1. Download Appium:
   1. From appium.io
   2. Extract into a folder called ‘Appium’ and place it in C:\
2. Download and install ADT:
   1. Download the standalone SDK tool zip file for Windows at: <https://developer.android.com/sdk/installing/installing-adt.html>
   2. Unzip the package. Note the ‘sdk’ folder address.
   3. Create ANDROID\_HOME environment variable and point it to your ADT’s sdk folder.
   4. Add the paths to ‘tools’ and ‘platform-tools’ to your PATH variable. Use %ANDROD\_HOME%
3. Download node.js Windows installer from <http://nodejs.org/download/> . Install it.
4. Download and install Maven
   1. Download the latest Maven version for your system from <http://maven.apache.org/download.cgi>
   2. Extract the downloaded file to a folder (recommend shortening folder’s name and moving it to C:\)
   3. If you don’t have Java JDK (not JRE) installed, download and install it.
   4. Create or edit a JAVA\_HOME variable and point it to your JDK folder. Then add %JAVA\_HOME%\bin to your PATH variable.
   5. Create or edit an M2\_HOME variable and point it to your extracted Maven folder
   6. Create or edit an M2 variable and point it to %M2\_HOME%\bin
   7. Add both variables to your PATH
   8. To check that Maven is properly installed, open cmd window and type mvn --version
   9. Depending on the network your system is in, you might have to configure proxy settings for Maven to work. Edit the proxy settings in the settings.xml file in the ‘conf’ folder inside your Maven extracted folder. In particular, edit this section:

<proxy>

<id>http\_proxy</id>

<active>true</active>

<protocol>http</protocol>

<username>proxyuser</username>

<password>proxypass</password>

<host>proxy-us.intel.com</host>

<port>911</port>

<nonProxyHosts>\*.intel.com</nonProxyHosts>

</proxy>

1. Install Cygwin and add it to your Windows PATH. Make sure you can do something like ‘ls | grep “string”’ on your cmd console.
2. Download the zipped file and copy it inside the Appium folder. Unzip it there.
3. Drill down the Appium folder structure until you find Appium.exe and open it. Then ‘Launch’ it with the default setting by clicking on the ‘Play’ icon in the upper right hand corner. If it does not launch, then there might be a problem with your node.js server installation.
4. Perform setup steps for Social Camera:
   1. Phone locking is on default
   2. Settings > Developer options:
   * Checkmark ‘Stay awake’
   * Enable USB debugging of course
   * Uncheck ‘Verify apps over USB’
   * Might be useful to checkmark ‘Pointer location’ to see the locations where the phone is tapped on.
5. Open cmd window and change directory to where your test assets are (i.e., go inside the unzipped ‘cam23\_UI\_tests\_BB’ folder and start ‘cmd’ console from there.)
6. Execute ‘adb root’ and ‘adb remount’.
7. Execute “adb shell pm clear com.intel.camera22” to reset the camera so we can get the ‘Enable Geo-tag’ dialog when we start camera.
8. Unlock your device and execute:

mvn clean install -Dtest=[Your test class, i.e., the .java file without the .java extension and square brackets]

Example: mvn -Dtest=FtuxAndBasicCaptureTest clean test

Appium can unlock your device for you automatically, but I found it a little glitchy!

P.S.:

For different ways to locate an element, you can try looking inside the tests here:

<https://github.com/appium/java-client/tree/master/src/test/java/io/appium/java_client/android>

For documentation on the Appium Java bindings, you can look here:

<http://appium.github.io/java-client/io/appium/java_client/AppiumDriver.html#findElementByAccessibilityId(java.lang.String)>