



SAN JOSÉ STATE  
UNIVERSITY

# UI Test Automation on Android

By

Karthik Vakati



# UI Test Automation on Android

No Quiz Today



# UI Test Automation on Android

Two ways to test your application's User Interface

## ➤ Manual Testing

- Run tests manually and verify that the app is behaving as expected
- Time-consuming, tedious, and error-prone



# UI Test Automation on Android

Two ways to test your application's User Interface

## ➤ Automated Testing

- Automate the UI testing with a software testing framework



# UI Test Automation on Android

The Android SDK provides the following tools:

- `uiautomatorviewer` - A GUI tool to scan and analyze the UI components of an Android application.
- `uiautomator` - A Java library containing APIs to create customized functional UI tests, and an execution engine to automate and run the tests.



# UI Test Automation on Android

Workflow for the **uiautomator** testing framework

1. Prepare to test
2. Create automated tests to simulate specific user interactions on your application
3. Compile your test cases into a JAR file and install it on your test device along with your app
4. Run the tests and view the test results
5. Correct any bugs or defects discovered in testing



# UI Test Automation on Android

## 1. Prepare to Test

1. Load the application to a device
2. Identify the application's UI components
3. Ensure that the application is accessible
4. Configure your development environment



# UI Test Automation on Android

## 2. Create Tests using `uiautomator` framework

- Built on top of Junit framework
- Create test cases that extend the `UiAutomatorTestCase`
- Since `UiAutomatorTestCase` extends junit framework's `TestCase`, you can make use of Junit `Assert` class
- Capture and manipulate UI components using classes like `UiDevice`, `UiSelector`, `UiObject`, `UiCollection`.....





# UI Test Automation on Android

## 2. Create Tests using `uiautomator` framework (cont...)

### ➤ `UiDevice`

- Device that contains the target app
- The first thing your test case should do is access the device

### ➤ `UiSelector`

- Represents a search criteria to query and get a handle on specific elements in the currently displayed UI



# UI Test Automation on Android

## 2. Create Tests using `uiautomator` framework (cont...)

### ➤ `UiObject`

- Represents a UI element
- Use `UiSelector` to get a handle on a specific UI element and assign it to `UiObject`

### ➤ `UiCollection`

- Represents a collection of items
- Use `UiSelector` to search for a UI element that is a container or wrapper of other child UI elements and assign it to `UiCollection`



# UI Test Automation on Android

## 3. Building and Deploying Your Tests

1. Create the required build configuration files

```
<android-sdk>/tools/android create uitest-project -n <name> -t 1  
-p <path>
```

2. From the commandline, set the ANDROID\_HOME var

- In Windows

```
set ANDROID_HOME=<path_to_your_sdk>
```

- In Linux

```
export ANDROID_HOME=<path_to_your_sdk>
```



# UI Test Automation on Android

## 3. Building and Deploying Your Tests (cont...)

3. Go to the project directory where your build.xml file is located and build your test JAR

```
ant build
```

4. Deploy your generated test JAR file to the test device

```
adb push <path_to_output_jar> /data/local/tmp/
```



# UI Test Automation on Android

## 4. Running uiautomator Tests

```
adb shell uiautomator runtest <name_of_jar> -c  
<name_of_package_that_contains_test_cases>
```



# UI Test Automation on Android

Today's Lab

Run automated tests on Homework 2



# UI Test Automation on Android

- Copy my Homework 2 solutions
- Connect a device or start the emulator
- Locate the hw02.apk file under hw02/bin/ directory
- Run *adb install <path\_to\_hw02/bin/hw02.apk>* in command prompt to install Homework 2 onto the device
- Bring the app *Homework 2* onto the home screen



