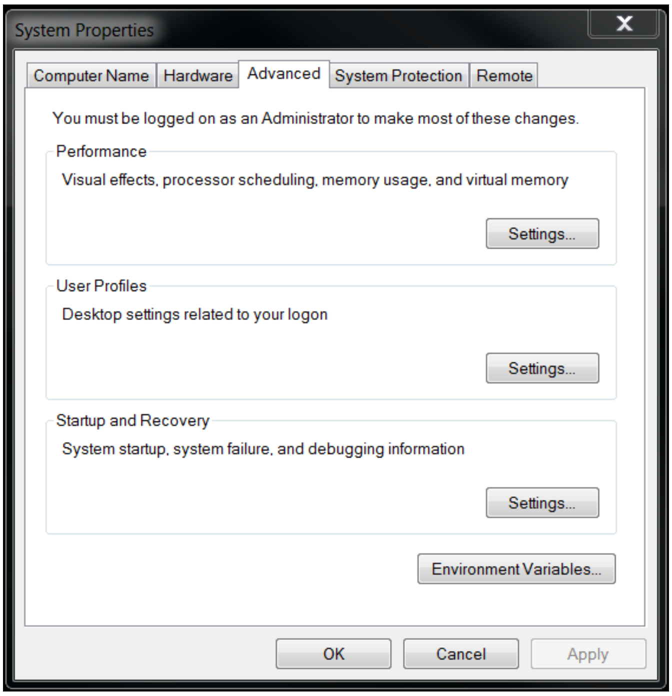
Appium environment setup for Windows and Android

1. **Java JDK 8**
   • Open Windows command line (type “cmd” in start menu search bar), run command: “java -version”
   • If you already have Java version 1.8 you can skip next steps. If command is not recognized or your version is other than 1.8 please proceed
   • Download file jdk-8u261-windows-x64.exe from Oracle website (may require registering but it’s recommended to use official source)
   • Install JDK with default settings using downloaded file, note the installation path (eg. “C:\Program Files\Java\jdk1.8.0_261”)
   • Open new command line window, run “java -version” again to make sure JDK is installed correctly

2. **IntelliJ IDEA**
   • Download current version of IntelliJ IDEA Community Edition from official website (version 2020.2 as of 8/14/2020)
   • Install IntelliJ IDEA with default settings from downloaded file

3. **Android Studio (with Android SDK)**
   • Download current version of Android Studio from official website (version 4.0.1 as of 8/14/2020)
   • Install Android Studio with default settings using downloaded file
   • Launch Android Studio (go with default settings, let the required files download)

4. **Environment variables**
   • Go to: Control Panel -> System and Security -> System -> Advanced system settings (for shortcut type: “system environment variables” in start menu search bar)



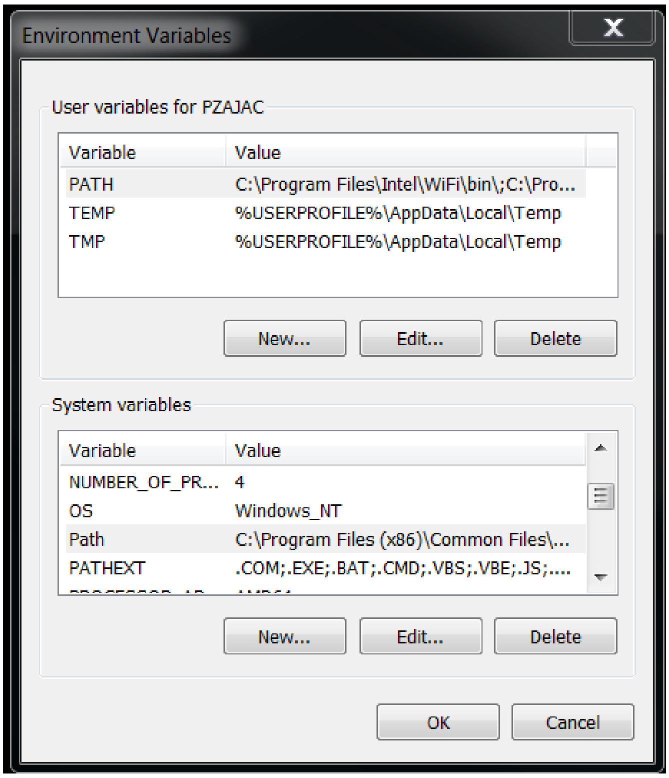
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your logon

Startup and Recovery
System startup, system failure, and debugging information

- Open “Environment Variables…” and under “System variables” choose “New…”



