

ANDROID UI testing

Application testing refers to the process of testing any software application using scripts, tools, or any test automation frameworks in order to identify errors. It helps teams release bug-free and robust software applications into the real world

Software testing is the act of examining the artifacts and the behaviour of the software under test by validation and verification. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Testing levels

Broadly speaking, there are at least three levels of testing: unit testing, integration testing, and system testing. However, a fourth level, acceptance testing, may be included by developers.

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.^[47]

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.

System testing

System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a login interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

Acceptance testing

Acceptance testing commonly includes the following four types:

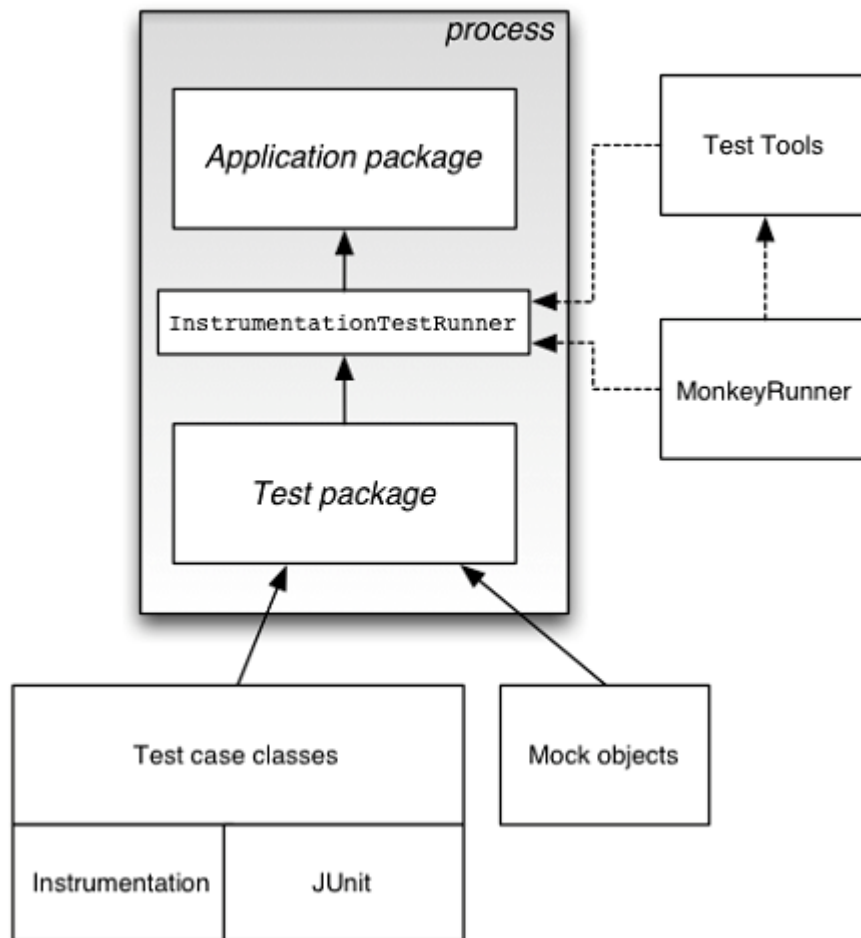
- User acceptance testing (UAT)
- Operational acceptance testing (OAT)
- Contractual and regulatory acceptance testing
- Alpha and beta testing

The Android framework includes an integrated testing framework that helps you test all aspects of your application and the SDK tools include tools for setting up and running test applications.

Whether you are working in Eclipse with ADT or working from the command line, the SDK tools help you set up and run your tests within an emulator or the device you are targeting.

Test Structure

Android's build and test tools assume that test projects are organized into a standard structure of tests, test case classes, test packages, and test projects.



Testing Tools in android

There are many tools that can be used for testing android applications. Some are official like JUnit, Monkey and some are third party tools that can be used to test android applications.

JUnit

You can use the JUnit **TestCase** class to do unit testing on a class that doesn't call Android APIs.

TestCase is also the base class for AndroidTestCase, which you can use to test Android-dependent objects.

