III. Appium

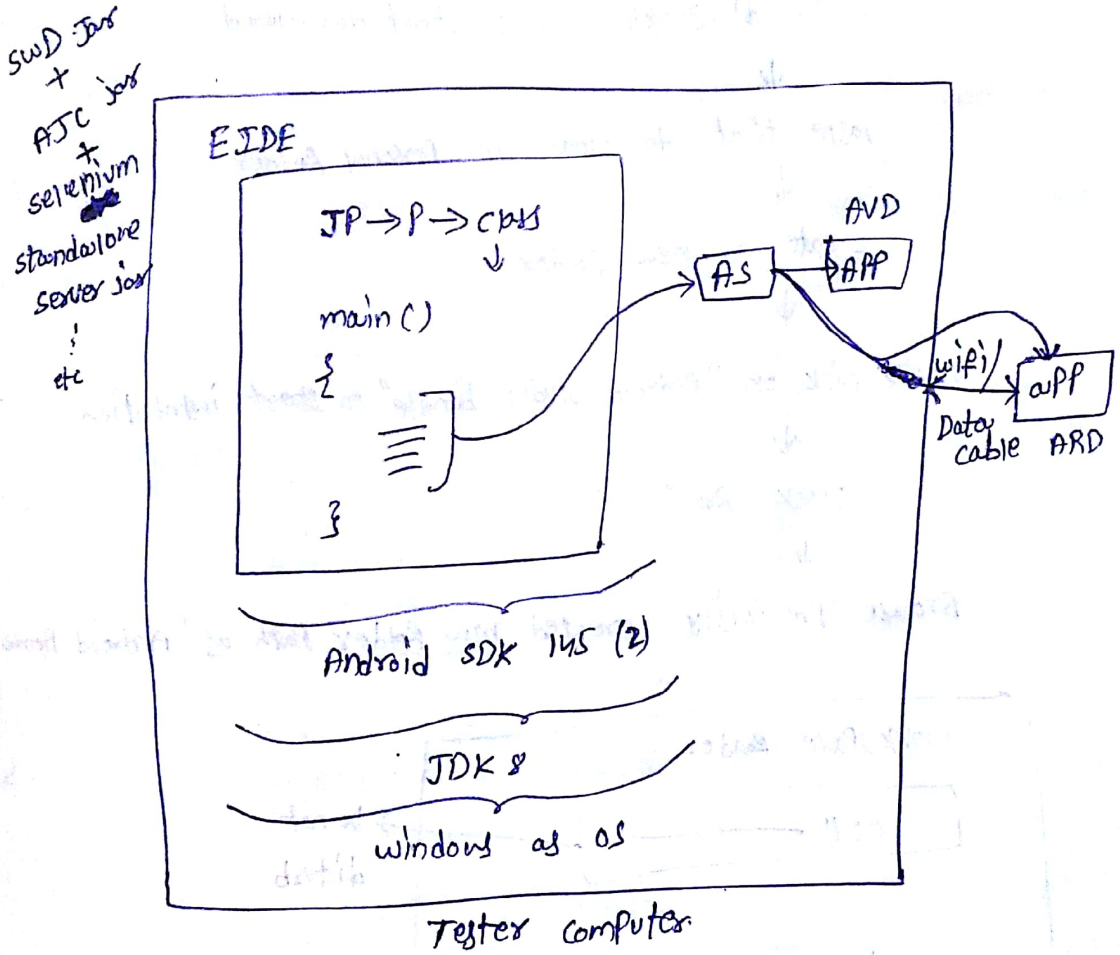
→ Developed by Don culler and extending by "srinivas sekhar"

→ Available as appium server (.exe) and appium java client (.jar)

a) Automate Android based apps in windows computer:-

→ Android based apps are two types, such as native apps and Hybrid apps.

→ when tester computer os is "windows", we need to follow below setup for Android based apps Test automation:



OS - operating system
JDK - Java development kit
SDK - software development kit
JP - Java Project
P - package

Jar - Java archive
AS - Appium server (.exe)
AJC - Appium Java client (Jar)
SWD - selenium webDriver (Jar)
AVD - Android virtual Device
ARD - Android Real Device

wifi - wireless fidelity

Step :- 1 :- Download and Install JDK 8 in tester computer

(Create JAVA_Home and extended "Path" variable)

Step :- 2 :- Download and Install Android SDK 2 (145 bundle)

Open Google site

↓

type "Android SDK bundle 145 for windows"

↓

Go to "dl.google.com" to start download

↓

Paste that download in Personal Folder

↓

Create a new Folder

↓

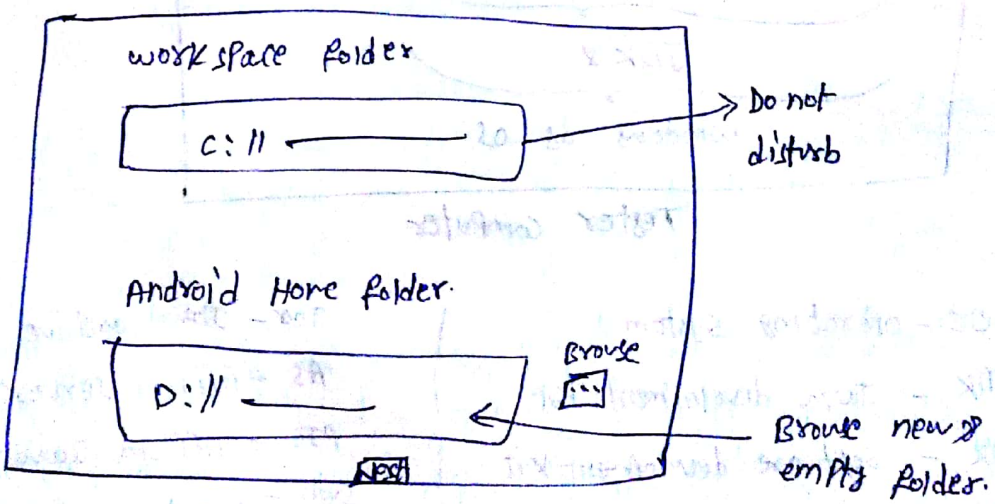
Double click on "Android studio bundle" to start installation

↓

Click "Run"

↓

Browse Previously created new Folder Path as "Android Home".



↓

Click 'Next' until Finish

↓

observe multiple subfolders and "AVD & SDK" manager in Personal folder. (E:\batch237\android folder)

↓

Right click on computer

↓

Properties

↓

Advanced system settings

↓

Environment variables

↓

click 'New' in system variables

↓

Enter "ANDROID_HOME" as variable name

↓

Paste Path of New folder (which was Android home) as value

↓

click "OK"

Ex:-

variable	ANDROID_HOME
value	E:\batch237\android

↓

select "Path" variable

↓

click "Edit"

↓

Add paths of 'tools", "build-tools", "Platform-tools" to end of Path variable value.

