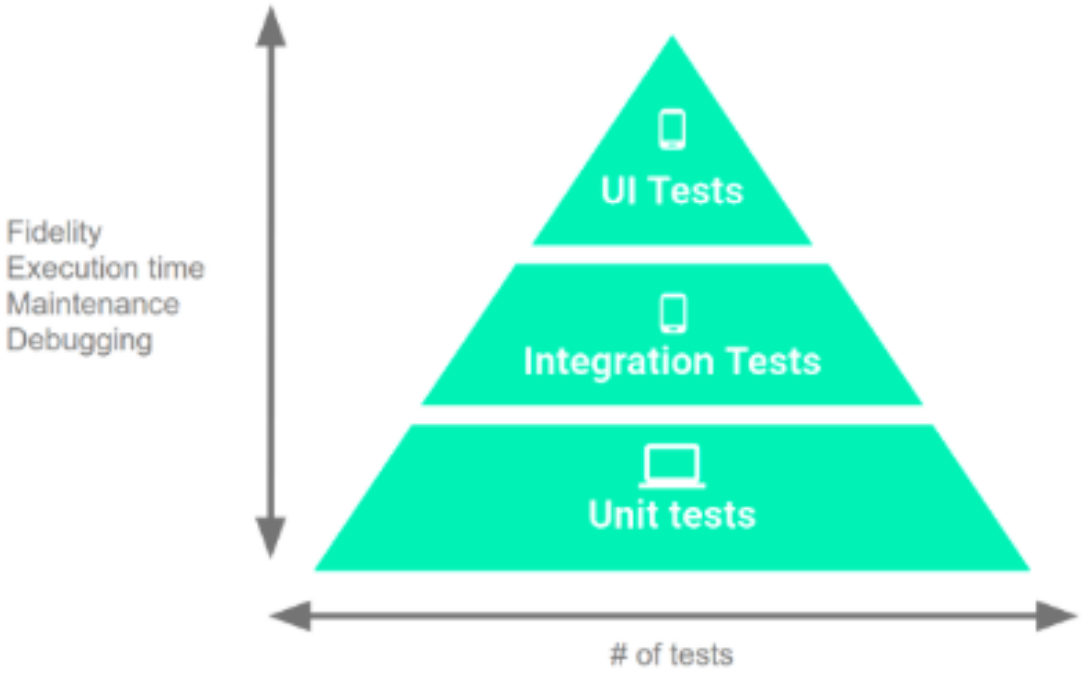
Testing in Android

Testing Android Applications

- Android applications run on a variety of devices
- To ensure application works well, it is important to write Software Tests
- Unit Test, Integration Tests, UI Tests are types of tests

Testing in Android



Unit test

- Unit testing is a testing a component or code for its function
- usually developer will carry out unit test on the code developed
- Unit test is classified into two types - Local unit tests and Instrumented unit tests
- Local unit tests run on the JVM
- Instrumented unit tests require the Android system

Testing Frameworks

Following are some of the testing frameworks used in Android:

- JUnit
- Mockito
- Powermock
- Robolectric
- Espresso
- Hamcrest

build.gradle

Whenever you start a new Android Studion Project, JUnit dependency is already present in the build.gradle

build.gradle

Whenever you start a new Android Studion Project, JUnit dependency is already present in the build.gradle

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0-beta
    implementation 'com.android.support.constraint:constraint-la
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1
    androidTestImplementation 'com.android.support.test.espresso
}
```

Folders in Android

In Android Studio Project, the following are the three important packages inside the src folder:

- app/src/main/java/ - Main java source code folder
- app/src/test/java/ - Local unit test folder
- app/src/androidTest/java/ - Instrumentation test folder

Creating Test Cases

- To create tests, we need to either extend the class with TestCase or add the annotation Test above the methods.
- TestCase was used mainly till JUnit3

A Test Case

```
package edu.manipal.util.temperature;

public class ConverterUtil {
  // converts to celsius
  public static float convertFahrenheitToCelsius(
      float fahrenheit) {
    return ((fahrenheit - 32 * 5 / 9));
  }

  // converts to fahrenheit
  public static float convertCelsiusToFahrenheit(
      float celsius) {
    return ((celsius * 9) / 5) + 32;
  }
}
```

A Test Case I

```
package edu.manipal.test;

import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import edu.manipal.util.temperature.ConverterUtil;

public class ConverterUtilTest {

    @Test
    public void testConvertFahrenheitToCelsius() {
        float actual = ConverterUtil.
            convertCelsiusToFahrenheit(100);
        // expected value is 212
        float expected = 212;
```

A Test Case II

```
        // use this method because float is not
            precise
        assertEquals("Conversion from celsius to
            fahrenheit failed", expected, actual,
            0.001);
    }

    @Test
    public void testConvertCelsiusToFahrenheit() {
        float actual = ConverterUtil.
            convertFahrenheitToCelsius(212);
        // expected value is 100
        float expected = 100;
        // use this method because float is not
            precise
        assertEquals("Conversion from celsius to
            fahrenheit failed", expected, actual,
            0.001);
    }
```

A Test Case III

```
}
```

Publishing and Distributing Apps

Distributing Apps

- Distributing the app through email, site, or App Store such as Google Play and Amazon
- Distributing through email is preferred when limited number of users
- Through website is convenient
- There is not much control over the app from the perspective of upgradation, maintenance

Distributing Apps through Online App Store

- Online App stores charge a nominal fee
- Provide various services such as monitoring the number and frequency of app downloads
- monitoring app usage patterns, establishing connect with service providers for localization and internationalization
- tracking app performance in the market, accessing end-user's feedback or viewing crash reports

Distributing Apps through Online App Store

- cater to distributing apps for a specific set of regions, carrier networks and device configurations
- Provide on-the-air update service for installed apps
- In case of paid pricing model, the online app stores may also charge percentage cut of the app selling price
- advanced app features are sold typically as in-app purchases
- content is sold as subscription(for content based apps)
- Indirect monetization is implemented via hosting advertisements in an app

Publishing Apps

Ground work

Steps

1. Sanitizing the app - removes unused resource files, Java source files used for testing and debugging purpose, log statements and unused libraries
2. app resources such as image files, locale specific strings should be cross checked for placement in appropriate folders
3. App manifst has to be checked for unique app package name
4. remote servers need to be ensured that they are ready and functional and prepared for anticipated load
5. pricing model has to be finalized
6. legal agreements such as End User License Agreement(EULA)
7. promotional artifacts such as app description, screen shots and promotional videos need to be designed.

Configuring App

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/
    apk/res/android"
package="com.mahe.Multimedia"
android:versionCode="1"
android:versionName="v1.0">
```

Next version of App

```
<manifest xmlns:android="http://schemas.android.com/
    apk/res/android"
package="com.mahe.Multimedia"
android:versionCode="2"
android:versionName="v1.1">
```

API requirements of an App

```
<uses-sdk android:minSdkVersion="18"
android:targetSdkVersion="23"/>
```

Packaging

- Android app need to be packaged as a signed .apk file
- apk file contains a .dex file, app manifest file and resource files such as images, icons, assets, layouts and menu
- During development and testing IDE creates an apk file and signs it using a default debug key

References

1. Pradhan, Anubhav, and Anil V. Deshpande. "Composing Mobile Apps" using Android." (2014).
2. Android App Development e-book, Abhishek, abhiandroid.com
3. http://developer.android.com
4. https://www.vogella.com/tutorials/AndroidTesting/article.html
5. https://www.journaldev.com/22674/android-unit-testing-junit4