Besides providing the JUnit framework, AndroidTestCase offers Android-specific setup, teardown, and helper methods.

In order to use TestCase, extend your class with TestCase class and implement a method call setUp(). Its syntax is given below –

```
public class MathTest extends TestCase {
    protected double fValue1;
    protected double fValue2;
```

```
protected void setUp() {
    fValue1= 2.0;
    fValue2= 3.0;
}
}
```

For each test implement a method which interacts with the fixture. Verify the expected results with assertions specified by calling `assertTrue(String, boolean)` with a boolean.

```
public void testAdd() {
    double result= fValue1 + fValue2;
    assertTrue(result == 5.0);
}
```

The assert methods compare values you expect from a test to the actual results and throw an exception if the comparison fails.

Once the methods are defined you can run them. Its syntax is given below –

```
TestCase test= new MathTest("testAdd");
test.run();
```

Monkey

The UI/Application Exerciser Monkey, usually called "monkey", is a command-line tool that sends pseudo-random streams of keystrokes, touches, and gestures to a device. You run it with the Android Debug Bridge (adb) tool.

You use it to stress-test your application and report back errors that are encountered. You can repeat a stream of events by running the tool each time with the same random number seed.

Monkey features

Monkey has many features, but it can be all be summed up to these four categories.

- Basic configuration options
- Operational constraints
- Event types and frequencies
- Debugging options

Monkey Usage

In order to use monkey, open up a command prompt and just navigate to the following directory.

```
android ->sdk ->platform-tools
```

Once inside the directory, attach your device with the PC , and run the following command.

```
adb shell monkey -p your.package.name -v 500
```

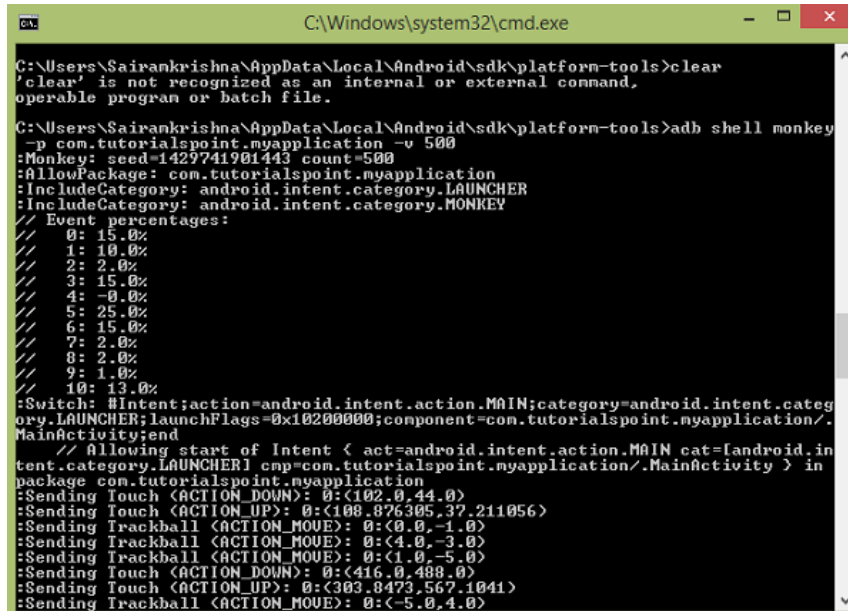
This command can be broken down into these steps.

- adb - Android Debug Bridge. A tool used to connect and sends commands to your Android phone from a desktop or laptop computer.

MODULE 2

- shell - shell is just an interface on the device that translates our commands to system commands.
- monkey - monkey is the testing tool.
- v - v stands for verbose method.
- 500- it is the frequency count or the number of events to be sent for testing.