Ex:-

variable	Path
value	——— ;E:\batch237\sdk\android\tools ; E:\batch237\build-tools\24.0.3; E:\batch237\ Platform-tools ;

↓

Click "ok"

↓

Close "Properties" window

Step :-3:- AVD's creation

To automate android based apps using appium, we need to create AVD's with Android 4.2.2 (API level -17 Jelly bean) to latest. To install required API levels in tester computer for AVD's creation, we need to follow below navigation:

Go to Android SDK Home folder
(Ex:- E:\batch237\my android)

↓

Double click on "SDK manager"

↓

select API levels 17 to latest (oc-27)
(atleast 17)

↓

Click Install Packages

↓

Close SDK manager

↓

Go to Android SDK Home folder

↓

Double click on "AVD manager"

↓

Click create

↓

Enter "Details"

Ex:-

Device name → steveh

device model → Nexus_4

target → Android 4.2.2 (API-17)

CPU → ARM

Keyboard → Physical keyboard

Skin → Skin with Dynamic hardware control

RAM → 1000 MB

Heap → 64 mb

storage → 200 mB

:

etc

↓

Click "ok"

↓

select that created "AVD"

↓

Click "start".

↓

Click "Launch"

↓

Close "AVD" manager

↓

Follow above navigation to create multiple AVD's if required with different targets.

Note:-1 :- Depends on mobile app test plan, we need to create AVD's with various android versions.

Ex:-

version name	version number	API level
Jelly Bean	4.2.2	API level-17
Kitkat	4.4 to 4.4.4	19 to 20
Lollipop	5.0 to 5.1.1	21 to 22
Marshmallow	6.0 to 6.0.1	23
Nougat	7.0 to 7.1.2	24 to 25
oreo	8.0 to 8.1	26 to 27

Note:- 2 while installing API levels we can get prevention from Anti-virus and other software. In this situation corresponding API level Packages will go to "temp" folder in Android Home folder. After completion of installation we need to copy temp folder contents to 'tools' folders. (if "temp" folder contents are zip files we need to extract before copy to tools folder).

Step:-4 :- Connecting ARD

To work with Android Real Devices, we need to follow below navigation.

Go to settings in ARD

↓

Go to Developer options

(tap 6 times on model number/Build number/ s/w version in "about device")

↓

select "stay awake" and "USB debugging"

↓

Back to home in ARD

↓

connect ARD to tester computer using Data cable.

↓

observe your device is ready to use and autoplay window

↓

close Autoplay window

Note:1 After completion of AVD creation and ARD connection we need to run below commands on those devices

[adb - android debug bridge]

cmd:- adb devices ↵

To get list of connected devices (AVDs/ARDs)

→ For AVD's, id is emulator → 0000

→ For ARD, id is alphanumeric

cmd:- adb (Android debug bridge) -s xxxxxx (device id) shell pm (package manager) list packages -f

get list of all apps in given AVD/ARD

cmd:- adb -s shell pm list packages -f

only one device connected

cmd:- adb -s xxxxx shell pm list packages -f -s

get list of system (in-built) apps in given (AVD/ARD)

cmd:- adb -s xxxxx shell pm list packages -f -3

to get list of third party apps in given (AVD/ARD)

cmd:- adb -s xxxxx shell pm list packages -f -u

get list of unsigned/uninstalled apps in given (AVD/ARD)

cmd:- adb -s xxxx pull path of apk in mobile

cmd:- adb -s xxxxx install path of apk in compAdg

cmd:- adb -s xxxxx uninstall appPackage name

Ex:- adb -s 4200e6517eac8467 pull /data/app/com.watsapp-2/base.apk

(4200e6517eac8467 = ARD device id / device id; pull = for copy; /data/app/com.watsapp-2/base.apk = Path of .apk in ARD)

* adb -s emulator-5554 install c:\users\mindq-PC\base.apk

(emulator-5554 = AVD id; c:\users\mindq-PC\base.apk = Path of .apk in computer)

Note:-2 we are able to connect android real device (ARD) to tester computer via wifi by following below navigation

* Connect ARD (selected "Stay awake" and "USB debugging") in tester computer via Data cable.

* Make sure ARD and Tester computer are connected to the same wifi.

* Run below command to assign a Port number to ARD over tcp

Cmd → adb tcpip 5555

→ Disconnect that ARD from tester computer.

→ Get IP address of ARD by going through

"settings"
↓
wifi → your connected network
↓
longpress → get network settings
↓
Identify IP address

* Run this command to connect adb to your device over wifi using IP address:

cmd:- adb connect <your Phones ip addresses>:5555

verify, that adb works remotly through below command:

cmd:- adb devices.

Note:- 3 while working in Agile scrum Process we need to Participate in daily scrum meet. To share connected ARD screen to others in meeting, we need to run "ASM" (Android screen Monitor) or 'Droid". These are sharing connected mobile screens in computer desktop.

Ex:-

Go to google site
↓
Enter "Droid jar" download
↓
click 'search'
↓
Goto Droid @ screen .org site
↓
click on download (stable version)
↓
Paste that download in Personal Folder
↓
Rename to Droid
↓
Run below command

→ java -jar Path of droid.jar↵

step 5:- Get Eclipse IDE and required Jars to Associate

open Eclipse IDE
↓
Right click on Java Project
↓
Properties
↓
Java build Path
↓
Libraries
↓
Add External Jars
↓
Browse paths of SWD Jars (libs folder inside & outside)

and "appium java client" jar including selenium standalone server (jar) (available in seleniumhq.org site)

↓

click "ok"

Note:- 1 swd jars are available in seleniumhq.org site.

Note:- 2 To get appium java client jar we can go to

google.com

↓

Enter "appium java client 5.0.4" download

↓

click "search"

↓

Go to Maven Repository site

↓

click on jar link in "files"

↓

Paste that download in Personal folder

↓

Associate to java Project in eclipse

step:- 6 Install Appium server via Node JS

Go to google site

↓

Enter "nodejs" download

↓

Goto "Nodejs.org" site

↓

Click w.r.t os name & bit size

↓

Paste that download in personal folder

↓

Double click on that download

↓

Click "Next" until finish (Nodejs home creation & path variable extension are done)

↓

Go to command prompt

↓

cmd:- npm install -g appium → latest appium server

(or)

npm install -g appium@version number

npm → node package manager

Note:- 1 To install specific version of appium we need to run

cmd:- npm install -g appium@1.6 → version

Note:- 2 To uninstall appium we need to run

cmd:- npm uninstall -g appium

Step:-7 Developing Automation scripts.

After completion of test setup creation, we need to write automation code in eclipse IDE by using selenium webdriver and appium java client Jar classes like described below.

Ex:- (Start appium server) → static class in java.awt in JDK

Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k \"appium -a 0.0.0.0 -p 4723\" "); → To start & display cmd on desktop; 0.0.0.0 → local host

To start appium server at given ip address and port number.

Thread.sleep(20000);

URL u = new URL("http://0.0.0.0:4723/wd/hub"); (or) 127

URL → class in java.net package in JDK

address for appium server; /wd/hub → webdriver

(Launch APP in AVD ex) ARD)

'instance class in SWD

```
DesiredCapabilities dc = new DesiredCapabilities();
                          → SWD
dc.setCapability(CapabilityType.BROWSER_NAME, "");
dc.setCapability("deviceName", "4200C657");
dc.setCapability("PlatformName", "android");
dc.setCapability("PlatformVersion", "7.1.1");
```

4200C657 → device id from output of "adb devices"

android → Name of os in device

7.1.1 → version of os in device

// Launch APP activity

```
dc.setCapability("appPackage", "com.whatsapp");
dc.setCapability("appActivity", "com.whatsapp.Main");
```

To get "app Package" & "appActivity" names, we need to follow below ways:

.apk in computer

cmd:- adb -s xxx install ___.apk

To install → Path of apk file in computer

.apk in mobile

cmd

adb -s xxxx pull ___.apk

To copy to computer → Path of apk in mobile

---

cmd:- aapt dump badging ___.apk

To get "appPackage" & "appActivity" → .apk in computer

AndroidDriver driver = new AndroidDriver(u, dc);

- AndroidDriver → instance class in AIC Jar
- u → object of "URL" class → java.net in JDK
- dc → object of "DesiredCapabilities" class → SWD Jars

* while creating 'driver' object by using above code, corresponding appium server can launch corresponding app in specified device either in AVD (or) ARD.

