Ui Testing

The Android SDK provides the following tools to support automated, functional UI testing on your application:

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

Minimum Requirements to set Environment

3. JDK 1.6
4. Android SDK Tools, Revision 21 or higher
5. Android SDK Platform, API 16 or higher
6. Ant

Analyzing Your Application's UI

To analyze the UI components of the application

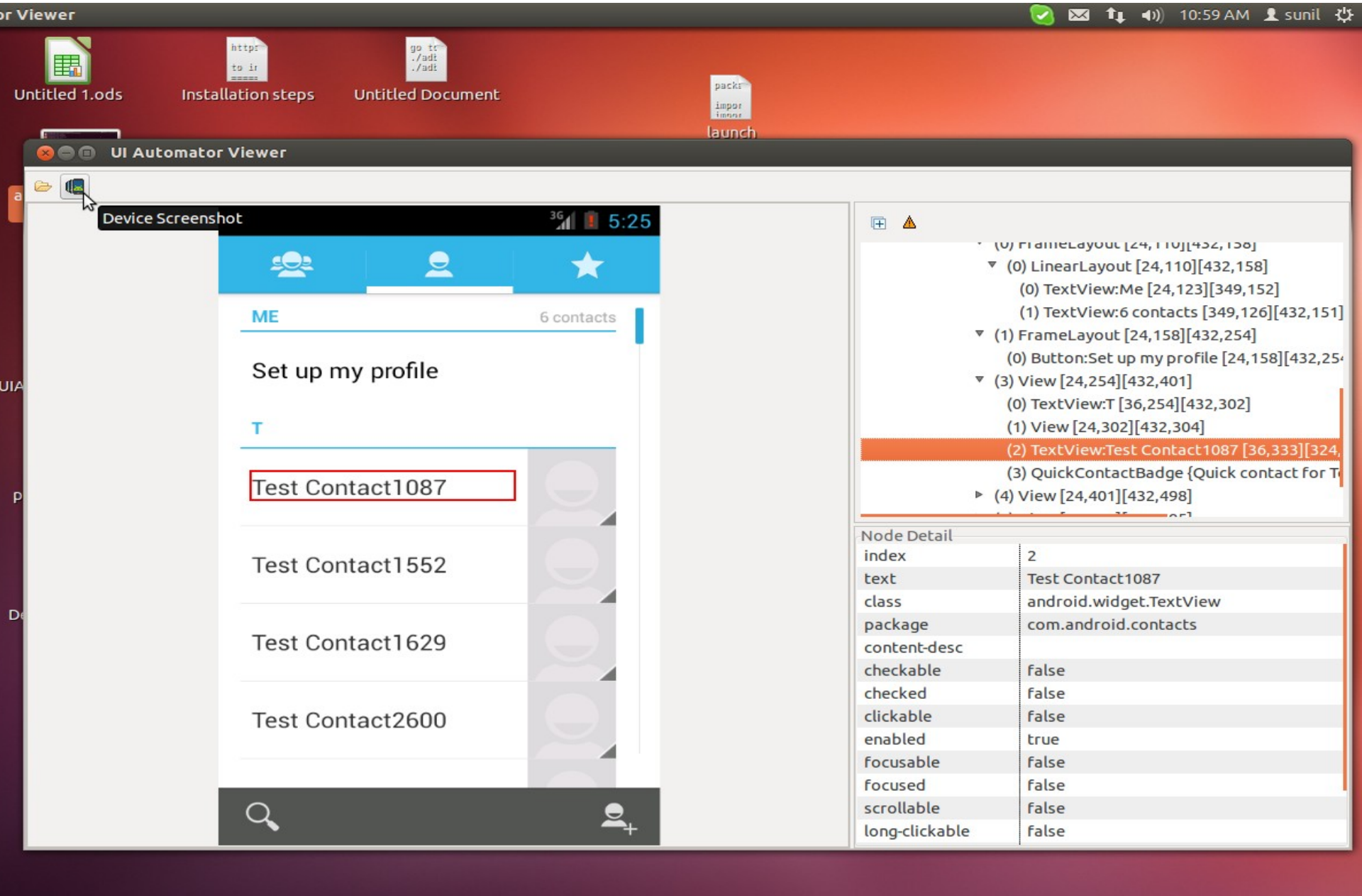
1. Connect your Android device to your development machine.
2. Open a terminal window and navigate to <android-sdk>/tools/ and Run the tool with this command: ./uiautomatorviewer