This is also shown in the figure –



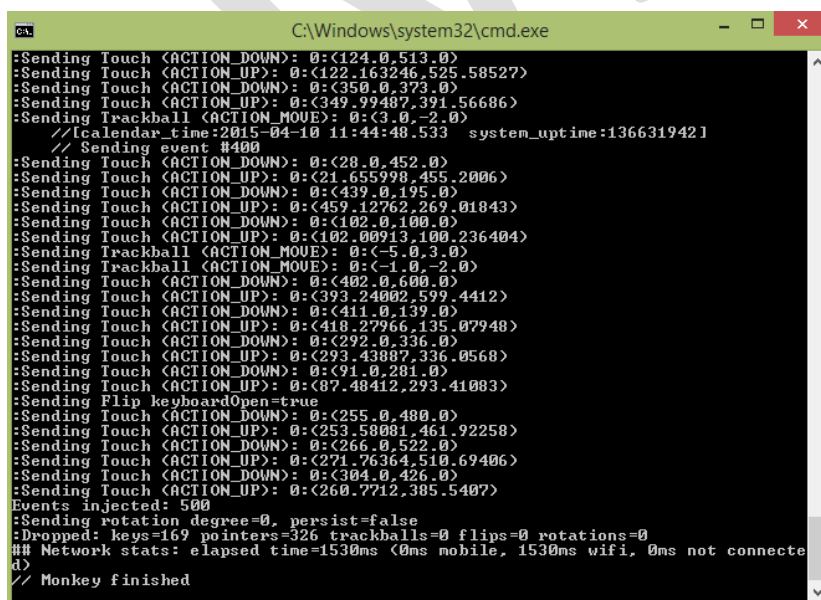
```
C:\Windows\system32\cmd.exe
C:\Users\Sairankrishna\AppData\Local\Android\sdk\platform-tools>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Sairankrishna\AppData\Local\Android\sdk\platform-tools>adb shell monkey
-p com.tutorialspoint.myapplication -v 500
Monkey: seed=1429741961443 count=500
AllowPackage: com.tutorialspoint.myapplication
IncludeCategory: android.intent.category.LAUNCHER
IncludeCategory: android.intent.category.MONKEY
Event percentages:
0: 15.0%
1: 10.0%
2: 2.0%
3: 15.0%
4: -0.0%
5: 25.0%
6: 15.0%
7: 2.0%
8: 2.0%
9: 1.0%
10: 13.0%
Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.tutorialspoint.myapplication/.MainActivity;end
// Allowing start of Intent < act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.tutorialspoint.myapplication/.MainActivity > in package com.tutorialspoint.myapplication
Sending Touch (ACTION_DOWN): 0:(102.0,44.0)
Sending Touch (ACTION_UP): 0:(100.076305,37.211056)
Sending Trackball (ACTION_MOVE): 0:(0.0,-1.0)
Sending Trackball (ACTION_MOVE): 0:(4.0,-3.0)
Sending Trackball (ACTION_MOVE): 0:(1.0,-5.0)
Sending Touch (ACTION_DOWN): 0:(416.0,488.0)
Sending Touch (ACTION_UP): 0:(303.8473,567.1041)
Sending Trackball (ACTION_MOVE): 0:(-5.0,4.0)
```

In the above command, you run the monkey tool on the default android UI application. Now in order to run it to your application , here what you have to do.