* when app launching was completed and 'driver' object was created, we need to write automation code to locate elements, to operate elements & to observe elements in App screens. To get details of elements in app screen we need to use "uiAutomaterviewer". It can provide details for elements to form xPath.

go to cmd prompt

↓

Run below command

cmd:- uiautomaterviewer

↓

click on "device screenshot" icon for (static screen) & click on "device screenshot with compressed heirarchy" icon (dynamic screen) to get details of elements to form "xpath"

Ex:-1 In AVD

cmd:- adb -s emulator-5554 shell pm list packages -f

To get list of all apps paths

cmd:- adb -s emulator-5554 pull /system/app/Calculator.apk

↓

Path of apk in AVD

copy app .apk from AVD to computer

cmd:- aapt dump badging calculator.apk

To get "AppPackage" & "appActivity" names

aapt - Android Assert Packaging Tool

Pgm in AVD

```
// start appium server
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                    \" appium -a 0.0.0.0 -p 4723 \" ");
Thread.sleep(20000);
URL u=new URL("http://0.0.0.0:4723/wd/hub");
//Provide device details
DesiredCapabilities dc=new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
dc.setCapability("deviceName", "emulator-5554");
dc.setCapability("PlatformName", "android");
dc.setCapability("PlatformVersion", "4.2.2");
// provide app details.
dc.setCapability("appPackage", "com.android.calculator2");
```

```
dc.setCapability("appActivity", "com.android.calculator2.calculator");

// Launch app

AndroidDriver driver = new AndroidDriver(u, dc);

Thread.sleep(5000);

// Automate 9+6 = 15 and get text

driver.findElement(By.xpath("//*[@text='9']")).click();

driver.findElement(By.xpath("//*[@content-desc='Plus']")).click();

driver.findElement(By.xpath("//*[@text='6']")).click();

driver.findElement(By.xpath("//*[@content-desc='equals']")).click();
Thread.sleep(5000);
String x = driver.findElement(By.xpath("//*[@class='android.widget.EditText']
                              ")).getAttribute("text");

System.out.println(x);

driver.closeApp(); // close APP in AVD

Thread.sleep(5000);

// close appium server.

Runtime.getRuntime().exec("taskkill /F /IM node.exe");

Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

Runtime → static class in java.awt in JDK

taskkill → command

/F → forcibly

/IM → Image icon of s/w

cmd.exe → s/w name

Ex:-2

→ Take two multi digits numbers from keyboard

→ Launch calculator app in AVD

→ Enter 1st input

→ click Plus

→ Enter 2nd input

→ click equals

→ Get output and display

→ close app

Pgm

```
// Get data from keyboard

Scanner sc = new Scanner(System.in);
System.out.println("Enter input1");
String x = sc.nextLine();
System.out.println("Enter input2");
String y = sc.nextLine();

// start appium server

==

// Provide device details

==

// Provide app details

==

// Launch app
AndroidDriver driver = new AndroidDriver(u, dc);
Thread.sleep(5000);

// Enter input 1
for(int i=0; i<x.length(); i++)
{
    char c=x.charAt(i);
    driver.findElement(By.xpath("//*[@text='"+c+"']")).click();
    Thread.sleep(5000);
```

instance class in java.util in SDK

```
}
driver.findElement(By.xPath("//*[@content-desc='plus']")).click();
Thread.sleep(5000);
// Enter input 2
for(int i=0; i<y.length(); i++)
{
    char c = y.charAt(i);
    driver.findElement(By.xPath("//*[@text='"+c+"']")).
                                                    click();
    Thread.sleep(5000);
}
driver.findElement(By.xPath("//*[@content-desc='equals']")).click();
Thread.sleep(5000);
string x = driver.findElement(By.xPath("//*[@class='android.widget.
                EditText']")).getAttribute("text");
system.out.println(x);
// close app
driver.closeAPP();
Thread.sleep(5000);
// stop Appium server
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

Note- whenever connected ARD is not recognized by tester computer, we need to follow below ways:

way:-1 connect ARD to computer using Data cable

(In ARD, USB debugging & Stay awake were already selected in settings)

↓

search "driver manager" to get "device manager" in computer.

↓

go to open "device manager"

↓

Right click on connected ARD name

↓

Click on install device drivers

way:-2 Download & Install USB drivers related to mobile manufacture.

Ex:- samsung kies for samsung

→ micromax USB drivers.

way-3:- Download and Install PDA net drivers for Android.

Ex:- → Launch calculator in ARD

→ Enter input 1

→ Click plus

→ Enter input 2

→ Click equal

→ Get output

→ Validate output = ip1+ip2

→ True (Test Passed)

→ false (Test failed and get screenshot with filename as todays date)

cmds

→ adb devices → To get ARD devices

→ adb -s xxxx shell pm list packages -f > calc.txt
device id

To get list of all apps in ARD

→ adb -s xxxx Pull /system/priv-app/secCalculator.apk

To copy corresponding app from ARD to computer.

→ aapt dump badging secCalculator_N.apk

To get "appPackage" and "appActivity" names

Pgm

① // start appium server

Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
\"appium -a 0.0.0.0 -p 2020\" ");

Thread.sleep(30000);

URL v=new URL("http://0.0.0.0:2020/wd/hub");

// Get data from keyboard

② // Provide device details

DesiredCapabilities dc=new DesiredCapabilities();

dc.setCapability(CapabilityType.BROWSER_NAME, "");

dc.setCapability("deviceName", "device id");

dc.setCapability("PlatformName", "android");

dc.setCapability("PlatformVersion", "6.0.1");

// Provide app details

dc.setCapability("appPackage", "com.sec.android.app.PopupCalculator");

dc.setCapability("appActivity", "com.sec.android.app.PopupCalculator.Calculator");

② //Get data from Keyboard

```
Scanner sc=new Scanner(System.in);
System.out.println("Enter input1:");
String x=sc.nextLine();
System.out.println("Enter input2:");
String y=sc.nextLine();
```

③ //Launch app

```
AndroidDriver driver=new AndroidDriver(u,dc);
Thread.sleep(5000);
//Enter input1
for(int i=0; i<x.length(); i++)
{
    char c=x.charAt(i);
    driver.findElement(By.xpath("//*[@text='"+c+"']")).click();
    Thread.sleep(5000);
}
driver.findElement(By.xpath("//*[@content-desc='plus']")).click();
Thread.sleep(5000);
//Enter input2
for(int i=0; i<y.length(); i++)
{
    char c=y.charAt(i);
    driver.findElement(By.xpath("//*[@text='"+c+"']")).click();
    Thread.sleep(5000);
}
driver.findElement(By.xpath("//*[@content-desc='Equal']")).click();
```

```
Thread.sleep(5000);
//Get output
String z= driver.findElement(By.xPath("//*[@class='android.widget.EditText']"
                                    )).getAttribute("text");
Thread.sleep(5000);
//Validate output
 int a= Integer.parseInt(x);
 int b= Integer.parseInt(y);
 int c= Integer.parseInt(z);
 if(c==a+b)
 {
    System.out.println("Addition test passed");
 }
else
 {
     System.out.println("Addition test failed");
    Dateformat df=new SimpleDateFormat("dd-MM-yy-hh-mm-ss");
     Date d=new Date();
    String imagename=df.format(d); //today date with time
    File src=driver.getScreenshotAs(OutputType.FILE);
    File dest=new File("E:\\batch237\\"+imagename+".png");
    FileUtils.copyFile(src,dest);
 }
 //close app
  driver.closeApp();
  Thread.sleep(5000);
 // stop appium server
   Runtime.getRuntime().exec("taskkill /F /IM node.exe");
```

Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");

In above code, The classes like DateFormat (interface), SimpleDateFormat (instance class) and Date are related to JDK.

