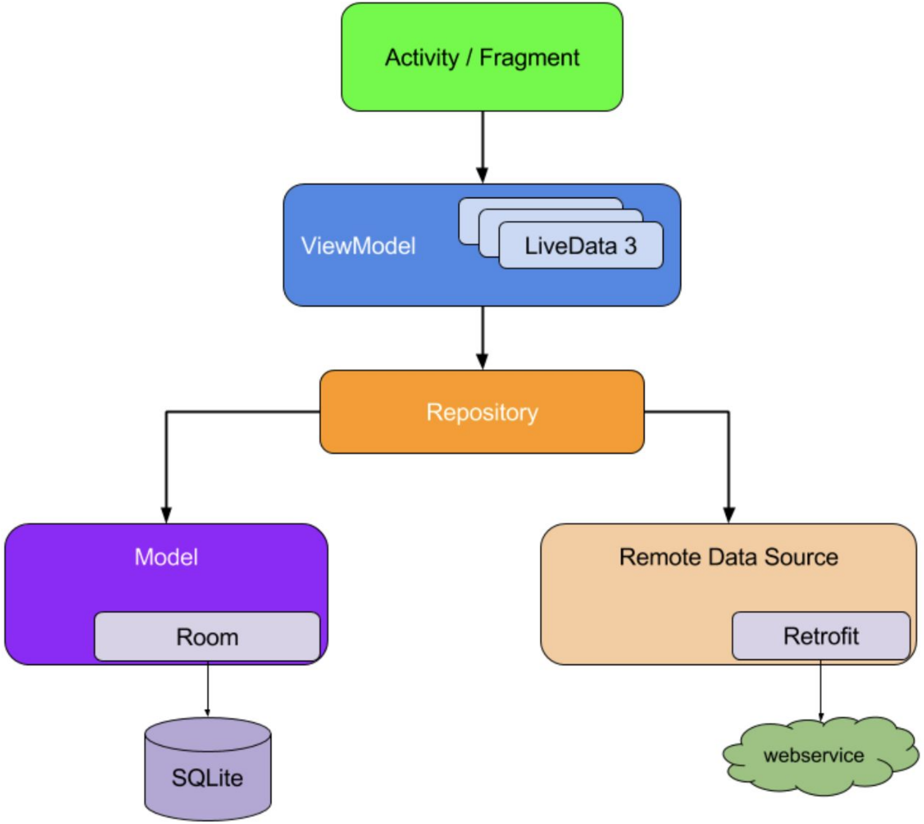
MVC

- Separate the business logic and application data from the presentation data to the user.
- Model: business logic, application data
- View: presentation of model, UI.
- Controller:
  - Intercept the request from view (UI) and pass to the model for proper actions.
  - Notify the views of the changes in the model.
- Model and view are independent to each other.
- In GUIs, the views and the controllers often work very closely together.
- Reusable and expressive.

MV?

- MVP
  - Presenter: as an interface between View and Model. Usually one-to-one relationship between Presenter and View.
- MVVM
  - ModelView: expose data from the Model to be consumed in the View
- MVWhatever?
  - Bottom line: separate model from presentation.
  - https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/

MVVM


Activity / Fragment
ViewModel
LiveData 3
Repository
Model
Room
SQLite
Remote Data Source
Retrofit
webservice

