G53MDP
Mobile Device Programming

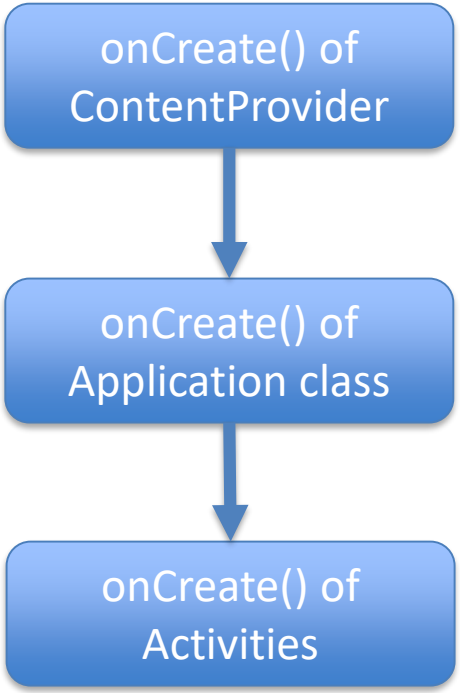
Lecture 19 – App Optimisation,
Testing & Deployment

Learning Outcomes

- Knowledge about code optimisation, testing.
- Familiarise with the App deployment process.

Application onCreate()

- onCreate() method is executed first in the Activity
- onCreate() of the ContentProviders is executed before onCreate() of Activity
- onCreate() of the Application is executed in between the ContentProvider and Activity



Application onCreate()

- Implementation:
  - Extending the Application object
  - Manifest: android:name="packagename.myapplicationname"
- What could be included in the onCreate() of Application?
  - Initialise the required SDK and libraries of your app
  - Register any dynamic broadcast receivers your app uses
  - Create and manage any services your app needs
- Provide the flexibility of controlling the entry point and able to respond efficiently at any stage of being invoked.
- Do not do any work that may block the app (e.g. create network connection)

Android Debug Bridge (adb)

- Adb is a command-line tool included in the Android SDK platform-tools package that allows installing/debugging/testing apps.
- Must enable USB debugging on the phone
- Examples
  - adb install *.apk
  - adb devices
  - adb shell dumpsys power

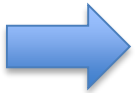
Check Memory Usage

- adb shell dumpsys meminfo *packagename*
  - *Check if memory size increased since launch*
  - *Number of objects that been created*
  - *Database information*
  - *Terminology:*
    - Native Heap: memory used by the process itself
    - Dalvik Heap: memory allocated by DalvikVM
    - Dalvik Other: memory used for JIT
    - Pss total: all connections to the main device memory thread
    - Private Dirty: actual amount of RAM the app is using on the heap since app started

Code Optimisation

- Minimise object creation
  - Not using temporary objects but use existing object. E.g. String returned from a method can be directly appended to a StringBuffer.
- Make method *static* if no need to modify the state of an object.
- Make any constants *static final*
- *Int* is 2x faster than *float*
- Enhanced *for* loops syntax (Java 1.5+)

```
int totalValue=0;
For (int i=0;i<myArray.length; ++i){
      totalValue+=myArry[i].myItem;
}
```



```
int totalValue=0;
For (<int> a:myArray){
      total+=a.myItem;
}
```

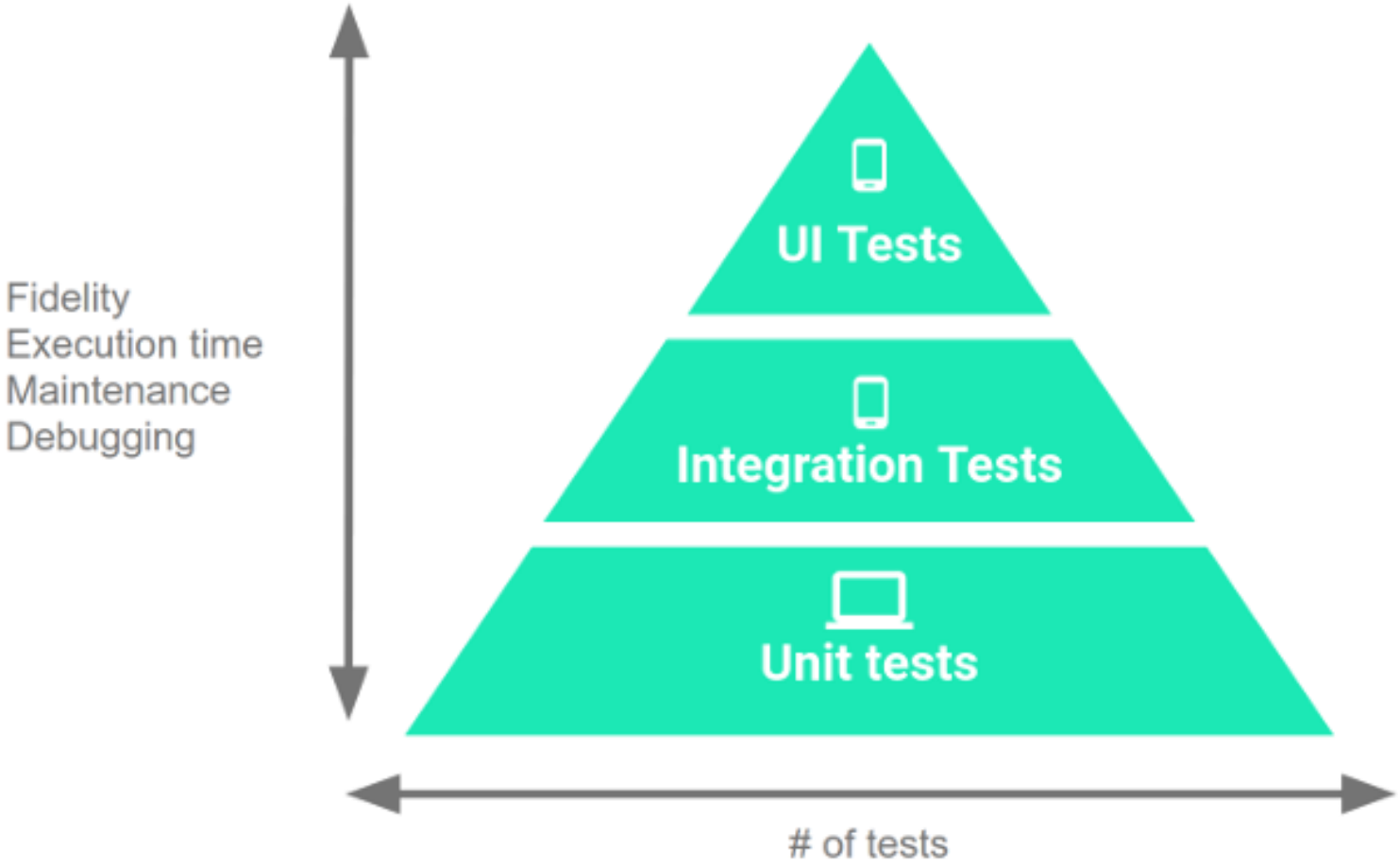
Shrink Code & Resources (Proguard)