"FileUtils" class related to "commons.io" jar.

Note:- when AVD or ARD was connected correctly (adb devices cmd is showing deviceid), but uiautomatorviewer is not able to get screen of app in device to provide details for elements, we need to follow:

way:-1 → close appium server by closing corresponding cmd Prompt window

way-2 → close app in device opened through code execution and we need to open app manually before using uiautomatorviewer.

way-3 → Run below commands before using uiautomatorviewer.

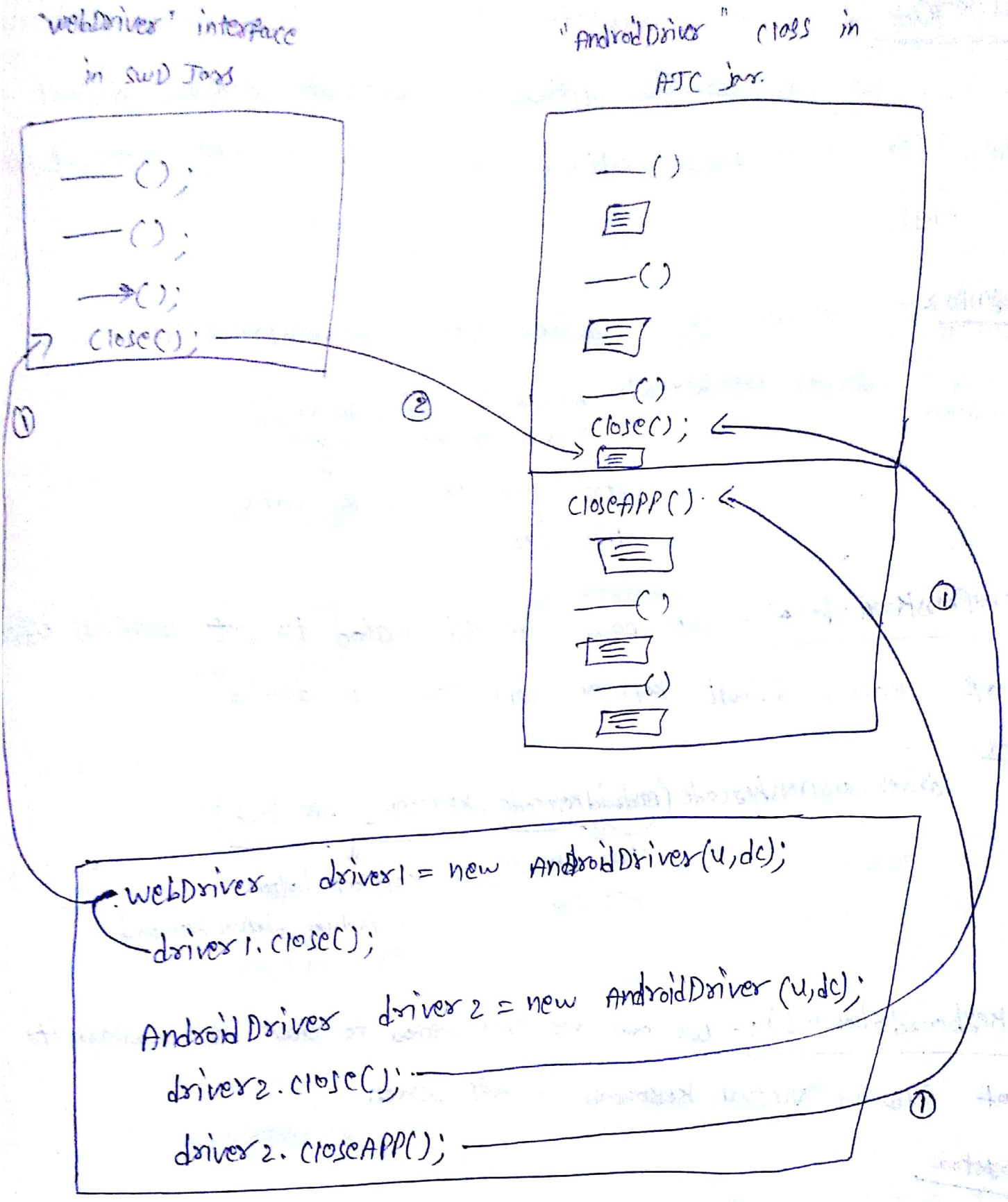
cmd:- adb kill-server ↵

adb start-server ↵

way-4 → Restart device (ARD)

b) "AndroidDriver" class methods:-

"AndroidDriver" class in "appium java client jar" have bodies to methods in "WebDriver" interface in SWD Jars and have some extra methods, called as appium bindings.



From the above diagram "AndroidDriver" class consists of bodies to methods declarate in "WebDriver" interface and have bodies to extra methods described below:

PressKeyCode :-

We can use this method to automate Android default keys. (Ex:- Home, Back, settings, Enter... etc) on app screen in AVD or ARD.

syntax:- AndroidDriver driver = new AndroidDriver(u, dc);

driver.PressKeyCode(AndroidKeyCode.xxxx);

- AndroidKeyCode → static class in AJC jar
- xxxx → key name

longPressKeyCode(•):- we can use this method to get longPress effect of Android default keys on app screen in device.

syntax

driver.longPressKeyCode(AndroidKeyCode.KEYCODE_CAPS_LOCK);

- AndroidKeyCode → static class in AJC jar
- KEYCODE_CAPS_LOCK → caps lock button in Android virtual keyboard

isKeyboardShown():- we can use this method to check the availability of Android virtual keyboard in app screen.

Syntax

```
if (driver.isKeyboardShown())
{
   ==
}
else
{
   ==
}
```

hideKeyboard():- we can use this method to hide visible keyboard in app screen in device.

syntax

```
driver.hideKeyboard();
```

launchAPP():- we can use this method to relaunch initial app again in device. (mentioned in desired capabilities)

syntax

```
driver.launchAPP();
```

closeAPP():- we can use this method to close initial app (mentioned in desired capabilities) in device.

syntax

```
driver.closeAPP();
```

getCurrentPackage():- we can use this method to get package name of currently running app.

syntax

```
string x = driver.getCurrentPackage();
system.out.println(x);
```

getCurrentActivity():- we can use this method to activity name of currently running app.

syntax

```
String x = driver.getCurrentActivity();
system.out.println(x);
```

runAPPInBackground():- we can use this method to send currently running app to background for given time.

syntax

```
Duration d= Duration.of(10, ChronoUnit.SECONDS);
```

Duration: Singleton class in jdk

ChronoUnit: static class in SDK

```
driver.runAPPInBackground(d);
```

resetAPP():- we can use this method to restart initially opened app (mentioned in desired capabilities).

syntax

```
driver.resetAPP();   //when app in active
```

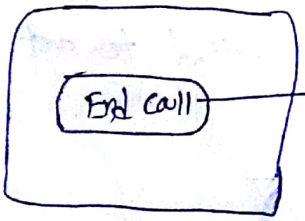
(or)

```
driver.launchAPP();
driver.resetAPP();
```

} when app in background.