Design and Implementation

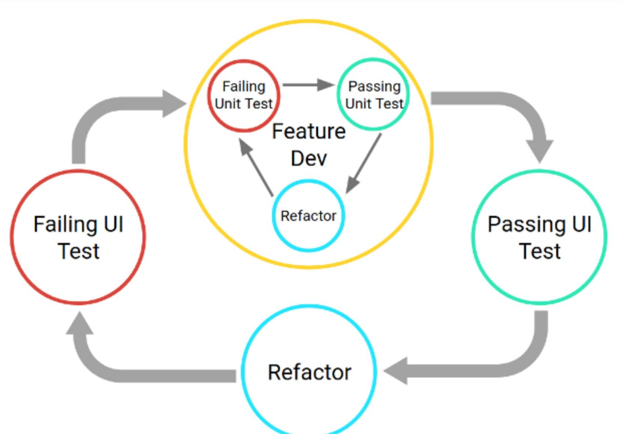
- Android Features
  - UI components: activities, fragments, view widgets, layouts, recycleviews, cardviews, viewpagers, etc.
  - Data storage and process: SQLite, Sharedpreferences, Files, Firebase storage
  - Background and asynchronous processing: services, threads, asyntasks, etc
  - Sensors, security, Graphics, animations, etc.
- MVC (one possible mapping) && MVVM
  - M: POJO, SQLite, V: View / Layout, C: Activity / fragment / Service
  - MVVM: ViewModel, Repository, LiveData
- Constantly Refactoring

Debugging

- Demo

Testing

- https://developer.android.com/training/testing/fundamentals
- Local tests vs Instrumented tests
  - Local tests: run on the local computer
  - Instrumented tests: run on the physical devices or emulators
- Unit tests vs UI tests
  - Unit Tests: test single units such as classes, methods
  - UI tests: test user interfaces
- Testing frameworks:
  - androidx.test, Robolectric, Espresso



Configuration Management

- Git

- GitHub

Intro to Kotlin

- Designed by JetBrains
- OO language, fully interoperable with Java, improved upon Java
- Statically typed with type inferences (safe and concise)
- Cross-platform
  - Multiple platform Mobile (Android, iOS)
  - Web development
    - server side framework,
    - front end: Kotlin/JS
- Kotlin compiler options:
  - Kotlin/JVM, Kotlin/JS, Kotlin/Native

https://kotlinlang.org/docs/home.html
https://developer.android.com/training/kotlinplayground

Basic Syntax

- Common keywords: import, package, class, **fun, open, override**
- Variable Keywords: val (value, immutable) & var (variable, mutable)
- Use “:” for type specification and inheritance.
- No need semicolon
- **Nullable and null check**:
    - Nullable vs non-nullable variable (use ?)
    - Safe call (?.)
    - Elvis operator (?:)
    -

Null Check

```
var str1:String? = null
var str2:String = null
println(str1?.length)
println(str1?.length?:-1)
println(str2.length)

str1 = "hello"
str2 = "world!"

println(str1?.length)
println(str1?.length?:-1)
println(str2!!.length)
```

Selection

- if/else: if(cond) statement1 else statement2
  - If expression:

```
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```

- when (similar to switch)

when{

cond1 -> statement1

cond2 -> statement2

…

else -> statementn

When expression

```
fun describe(obj: Any): String =
    when (obj) {
        1          -> "One"
        "Hello"    -> "Greeting"
        is Long    -> "Long"
        !is String -> "Not a string"
        else       -> "Unknown"
    }
```

Loop

- For loop

```
for (i in 1..3) {
    println(i)
}
for (i in 6 downTo 0 step 2) {
    println(i)
}
```

- While loop