finally you will get finish as shown bellow

This has also been shown in the figure below. By typing this command , you are actually generating 500 random events for testing.



```
C:\Windows\system32\cmd.exe
Sending Touch (ACTION_DOWN): 0:(124.0,513.0)
Sending Touch (ACTION_UP): 0:(122.163246,525.58527)
Sending Touch (ACTION_DOWN): 0:(350.0,373.0)
Sending Touch (ACTION_UP): 0:(349.99487,391.56686)
Sending Trackball (ACTION_MOVE): 0:(3.0,-2.0)
// [calendar_time:2015-04-10 11:44:48.533 system_uptime:1366319421
// Sending event #400
Sending Touch (ACTION_DOWN): 0:(28.0,452.0)
Sending Touch (ACTION_UP): 0:(21.655998,455.2006)
Sending Touch (ACTION_DOWN): 0:(439.0,195.0)
Sending Touch (ACTION_UP): 0:(459.12762,269.01843)
Sending Touch (ACTION_DOWN): 0:(102.0,100.0)
Sending Touch (ACTION_UP): 0:(102.00913,100.236404)
Sending Trackball (ACTION_MOVE): 0:(-5.0,3.0)
Sending Trackball (ACTION_MOVE): 0:(-1.0,-2.0)
Sending Touch (ACTION_DOWN): 0:(402.0,600.0)
Sending Touch (ACTION_UP): 0:(393.24002,599.4412)
Sending Touch (ACTION_DOWN): 0:(411.0,139.0)
Sending Touch (ACTION_UP): 0:(418.27966,135.07948)
Sending Touch (ACTION_DOWN): 0:(292.0,336.0)
Sending Touch (ACTION_UP): 0:(293.43887,336.0568)
Sending Touch (ACTION_DOWN): 0:(91.0,281.0)
Sending Touch (ACTION_UP): 0:(87.48412,293.41083)
Sending Flip keyboardOpen=true
Sending Touch (ACTION_DOWN): 0:(255.0,480.0)
Sending Touch (ACTION_UP): 0:(253.50001,461.92258)
Sending Touch (ACTION_DOWN): 0:(266.0,522.0)
Sending Touch (ACTION_UP): 0:(271.76364,510.69406)
Sending Touch (ACTION_DOWN): 0:(304.0,426.0)
Sending Touch (ACTION_UP): 0:(260.7712,385.5407)
Events injected: 500
Sending rotation degree=0, persist=false
Dropped: keys=169 pointers=326 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=1530ms (0ms mobile, 1530ms wifi, 0ms not connected)
// Monkey finished
```

Example

The below example demonstrates the use of Testing. It creates a basic application which can be used for monkey.

To experiment with this example, you need to run this on an actual device and then follow the monkey steps explained in the beginning.

Here is the content of **MainActivity.java**.

```
package com.bvocsaad.myapplication;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    Button b1;
    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        b1=(Button)findViewById(R.id.button);
    }

    public void button(View v){
        Intent in =new Intent(MainActivity.this,second.class);
        startActivity(in);
    }
}
```

Here is the content of **second.java**.

```
package com.bvocsaad.myapplication;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class second extends Activity{
    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view);
        Button b1=(Button)findViewById(R.id.button2);
    }
}
```

```

b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(second.this,"Thanks",Toast.LENGTH_SHORT).show();
    }
});
}
}
}

```

Here is the content of **activity_main.xml**.

In the below code **abc** indicates the logo of bvocsaad.com

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="UI Animator Viewer"
        android:id="@+id/textView"
        android:textSize="25sp"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bvoc saad"
        android:id="@+id/textView2"
        android:layout_below="@+id/textView"
        android:layout_alignRight="@+id/textView"
        android:layout_alignEnd="@+id/textView"
        android:textColor="#ff36ff15"
        android:textIsSelectable="false"
        android:textSize="35dp" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/abc"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true" />

```

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    android:onClick="button"
    android:id="@+id/button"
    android:layout_below="@+id/imageView"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="100dp" />

</RelativeLayout>

```

Here is the content of **view.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="button"
        android:id="@+id/button2"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bvoc saad "
        android:id="@+id/textView3"
        android:textColor="#ff3aff22"
        android:textSize="35dp"
        android:layout_above="@+id/button2"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="90dp" />

</RelativeLayout>

```

Here is the content of **Strings.xml**.

```

<resources>
    <string name="app_name">My Application</string>
</resources>

```

Here is the content of **AndroidManifest.xml**.

Android UI testing can be a challenge, but there are some best practices that can make it easier.

- It is important to create an effective test plan. This should include what you want to test and how you want to test it.

MODULE 2

- Make a test plan keeping in mind [code coverage and test coverage](#). The right balance between coverage and time has to be maintained for delivering a high-quality application in a shorter period of time.
- Take Screenshots or Video recordings in case of a failed test for better debugging.
- Share Bug Report with the team via integrations like Slack, GitHub, JIRA, or Trello for an effective project management
- Test on a [real device cloud](#) for better accuracy
- Integrate to run tests from CI/CD pipeline