FindElementByAndroidUIAutomator():- we can use this method to locate elements in dynamic screens of app in device.

Ex:-



dynamic screen

syntax

```
driver.findElementByAndroidUIAutomator("new UiSelector().text("End call").click();
```

new: Android code to locate element

.click(): click()

(or)

```
driver.findElementByAndroidUIAutomator("new uiselector().
    textStartsWith(\"End\")").click();
```

(or)

```
driver.findElementByAndroidUIAutomator("new uiselector().
    textContains(\"nd c\")").click();
```

(or)

```
driver.findElementByAndroidUIAutomator("new uiselector().
    textMatches("\"[E][a-z]+[\s][c][a-z]+
    \")").click();
```

Android code with regular expression

(or)

```
driver.findElementByAndroidUIAutomator("new uiselector().description
    (\"End call\")").operation();
```

description — content-desc attribute is available for element

(\"End call\") — Android code

(or)

```
driver.findElementByAndroidUIAutomator("new uiselector().resourceId
    (\"___\")").operation();
```

(or)

```
driver.findElementByAndroidUIAutomator("new uiselector().text("\"-\").
    index("\"-\")").operation();
```

Exp. 4

Pgm

→ Launch "Phone" app in ARD
→ Dial a number
→ click end call
→ close app

```
//Get mobile number from Keyboard
Scanner sc= new Scanner (System.in);
System.out.println ("Enter mobile number:");
String x= sc.nextLine();
//start appium server
=
try{
//Details of ARD and app in testNG
=
//Launch app
AndroidDriver driver= new AndroidDriver (u, dc);
Thread.sleep (10000);
for (int i=0; i<x.length(); i++)
{
    char ch= x.charAt(i);
driver.findElement(By.xpath("//*[@text='"+ch+"'][@index='0']")).
click();
    Thread.sleep (5000);
}
driver.findElement (By.xpath ("//*[@content-desc='call'][@index='0']")).click
Thread.sleep (10000)
```

```
driver.findElementByAndroidUIAutomator("new UiSelector().description("End call")").
                                        (content-desc)                    click();

// close app
  driver.closeApp();
catch (Exception e)
    {
        System.out.println(e.getMessage());
    }

// close appium server
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

Note:—

App screen	element	coding to locate
static	static	findElement() with ~~By.xpath()~~ with all attributes
static	Dynamic	findElement() with By.xPath() with "resource-id" value
Dynamic	static	findElementByAndroidUIAutomator() with Android code with any attribute value
Dynamic	Dynamic	findElementByAndroidUIAutomator() with Android code with "resource-id" value.

findElementsByAndroidUIAutomator()

we can use this method to collect multiple elements in app screen.

syntax

List<MobileElement> l = driver.findElementsByAndroidUIAutomator("_____");

- List: class in jdk (java.util)
- MobileElement: class in AJC jar
- "_____": Android code to locate dynamic elements in static app screen or static elements in dynamic app screen or dynamic elements in dynamic screens.

findElementByAccessibilityId()

we can use this method to locate an element in mobile app screen like findElement() and findElementByAndroidUIAutomator().

syntax

driver.findElementByAccessibilityId("xxxx").operation();

- xxxx: resource-id value of an element

ex:- com.android.vending:id/search_box_idle_text

- com.android.vending: class name
- search_box_idle_text: AccessibilityId (unique to elements)
- com.android.vending:id/search_box_idle_text: resource-id value of an element

findElementsByAccessibilityId()

we can use this method to collect multiple elements in a/f screen.

Syntax

```
List<MobileElement> l = driver.findElementsByAccessibilityId("xxxx");
```

"xxxx" — id value in resource-id of elements.

note 1:- The above two methods are useful to locate elements when those elements have "resource-id".

note 2:- mobileElement in AJC jar is similar to webElement class in SWD Jars.

note 3:- In mobile apps, we are able to locate elements by using:

In AndroidDriver class:
- → findElement() } related to "webDriver" interface
- → findElements() } related to "webDriver" interface
- → findElementByAndroidUIAutomator()
- → findElementsByAndroidUIAutomator()
- → findElementByAccessibilityId()
- → findElementsByAccessibilityId()

getOrientation()

we can use this method to get orientation of current app screen.

Ex:- Landscape (or) Portrait.

rotate():- we can use this method to change orientation of current app screen.

Pgm

```
// start appium server

try
{
    // Get Details of AVD or ARD

// Launch site
AndroidDriver driver = new AndroidDriver(U, dc);
Thread.sleep(5000);
String x=driver.getOrientation().name();
System.out.println(x);
if(x.equals("PORTRAIT"))
  {
     driver.rotate(ScreenOrientation.LANDSCAPE);
  }
Thread.sleep(10000);
driver.closeApp();
// close and stop appium server
```

isLocked() / lockDevice()

we can use this method to know whether mobile screen is locked or not.

~~driver.lockDevice();~~

lockDevice():- we can use this method to lock the device.

unlockDevice():- we can use this method to unlock the locked device.

Pgm

```
driver.lockDevice();
Thread.sleep(10000);
if(driver.isLocked())
{
    System.out.println("Locked");
    driver.unlockDevice();
}
```

installAPP() :- we can use this method to install app in to device by giving .apk file

Syntax

```
driver.installAPP("Path of app.apk");
```

android Package kit

isAPPInstalled () :- we can use this method to check that given app is installed or not.

Syntax

```
if (driver.isAPPInstalled ("APPPackage name")) -
{
    ==
}
else
{
    ==
}
```

startActivity () :- we can use this method to open specific activity of given app.

Syntax

```
Activity a = new Activity ("app Package name", "appActivity name");
driver.startActivity (a);
```

removeApp () :- we can use this method to uninstall specify app from device.

```
driver.removeAPP ("appPackage name");
```

Pgm

```
// install app and start before remove
driver.installApp("C:\\users\\mindv.Pc\\com.whatsapp_2.apk");
Thread.sleep(5000);
if(driver.isAppInstalled("com.whatsapp"))
{
    Activity a=new Activity("com.whatsapp","com.whatsapp.Main");
    driver.startActivity(a);
    Thread.sleep(50000);
    driver.removeApp("com.whatsapp");
}
```

(Activity → AUT Jar)

setLocation() :- we can use this method to change map location from current to specified.

Syntax

```
Location l=new Location(latitude, longitude, Altitude);
driver.setLocation(l);
```

(Location → SWD Jar)

latitude, longitude: from internet website for our required location

Altitude: value in feets to set zoom level of map

Ex:- latlong.net website

Pgm

```
// start appium server

try
{
    // details of device & app details
    DesiredCapabilities dc=new DesiredCapabilities();
    dc.setCapability(CapabilityType.BROWSER_NAME,"");
    dc.setCapability("deviceName","xxxx");
    dc.setCapability("PlatformName","android");
```

```
dc.setCapability("PlatformVersion", "6.0.2");
dc.setCapability("locationServicesEnabled", true);
dc.setCapability("locationServicesAuthorized", true);
dc.setCapability("appPackage", "xxxx");
dc.setCapability("appActivity", "xxxx");

// Launch app

AndroidDriver driver = new AndroidDriver(u, dc);
Thread.sleep(5000);
// set location to Banglore