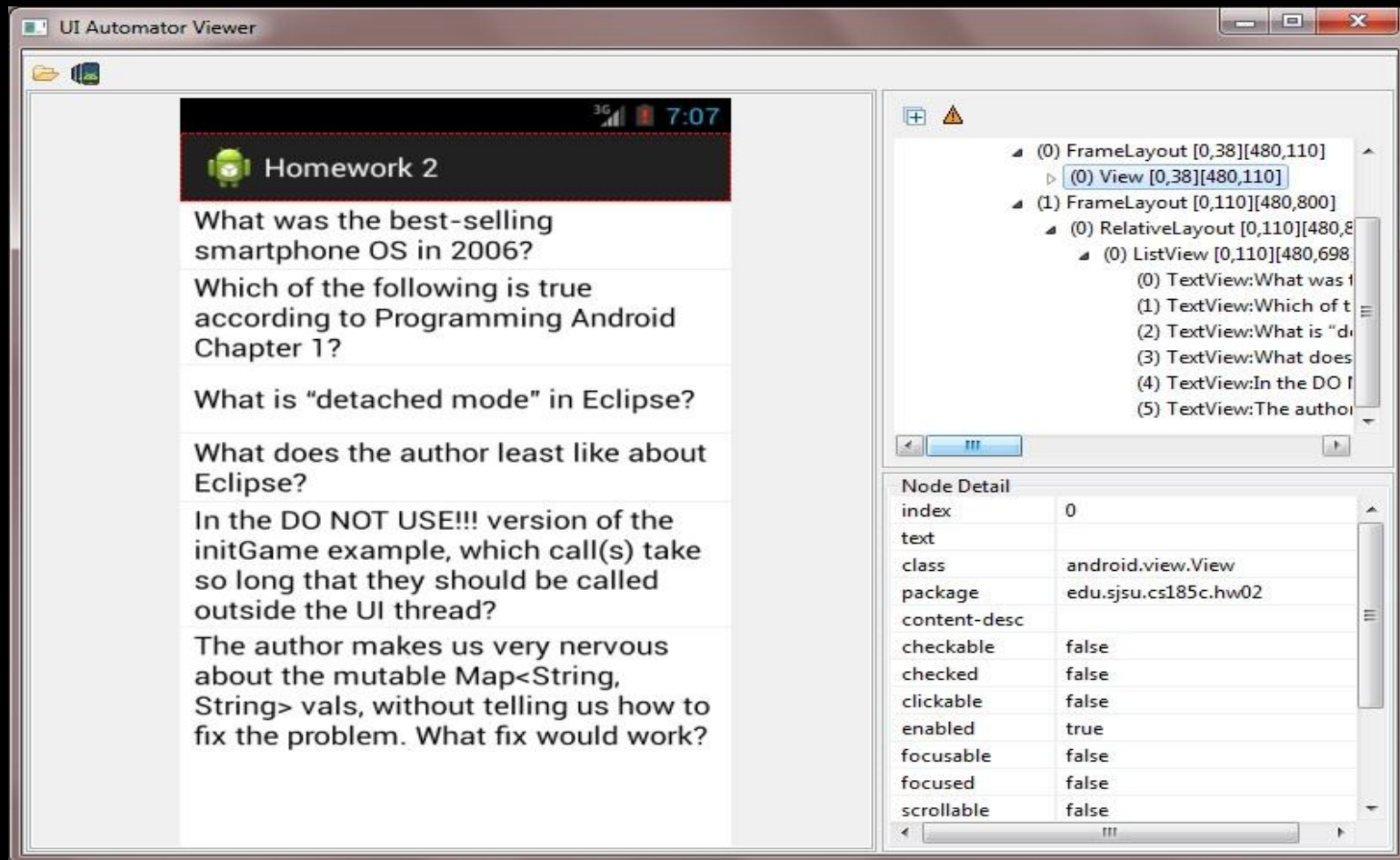
# UI Test Automation on Android

1. To identify the UI components, open *Homework 2* on the emulator.
2. In the terminal, navigate to the folder `<android-sdk>/tools/` and run `$ uiautomatorviewer`
3. From the GUI of the uiautomatorviewer tool, click on Device Screenshot to analyze the components of the current screen
4. Note down the fields *text* or *content-desc* for the UI elements that you want to test.
5. Select one of the Questions.
6. Perform Step 2, 3 and 4 again





# UI Test Automation on Android



The screenshot shows the UI Automator Viewer application. The main window displays a simulated Android interface with a homework assignment titled "Homework 2". The assignment contains five questions:

- What was the best-selling smartphone OS in 2006?
- Which of the following is true according to Programming Android Chapter 1?
- What is "detached mode" in Eclipse?
- What does the author least like about Eclipse?
- In the DO NOT USE!!! version of the initGame example, which call(s) take so long that they should be called outside the UI thread?
- The author makes us very nervous about the mutable Map<String, String> vals, without telling us how to fix the problem. What fix would work?

The right-hand side of the application shows the UI hierarchy tree. The selected node is a View with the following details:

Node Detail	
index	0
text	
class	android.view.View
package	edu.sjsu.cs185c.hw02
content-desc	
checkable	false
checked	false
clickable	false
enabled	true
focusable	false
focused	false
scrollable	false



# UI Test Automation on Android

- Create a Java Project in Eclipse and name it *hw02Test*
- From the Java Build Path
  - Click Add Library > JUnit then select JUnit3 to add JUnit support.
  - Click Add External JARs... and navigate to the SDK directory. Under the platforms directory, select the latest SDK version and add both the `uiautomator.jar` and `android.jar` files.
- Create the package `edu.sjsu.cs185c.hw02.test` and the class *SimpleTestCase* under it.



# UI Test Automation on Android

- Now it's time to create the tests.
- Open the file `SimpleTestCase.java` and copy the code.
- Now it's time to deploy and run the tests.
- In the terminal run `$ android list targets`
- Note the value of the field `id` for android level 17 or higher



# UI Test Automation on Android

- Create the required build configuration files to build the output JAR. In the terminal, run

```
$ <android-sdk>/tools/android create uitest-project -n <name> -t  
<id> -p <path>
```

where <name> represents the name of the test project, <path> represents the path for the test project and <id> represents the id value you captured earlier.

- export ANDROID\_HOME=<path\_to\_your\_sdk> under linux



# UI Test Automation on Android

- Go to the project directory where your build.xml file is located and build your test JAR.

```
$ ant build
```

- Deploy your generated test JAR file to the test device by using the adb push command

```
$ adb push <path_to_output_jar> /data/local/tmp/
```

- Run the test using the adb shell command

```
$ adb shell uiautomator runtest hw02Test.jar -c  
edu.sjsu.cs185c.hw02.test
```



# UI Test Automation on Android

- Now let's add more code to our test class.
- Open the file SimpleTestCase.java and add the following code.

```
UiObject smartphone= new UiObject(new UiSelector()  
.text("What was the best-selling smartphone OS in 2006?"));  
if (smartphone.exists()) { smartphone.click(); sleep(5000); }  
assertTrue(new UiObject(new UiSelector()  
.text("What was the best-selling smartphone OS in 2006?")).exists());  
UiObject option1 = new UiObject(new UiSelector().text("Android"));  
if (option1.exists()) { option1.clickAndWaitForNewWindow(); }
```

- Deploy and run the test again.