   To Start analyze click on Device Screen shot button in ui automator viewer tool

Reference URL:

http://developer.android.com/tools/testing/testing_ui.html#uianalysis

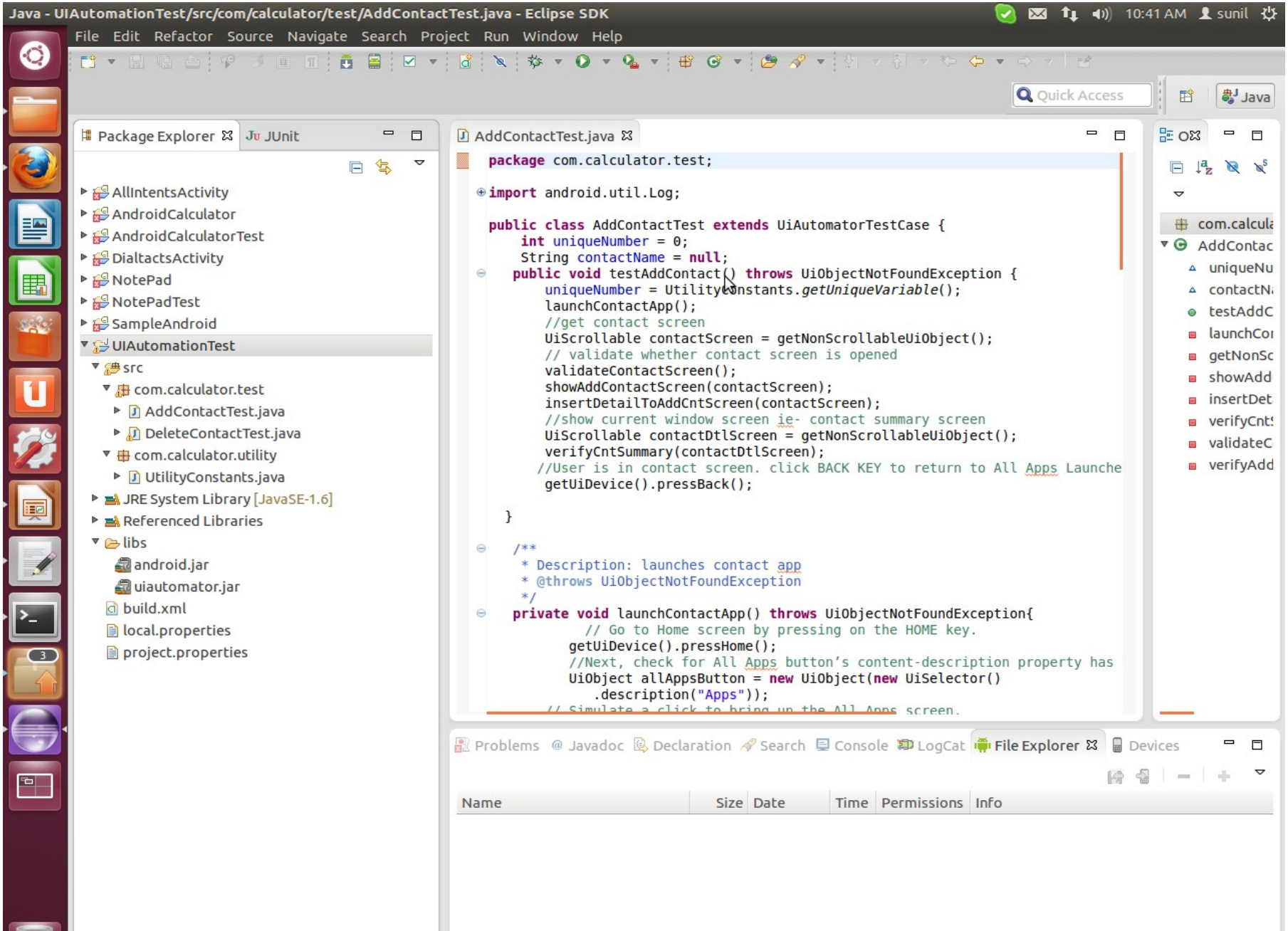
UI Automator Tool



Configuring development environment

- Create a new Java project in Eclipse, and give your project a name that is relevant to the tests you're about to create (for example, "MyAppNameTests"). In the project, you will create the test cases that are specific to the application that you want to test.
- From the **Project Explorer**, right-click on the new project that you created, then select **Properties > Java Build Path**, and do the following:
  - 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.

Development Project Structure



Developing

- For further reference please have a look on following link

http://developer.android.com/tools/testing/testing_ui.html#creating

Ui Elements

Classes

Class	Description
com.android.uiautomator.core.UiCollection	Used to enumerate a container's user interface (UI) elements for the purpose of counting, or targeting a sub elements by a child's text or description.
com.android.uiautomator.core.UiDevice	Provides access to state information about the device. You can also use this class to simulate user actions on the device, such as pressing the d-pad hardware button or pressing the Home and Menu buttons.
com.android.uiautomator.core.UiObject	Represents a user interface (UI) element.
com.android.uiautomator.core.UiScrollable	Provides support for searching for items in a scrollable UI container.
com.android.uiautomator.core.UiSelector	Represents a query for one or more target UI elements on a device screen.

Interfaces

Interface	Description
com.android.uiautomator.core.UiWatcher	Represents a conditional watcher on the target device.
com.android.uiautomator.testrunner.IAutomationSupport	Provides auxiliary support for running test cases.
com.android.uiautomator.testrunner.UiAutomatorTestCase	Defines an environment for running multiple tests. All `uiautomator` test cases should extend this class.

Exceptions