Location l1 = new Location(latitude, longitude, Altitude);
                          [12.971223, 77.55456, 2000]
driver.setLocation(l1);
Thread.sleep(20000);
// set location to Vijayawada

Location l2 = new Location(xxxx, xxxxx, xxxx);
driver.setLocation(l2);
Thread.sleep(20000);
driver.resetAPP();
Thread.sleep(10000);
driver.closeAPP();
}
catch(Exception e)
{
    System.out.println("e.getMessage());
}
// close appium server
```

Ex~ Automation code without "Thread.sleep()"

Prgm

```
//start appium server
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                          \"appium -a 0.0.0.0 -p 5120\"");
URL u = new URL("http://0.0.0.0:5120/wd/hub");
//Provide device and app details
DesiredCapabilities dc = new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
dc.setCapability("deviceName", "emulator-5554");
dc.setCapability("PlatformName", "android");
dc.setCapability("PlatformVersion", "4.2.2");
dc.setCapability("appPackage", "xxxx");
dc.setCapability("appActivity", "xxxx");
// create driver object to launch app in AVD
AndroidDriver driver;
while(2>1) //infinite loop
{
    try
    {
        driver=new AndroidDriver(u,dc);
        break; //terminate from loop
    }
    catch(Exception e)
```

```
    {
    }
}
// Automate elements  9+5=
    try
    {
        driver.manage().timeouts().implicitlyWait(100, TimeUnit.SECONDS);
        driver.FindElement(By.xPath("//*[@text='9']")).click();
        driver.FindElement(By.xPath("//*[@content-desc='Plus']")).click();
        driver.FindElement(By.xpath("//*[@text='5']")).click();
        driver.FindElement(By.xpath("//*[@content-desc='equals']")).click();
        string x=driver.FindElement(By.xPath(
                    "//*[@class='android.widget.EditText']"))
                        .getAttribute("text");
        System.out.Println(x);
        // close app
        driver.closeAPP();
    }
catch(Exception e)
    {
        System.out.println(e.getMessage());
    }
// stop and close appium server.
```

```
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

Note:- To get "appPackage" and "appActivity" names for an app, we can use "aapt" command. we are able to use "apkinfo" app in device to get app details like appPackage, appActivity ... etc.

goto "Playstore"
↓
Enter "apkinfo"
↓
click search
↓
"APK info app" download & install
↓
open "apk info"
↓
select required app
↓
Go to Activities
↓
Identify launchable activities (appname will come as headings for launchable activities)

Ex:- com.whatsapp.Main
appPackage
appActivity

getContextHandles() :- we can use this method to get contexts related to corresponding app. In general, some apps running with "NATIVE_APP" context. some apps are running with "NATIVE_APP" and "WEBVIEW_xxxx" contexts.

getContext():- we can use this method to get current context of the app.

isBrowser():- we can use this method to known whether app in "WEBVIEW" context or not?

context():- we can use this method to change context.

Ex:-

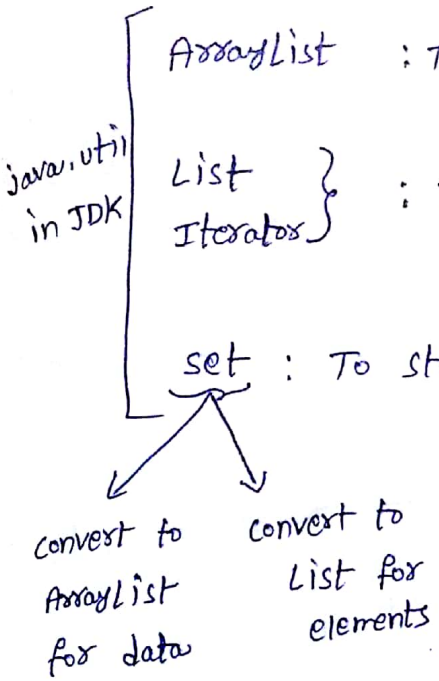
```
// get context related to app
Set s = driver.getContextHandles();
ArrayList<String> al = new ArrayList<String>(s);
for(int i=0; i<al.size(); i++)
{
    System.out.println(al.get(i));
}
// get current context
String x = driver.getContext();
System.out.println(x);
// change context
if(driver.isBrowser())
{
    driver.context("NATIVE_APP");
}
```

Note 1:- Above concept is useful while automating:

→ Hybrid app, which is redirect to Browser

→ Browser page, which is redirect to app.

Note 2:- In java language, we are able to use collections in different ways:



getRemoteAddress():- we can use this method to get details of currently running appium server.

getDeviceTime():- we can use this method to get time of the device.

getPlatformName():- we can use this method to get Platform name of active device.

getDisplayDensity():- we can use this method to get active device display density value.

openNotifications():- we can use this method to open notification screen in device.

Pgm

```
// get remote address
    URL    x = driver.getRemoteAddress();
System.out.println("Appium server url is " +x.getPath());
System.out.println("Appium server Port is " +x.getPort());
System.out.println("Appium seaves Protocol is " +x.getProtocol());
// get DeviceTime()
    String  y = driver.getDeviceTime();
    system.out.println("Device time is " +y);
    //get PlatformName()
    String  w= driver.getPlatformName();
    System.out.println("Platform Nave is " +w);
    //get Display Density()
     long   l = driver.getDisplayDensity();
     System.out.println("Display density is " +l);
    // open Notifications()
      driver.openNotifications();
    WebDriverwait   wait = new WebDriverwait(driver, 100);
    wait.until(ExpectedConditions.VisibiliTy of ElementLocated(
                    By.xPath("//*[@text='airtel'][@index='2']")));
     driver.PressKeyCode(AndroidKeyCode.BACK);
     // close app
     driver.closeAPP();
    // stop appium server
     Runtime.getRuntime.exec("taskkill  /F /IM  node.exe");
     Runtime.getRuntime.exec("taskkill  /F /IM  cmd.exe");
```

getSupportedPerformanceDataTypes() :- we can use this method to get Performance type names related to app.

Ex:- cpuinfo, memoryinfo, networkinfo, batteryinfo

getPerformanceData() :- we can use this method to get required performance information of app within given duration.

syntax List<List<object>> l = driver.getPerformanceData("appPackage name", "info", duration);

"info" → memory info, battery info, cpu info, n/w info.

duration → in msec

Pgm

```
List<String> Pl = driver.getSupportedPerformanceDataTypes();
for(int i=0; i<Pl.size(); i++)
{
    System.out.print(Pl.get(i) + "\t");
}
System.out.println();
// get PerformanceData
System.out.println("CPU information:");
List<List<object>> ci = driver.getPerformanceData("appPackage name",
                                                   "cpuinfo", 10000);
for(int i=0; i<ci.size(); i++)
{
    for(int j=0; j<ci.get(i).size(); j++)
    {
        try
```

```
        {
            System.out.Print(ci.get(i).get(j).toString()+"\t");
        }
    }
    Catch (Exception ex)
    {
        System.out.Print(e+"\t");
    }
}
System.out.println();
}
```

Note 1:- The above example code is useful to validate performance criteria of an app interms of "cpu info, meminfo, Networkinfo, batteryinfo".

Note 2:- Instead of above like code, we are able to run below commands to get performance criteria of an app.

cmds

→ adb shell dumbsys battery

→ adb shell dumbsys wifi

→ adb shell dumbsys cpuinfo

→ adb shell dumbsys meminfo 'app Package name'

→ add -s xxxx ('device id') to above commands when more devices are connected to tester computer.