Environment Variables

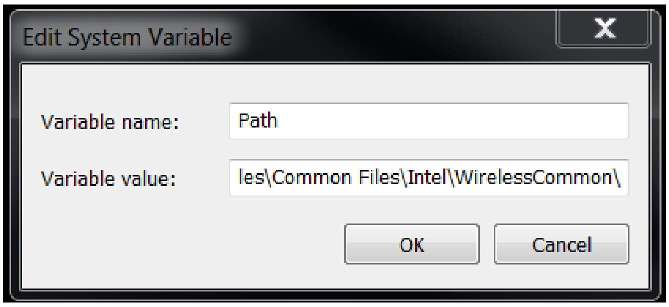
User variables for PZAJAC

Variable	Value
PATH	C:\Program Files\Intel\WiFi\bin\;C:\Pro...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
NUMBER_OF_PR...	4
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

- For variable name use “JAVA_HOME”, for value JDK installation path you noted earlier (eg. “C:\Program Files\Java\jdk1.8.0_261”), confirm
- In the same way create new variable “ANDROID_HOME”, with value pointing to Android SDK location (with default settings it should be at “C:\Users\your_user_name\AppData\Local\Android\Sdk”, confirm whether it’s there or somewhere else and use the correct path)
- Under “System variables” choose “Path” and click on “Edit…”



Edit System Variable

Variable name: Path

Variable value: les\Common Files\Intel\WirelessCommon\

- As you can see “Variable value” contains several paths separated by “;”. You need to add the following paths to it (mind the order):
“%JAVA_HOME%\bin”
“%ANDROID_HOME%\emulator”
“%ANDROID_HOME%\platform-tools”
“%ANDROID_HOME%\tools”
“%ANDROID_HOME%\tools\bin”

(These are just paths to folders starting with what you defined under JAVA_HOME and ANDROID_HOME so “%JAVA_HOME%\bin” = “C:\Program Files\Java\jdk1.8.0_261\bin”. Defining those paths will let you access files in these folders using command line opened at any location)

So basically you need to paste

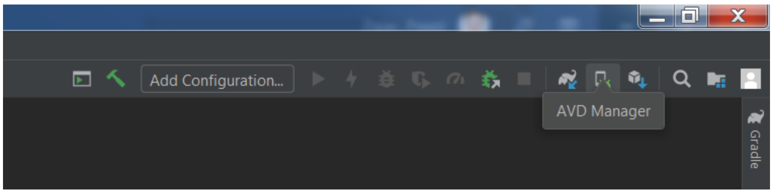
```
;%JAVA_HOME%\bin;%ANDROID_HOME%\emulator;%ANDROID_HOME%\platform-tools;%ANDROID_HOME%\tools;%ANDROID_HOME%\tools\bin
```

at the end of “Variable value”.
Be careful so you don’t copy any extra characters which may make Path impossible to read, if there is already “;” at the end don’t paste another one before %JAVA_HOME%.

- As check, in new command line window run the following commands to see if they are recognized:
“java -version”
“emulator -help-virtual-device”
“adb version”
“avdmanager”

5. Android Emulator

- Open Android Studio
- Start new Android Studio project with default settings (just to open project view)
- Choose AVD Manager from menu



- Follow the steps to create new virtual device:
Choose “Phone” category and a model, eg. “Pixel 2”
Select system image from recommended, preferably Android Oreo 8.1 (may require downloading)
Name your virtual device eg. “Pixel_2_API_27” and finish
- In new command line window run “emulator -list-avds”. You should see your virtual device listed
- Run command “emulator -avd VIRTUAL_DEVICE_NAME” using name of your listed device eg. “emulator -avd Pixel_2_API_27”. Emulator should start, this can take a while depending on your hardware. You can also run emulator from Android Studio UI, but running it from command line is more efficient than keeping Android Studio running just for emulator
- If you come across issues with message about no Virtualization Technology (HAXM , VT-x, AMD-V, SVM, Vanderpool) available it might be turned off on your computer. In that case try this guide

6. ADB connection with your device

- Turn off any open emulators
- In new command line window run “adb devices”. Device list should be empty at this point
- Enable USB debugging on your device using this guide
- With USB debugging enabled connect your phone with your computer using USB cable
- In new command line window run “adb devices”. If your device is listed you may skip remaining steps (name might be different from your device model)
- If device list is empty you may need extra drivers. Win 10 usually doesn’t have this problem but it’s quite common with 7. Download ADBDriver, open .exe file and follow instructions from this guide
- In new command line window run “adb devices”. Your device should be listed now
- At some point, on device screen you may see dialog asking if USB debugging should be allowed. Allow and remember the choice if you happen to see it