- Set in build.gradle Do not use in debug mode which will slow down the build time

```
buildTypes {
    release {
        minifyEnabled true
        shrinkResources true
        proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
    }
}
```

- ProGuard setting file is used for code shrinking
  - Located in: /build/intermediates/proguard-files/
  - Try *proguard-android-optimize.txt* for more code shrinking
- Inappropriately code removal
  - Check app/build/outputs/mapping/release/usage.txt
  - add *-keep public class MyClass* in the ProGuard configuration file

App Testing



https://developer.android.com/training/testing/fundamentals.html

Unit Test

- Unit Tests: modules/ classes/ functions
- Test code in path: app/src/test/java
- Modify gradle.build:

  dependencies{ testCompile ‘junit:junit:4.12’ }
- Tools: Mockito, Robolectric
- Examples:
  https://github.com/googlesamples/android-testing
  https://developer.android.com/training/testing/samples.html

Integration & UI Test

- Test entire sequence of events, user interface, end-to-end testing on emulators/devices
- Test code in path: app/src/androidTest/java/
- Modify build.gradle:

```
dependencies {
    androidTestCompile 'com.android.support:support-annotations:24.0.0'
    androidTestCompile 'com.android.support.test:runner:0.5'
    androidTestCompile 'com.android.support.test:rules:0.5'
    // Optional -- Hamcrest library
    androidTestCompile 'org.hamcrest:hamcrest-library:1.3'
    // Optional -- UI testing with Espresso
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
    // Optional -- UI testing with UI Automator
    androidTestCompile 'com.android.support.test.uiautomator:uiautomator-
v18:2.1.2'
}
```

- Tools: AndroidJUnitRunner/ Espresso API / UI Automator/ Android Test Orchestrator

Application Release

- Make app as clean as possible:
  - Remove Logging, disable debugging (i.e. remove *android:debuggable* in manifest), remove tracing calls
  - Remove test libraries, frameworks, extra JAR files, unused layouts, strings, etc
  - Only keep the required <users-permission>.
  - Specify compatibility

```
<uses-sdk android:minSdkVersion="integer"
          android:targetSdkVersion="integer"
          android:maxSdkVersion="integer" />
```

  - Support library for backward compatibility: https://developer.android.com/topic/libraries/support-library/setup.html

APK Generation

- Use a good/unique *packagename:*
  - *Good: [org/com].[company].[product].[component]*
  - *Not allowed: com.android; com.google; android; com.example*
- Build->Generate Signed APK
- Create keystore file
- APK Signature Scheme V2
- Upload it to Google Play Developer Console

Application Publishing

- Production Checklist:
  - Certificate Keys: gain authorship and required for app maintainence
  - Contact Email: compulsory when publishing an app
- Google Play Developer Console
  - Sign up a Google Play Developer Account
  - Accept the developer distribution agreement
  - Pay one-off $25 registration fee
  - Application Screenshots
  - Promo Video
  - Policy: https://play.google.com/about/developer-content-policy/#!?modal_active=none

Summary

- Application onCreate()
- Check memory usage: android debug bridge
- Code optimisation & Shrinking
- Code tests
- Application deployment process

References

- https://developer.android.com/studio/publish/index.html
- https://developer.android.com/training/testing/fundamentals.html
- https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests.html
- https://developer.android.com/studio/build/shrink-code.html