Note 3:- while automating apps using appium coding, we need to provide corresponding app "appPackage and appActivity" names. For it, we have 3 ways:

way-1:- * aapt dump badging Path of .apk ↵

way-2:- Download and install "APK info" app to device (android device)

↓

open "APK info" in device

↓

select required app and observe "activities" list.

way-3:- open required app in device

↓

Run below command;

" adb -s xxxx shell dumpsys window windows "

↓

goto last but one line in output of above cmd to get "appPackage & "appActivity".

getConnection():- we can use this method to get details related to wifi connection, Mobile data connection and Airplane mode.

Pgm

instance class.

```
Connection c = new driver.getConnection();
if(c.compareTo(connection.AIRPLANE)==0)
{
    System.out.println("AIRPLANE is ON");
}
```

static properties in connection class

```
else
 {
    system.out.println ("AIRPLANE is OFF");
 }
if (C.compareTo (connection. DATA) == 0)
 {
    system.out.println ("DATA is ON");
 }
else
 {
    system.out.println ("DATA is OFF");
 }
if (C.compareTo (connection. WIFI) == 0)
 {
    system.out.println ("WIFI is ON");
 }
else
 {
    system.out.println ("WIFI is OFF");
 } // driver .setConnection (connection.WIFI);
```

static properties in connection class

[0 - for ON
non-0 for OFF]

setConnection() :- we can use this method to 'ON' data connection, wifi connection and Airplane mode if required.

Ex:-

```
// Turn just wifi on
driver. setConnection (connection. WIFI);
// Turn just data on
driver. setConnection (connection. DATA);
```

→ Class in appium java client jar

// Turn just airplane on

driver.setConnection (Connection.AIRPLANE);

(It is last interaction with device)

// turn off all connections (data and wifi)

driver.setConnection (Connection.NONE);

// Turn on all connections (data and wifi)

driver.setConnection (Connection.ALL);

c) using "bounds" attribute to locate static element in app screen:-

To locate static elements in app screen, we can use "findElement()" method with "xpath" of elements. But sometimes, given "xpath" can match with more than one element. To solve this issue, we can use:

way-1:- driver.findElement (By.xpath ("(//*[@xxxx='xxx'])[index]")).operation();

index value complex to guess.

way-2:- adding extra attribute to xpath.

way-3:- using "bounds" attribute in xpath.

driver.findElement (By.xpath ("//*[@bounds='[x,x][x,x]']")).operation();

** d) Using "uiautomator2"

while automating apps (Native (or) Hybrid) in ARD's which have android version 7 (or) more, we can get appium server problem like finding element with "xpath" and appium server can goto silent

To solve this issue, we need to use uiautomator 2. For it we need to run below command:

cmd:- npm install appium-uiautomator2-driver

npm → node package manager (related to nodejs)

After sucessful installation, we need to all extra capability to strip.

dc.setCapability("automationName", "uiautomator2");

e) working with "Toasted messages"

In gereral, mobile apps are providing messages which are staying less time in screen. This type of messages are called as toasted messages.

To handle toasted messages in automation, we need to follow different logic.

step 1:- Take screenshot of app screen which have toasted message.

step 2:- Apply OCR (optical character recognition) on screenshot image to capture text.

we need to follow below navigation to configure "tessj". It is available as a set of jars:

Go to google site

↓

Enter Tessj 3.0 jar download

↓

Go to jar download.com site

↓

click on Download Tess4j

↓

Paste that download in personal folder.

↓

Extract the download to get jars

↓

Right click on project in eclipse IDE

↓

java build path

↓

~~Add External~~ Libraries

↓

click Add External jars

↓

~~Brow~~ Browse path of previously extracted jars

↓

Click 'OK'.

while using "Tess4j" for capture text from image (OCR → optical character Recognization), we need to use below classes objects and methods:

1) `LoadLibs.extractTessResources("tessdata");`

LoadLibs → static class in Tess4j Jars

"tessdata" → Default folder name, which folder have resources for OCR.

2) `Tesseract obj = new Tesseract();`

Tesseract → instance class in Tess4j Jars

3) `obj.setDataPath("Path of resources folder");`

4) `String x = obj.doOCR(image file object);`

Prgm (selenium) "Take a text from image file"

Ex:-1

```
// Take any image file, which have text in content
File f = new File("E:\\untitled.png");
// Load "tessdata" file
File fo = LoadLibs.extractTessResources("tessdata");
Tesseract obj = new Tesseract();
obj.setDatapath(fo.getAbsolutePath());
// convert image content as text
~~String s=0;~~
String result = obj.doOCR(f);
Thread.sleep(20000);
System.out.println(result);
```

Ex:-2

```
// Get mobile number from keyboard
Scanner sc = new Scanner(System.in);
System.out.println("Enter a mobile number");
String x = sc.nextLine();
// start appium server
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /K
                          \"appium -a 0.0.0.0 -p 4723\"");
URL u = new URL("http://0.0.0.0:4723/wd/hub");
// Details of ARD and AUT (App under Testing)
DesiredCapabilities dc = new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
```

```
dc.setCapability("deviceName", "xxxx");
dc.setCapability("PlatformName", "android");
dc.setCapability("PlatformVersion", "xxxx");
dc.setCapability("appPackage", "xxxx");
dc.setCapability("appActivity", "xxxx");
// create driver object to launch app in ARD
AndroidDriver driver;
while(2>1)
    {
      try
       {
          driver = new AndroidDriver(u, dc);
          break; //terminate from loop
       }
      catch(Exception e)
       {

       }
}
// dail number and activate call
try
 {
   driver.manage().timeouts().implicitlyWait(100, TimeUnit.SECONDS);
    for(int i=0; i<x.length(); i++)
    {
       char d = x.charAt(i);
       driver.findElement(By.xpath("//*[@class='android.widget.TextView']
                          [@text='"+d+"'][@index='0']")).click();
}
```

```
driver.findElement(By.xpath("//*[@content-desc='call'][@index='0']")).click();
WebDriverWait w=new WebDriverWait(driver,100);
w.until(ExpectedConditions.visibilityOf(driver.findElementByAndroidUIAutomator
        ("new UiSelector().description(\"End Call\")")));
// Take screenshot
File src=driver.getScreenshotAs(outputType.FILE);
File dest=new File("toastedreg.png");
FileUtils.copyFile(src,dest);
// click End call
driver.findElementByAndroidUIAutomator("new UiSelector().description(\"End call
                                        ").click();
driver.closeApp();
//Get toasted message (3.0 is stable for tess4j)
File tessDataFolder=LoadLibs.extractTessResources("tessdata");
Tesseract obj=new Tesseract();
obj.setDatapath(tessDataFolder.getAbsolutePath());
String result=obj.doOCR(dest);
Thread.sleep(10000);
System.out.println(result);
if(result.contains("conditional call for"))
  {
     system.out.println("Test Passed");
  }
else
  {
     system.out.println("Test Failed");
}
```

```
}
catch (Exception ex)
    {
        System.out.println(ex.getMessage());
    }

// stop and close appium server
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

f) TouchAction class Methods:

TouchAction is related to appium java client jar. This class methods are useful to automate touch related operations on app screens in mobiles on AVD (or) ARD.

Press():- we can use this method to get press effect on an element.

Syntax 1)