```
while (x > 0) {
    x--
}

do {
    x++
} while (x <max)
```

Types

- Kotlin is a static type language as Java. Each variable has a type and it cannot be changed dynamically.
- It can be specified directly using “:” or inferred based on the context.

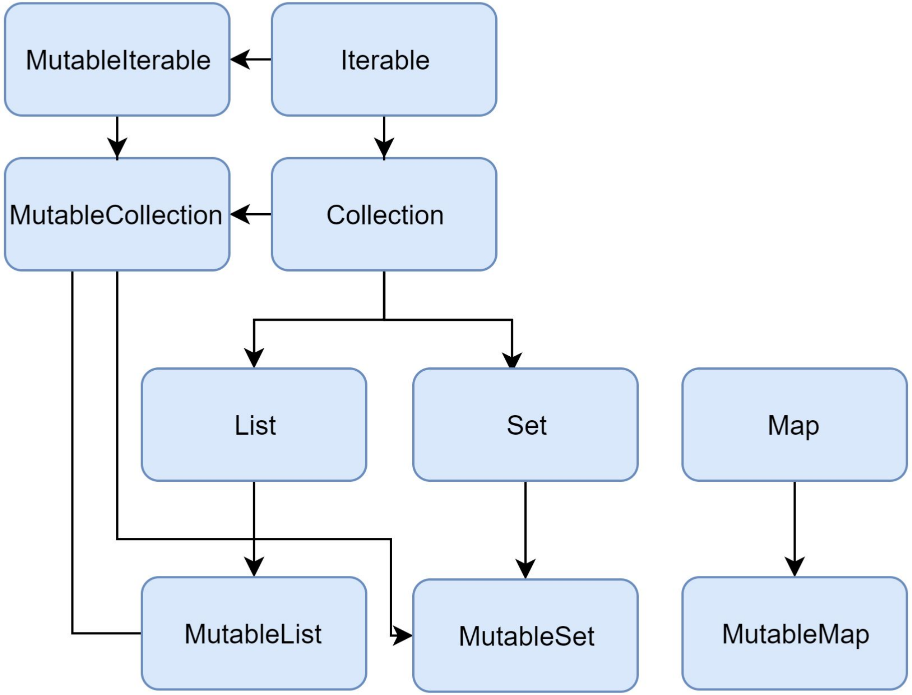
  val i: Int   var i = 5
- No primitive types. Everything is an object which has member functions and properties.
  - Byte, Short, **Int**, Long: 123, 456L, 0xa2d, 0b0102,
  - Float, **Double:** 3.4554, 39.23f, 8.3e2
  - Boolean: true, false
  - Char, String: '1', 'a', '\n', '\uFF00', "abc!\n","i=$i\n", "s is ${s.length} char(s). \n",

    """ abc .... def""" (raw string)
  - Array: arrayOf(1, 2, 3), IntArray(5) { 42 }

Collections

- List
- Set
- Map

```
val stringList = listOf("one",
"two", "one")
val stringSet = setOf("one",
"two", "three")
val numbersMap = mapOf("key1"
to 1, "key2" to 2, "key3" to 3,
"key4" to 1)
```



https://kotlinlang.org/docs/collections-overview.html#collection-types

Class

- The initializer, getter, and setter are optional if it can be inferred. Can have a default constructor.
- Late initialization

//(the Keyword constructor is usually omitted in the primary constructor)

```
class Person ~~constructor~~(val firstName: String,
                             val lastName: String,
                             var isEmployed: Boolean = true){
      lateinit var jobtitle: String
      /*...*/ }
```

- abstract, open, override
  - By default, a class is final (cannot be subclassed).
  - Need to use the keyword “open” to make it explicitly inheritable
  - Requires explicit modifiers for overridable members and overrides:
  - A class can be abstract. A member function can be abstract.

Kotlin Functions

- Kotlin functions are first-class
  - can be stored in variables and data structures,
  - can be passed as arguments to and returned from other higher-order functions.
  - can be operated as for other non-function values.
  - A function type: (A,B) -> C (list of parameter types and return type)
    - `((Int, Int) -> Int)?`
    - `(Int) -> ((Int) -> Unit)`
    - `()-> Unit`

Function literals

- More on: https://kotlinlang.org/docs/lambdas.html
- Lambda expressions and anonymous functions are *function literals*.
  - Not declared but are passed immediately as an expression.
  - Use anonymous functions if you need to declare the return type explicitly.

```
max(strings, { a, b -> a.length < b.length })
val sum = { x: Int, y: Int -> x + y }
val product = items.fold(1){ acc, e -> acc * e }
//trailing lambda
ints.filter { it > 0 } // this literal is of type
'(it: Int) -> Boolean'
strings.filter { it.length == 5 }.sortedBy { it }.map
{ it.uppercase() }
ints.filter(fun(item) = item > 0)
```

Additional Coding Example

- Dice Example
-
    -

Questions?