Exception	Description
com.android.uiautomator.core.UiObjectNotFoundException	Indicates when a a `UiSelector` could not be matched to any UI element displayed.

Deploying And Running uiautomator Tests

1. First check list of android devices that you have. Use following syeps

   Open your terminal. Navigate to tools

   Example: Run following command

   cd /home/sunil/Development/android-sdk-linux/tools/

   Run following command in your terminal to get list of devices and its id.

   ./android list targets

2. Create the required build configuration files i.e. (build.xml) .

   Run the following command: in terminal

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

   <name> is the name of the project that contains your uiautomator test source files, and the <path> is the path to the corresponding project directory.

   Example: ./android create uitest-project -n UIAutomationTest -t 6 -p /home/sunil/workspace/UIAutomationTest

   O/p: build.xml will be generated if not generated before, otherwise build.xml file will be updated

3. From the command line, set the ANDROID_HOME variable:

   In Windows:set ANDROID_HOME=<path_to_your_sdk>

   In UNIX:export ANDROID_HOME=<path_to_your_sdk>

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

Example:

cd /home/sunil/workspace/UIAutomationTest

And run following command in terminal to generate jar file

ant build

5.Deploy your generated test JAR file to the test device by using following command

./adb push command:adb push <path_to_output_jar> /data/local/tmp/

Example:

cd /home/sunil/Development/android-sdk-linux/platform-tools/

push the jar file

./adb push /home/sunil/workspace/UIAutomationTest/bin/UIAutomationTest.jar /data/local/tmp/

Running uiautomator Tests

Example:

./adb shell uiautomator runtest UIAutomationTest.jar -c com.calculator.test.AddContactTest

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

Example:

cd /home/sunil/workspace/UIAutomationTest

And run following command in terminal to generate jar file

ant build

5.Deploy your generated test JAR file to the test device by using following command