7. Node.js

- Download latest LTS installer for Windows from official site (version 12.18.3 as of 8/14/2020)
- Install Node.js with default settings using downloaded file
- In new command line window run “npm” to make sure Node.js is installed correctly

8. Appium

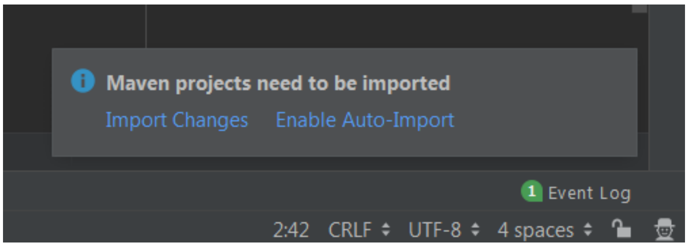
- In new command line window run “npm install -g appium@1.18”, wait for Appium to install
- In new command line window run “appium”, to make sure Appium is installed correctly

9. Appium-doctor

- In new command line window run “npm install appium-doctor -g”, wait for Appium-doctor to install
- In new command line window run “appium-doctor --android”, wait as it executes
- If you followed instructions all checks for necessary dependencies should be successful, don’t worry about checks for optional dependencies – you can install them later if you ever need them

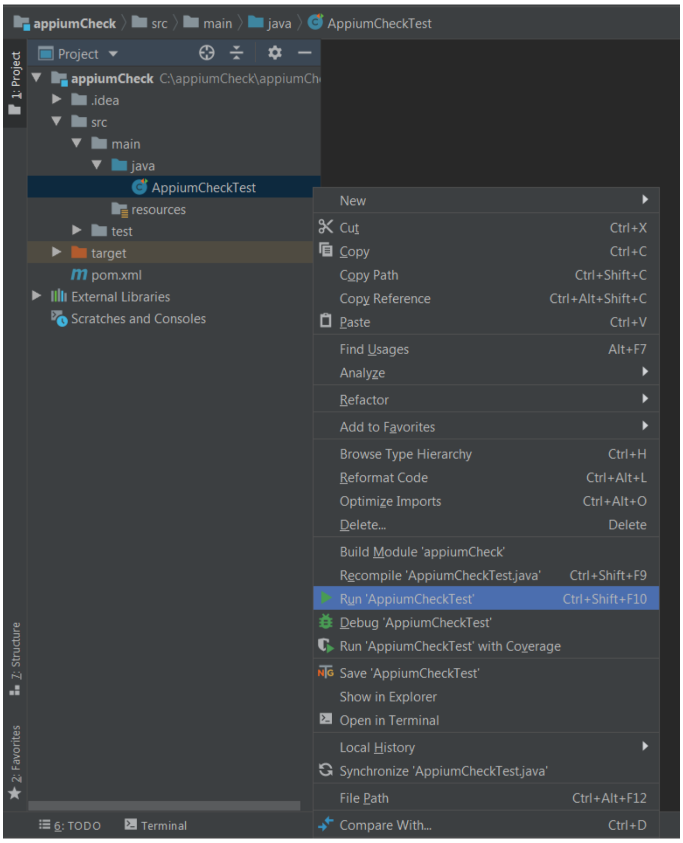
10. Final check

- Download and unzip this project
- Start IntelliJ IDEA, choose “Open” on welcome screen, point to folder with unzipped project
- Wait as project is opened and dependencies downloaded (messages at bottom bar), if you happen to see the below message in right bottom corner choose “Import Changes”



Maven projects need to be imported

- In new command line window run “appium --address 0.0.0.0 --port 4723”, wait for Appium server to start
- Make sure your Android device is plugged in, with USB debugging enabled (run “adb devices and see if it’s listed), device should be unlocked and awake
- After IDE finished its work on importing and downloading dependencies run AppiumCheckTest file as shown below



- The “test” will be executed on your device – you should see Gmail welcome screen which should close after ~10 seconds. If for some reason Gmail app doesn’t start don’t worry – in your version of Android Gmail may have different package or activity name. The most important thing is, you should see a lot of output in command line window where you started your Appium server – that’s computer communicating with your device
- After test executes you can try the same thing with emulator. Unplug your device, start emulator as in step 5 (give it a moment to launch) and run test the same way you did for real device. If you created emulator running Android Oreo this should definitely work

Good work, you are ready to start mobile test automation with Appium now!
Remember to bring your device and USB cable – your hardware may not be powerful enough to run emulator and test suites at the same time.