./adb push command:adb push <path_to_output_jar> /data/local/tmp/

Example:

cd /home/sunil/Development/android-sdk-linux/platform-tools/

push the jar file

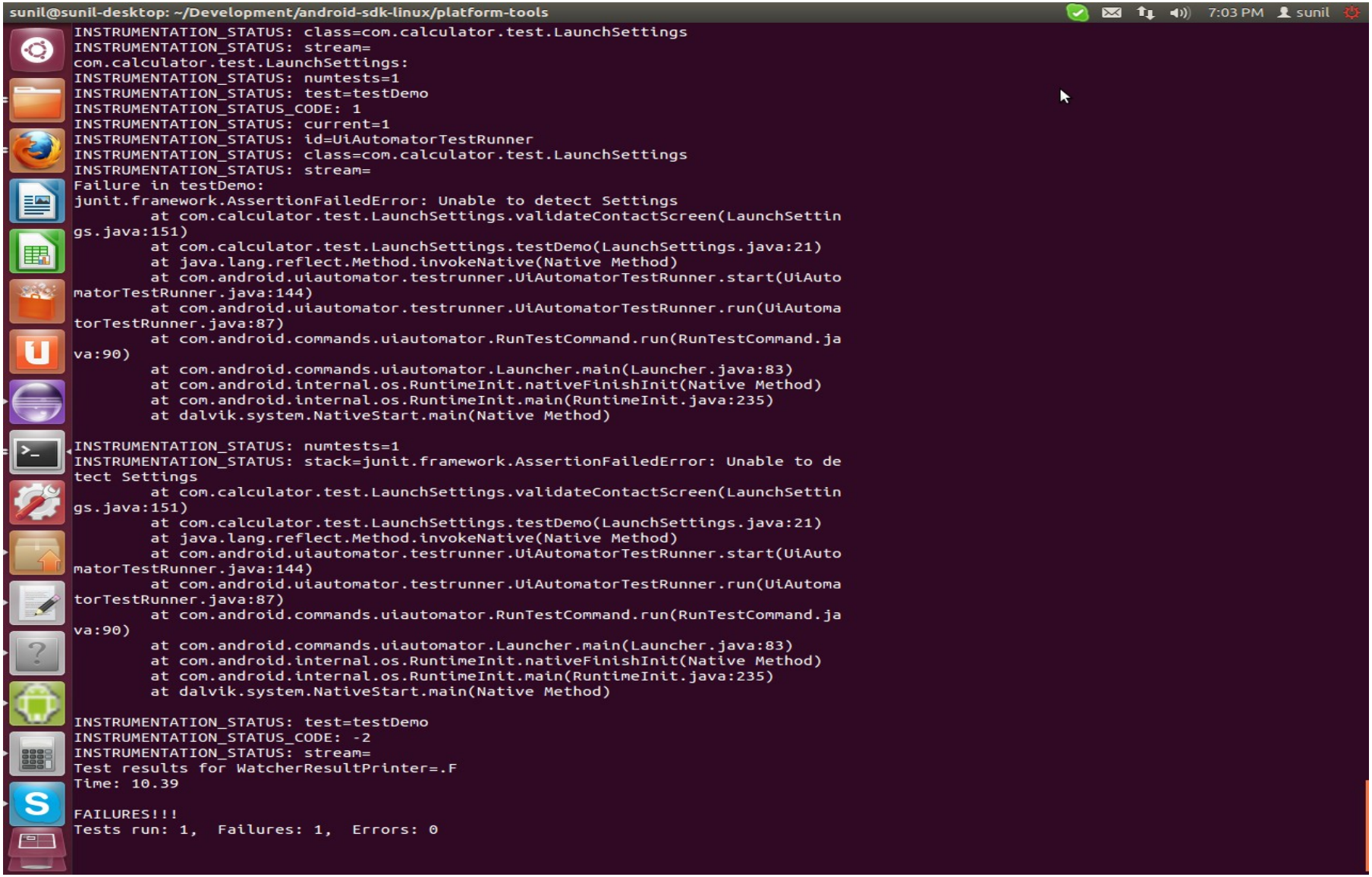
./adb push /home/sunil/workspace/UIAutomationTest/bin/UIAutomationTest.jar /data/local/tmp/

Running uiautomator Tests

Example:

./adb shell uiautomator runtest UIAutomationTest.jar -c com.calculator.test.AddContactTest

Assertion Failure Message



Thank You