```
WebElement e=driver.findElement(By.xPath("xxxx"));
TouchAction ta=new TouchAction(driver);
ta.press(e).perform();
```

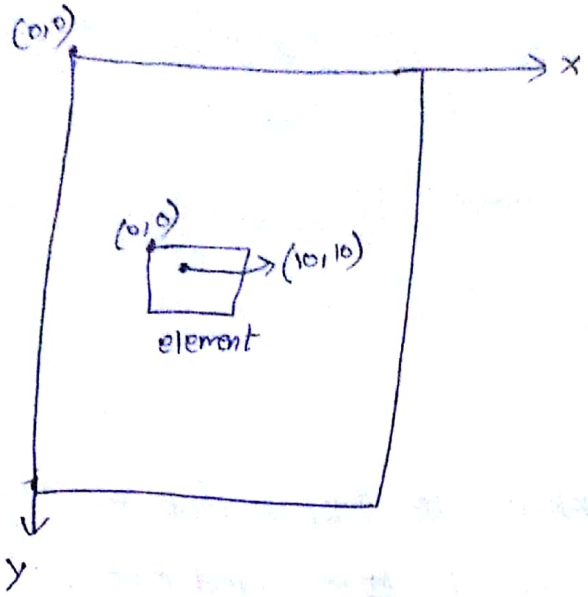
(or)

2)
```
WebElement e=driver.findElement(By.xPath("xxx"));
int x= e.getLocation().getX();
int y= e.getLocation().getY();
TouchAction ta= new TouchAction(driver);
ta.press(x,y).perform();
```

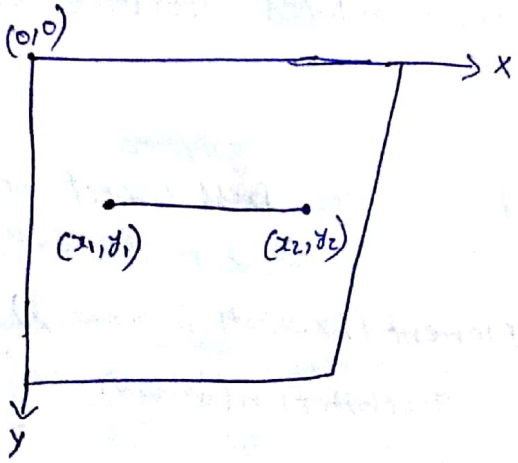
(or)

3)
```
WebElement e= driver.findElement(By.xPath("xxx"));
TouchAction ta= new TouchAction(driver);
ta.press(e,10,10).perform();
```

syn:-3 diagram



moveTo() :- we can use this method to get swipe effect on mobile app screen.



syntax

ta.press(x_1, y_1).moveTo(x_2-x_1, y_2-y_1).release().perform();

(x_1, y_1) → starting point

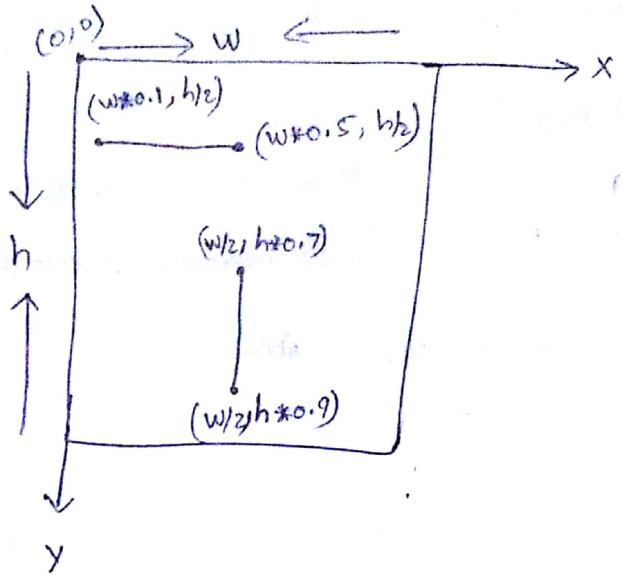
(x_2-x_1, y_2-y_1) → offset

release() :- we can use this method to end existing operation.

cancel() :- we can use this method to cancel existing operation.

Ex:-

Appm Ex:-1



Scenario → Open clock app in AVD

→ Click on cities

→ search for "Delhi" by swiping from bottom to top.

→ select "Delhi"

→ search for "Adelaide" by swiping from top to bottom.

→ select "Adelaide"

→ Close app

Pgm (vertical swiping)

```
// start appium server
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                          \"appium -a 0.0.0.0 -p 4723\" ");
URL u = new URL("http://0.0.0.0:4723/wd/hub");
// Details of AVD and AUT (App under Testing)
DesiredCapabilities dc = new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
dc.setCapability("deviceName", "emulator-5554");
```

```
dc.setCapability("PlatformName", "android");
dc.setCapability("PlatformVersion", "4.2.2");
dc.setCapability("appPackage", "xxxx");
                                  ↳ com.android.deskclock
dc.setCapability("appActivity", "com.android.deskclock.DeskClock");
//create driver object to launch app in AVD
AndroidDriver driver;
while(2>1)
{
    try
    {
        driver=new AndroidDriver(u,dc);
        break; //terminate from loop
    }
    catch(Exception e)
    {
    }
}
try
{
    driver.manage().timeouts().implicitlyWait(5,TimeUnit.SECONDS);
    driver.findElement(By.xpath("//*[@content-desc='Cities']")).click();
    //webdriver wait perform swipe for required element.
    driver.context("NATIVE APP"); //use for browser in apps
    int w=driver.manage().window().getSize().getWidth();
    int h=driver.manage().window().getSize().getHeight();
    TouchAction ta=new TouchAction(driver);
// swipe from bottom to top for required element
```

```
int x=(int) (w/2);
int y=(int) (h*0.9);   //near to bottom
int temp = (int) (h*0.7);   //near to top
while (2>1) //infinite loop
 {
    try
      {
        driver.findElement(By.xpath ("//*[@text='Delhi']")).click();
        break;  //terminate from loop
      }
    catch (Exception e)
      {
        ta.press(x,y).moveTo(0, temp-y).release().perform();
      }
 }
// swipe from top to bottom for required element
 y=(int) (h*0.7);  //near to top
 temp=(int) (h*0.9);  //near to bottom
 while (2>1)
  {
     try
      {
        driver.findElement (By.xpath ("//*[@text='Adelaide']")).click();
        break;
      }
    catch (Exception e)
      {
        ta.press(x,y).moveTo(0, temp-y).release().perform();
      }
  }
```

```
// Back to app here
    driver.pressKeyCode(AndroidKeyCode.BACK);
    // close app
    driver.closeApp();
}
catch (Exception e)
    {
        System.out.println(e.getMessage());
    }
// stop app and close appium server.
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

Ex:-2 (Horizontal swiping)

→ Launch calender app in AVD

→ ~~Scroll~~ swipe from right to left 5 times

→ swipe from left to right 5 times

→ close app.

Prgm

```
// start appium server.
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                    \"appium -a 0.0.0.0 -p 4723\"");
URL u=new URL("http://0.0.0.0:4723/wd/hub");
```

```
//Details of AVD and AUT (app under testing)
DesiredCapabilities dc=new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME,"");
dc.setCapability("deviceName","emulator-5554");
dc.setCapability("PlatformName","android");
dc.setCapability("PlatformVersion","4.2.2");
dc.setCapability("appPackage","com.android.calendar");
dc.setCapability("appActivity","com.android.calendar.AllInOne
                                Activity");
//Create driver object to launch app in AVD
AndroidDriver driver;
while(2>1)
  {
    try
     {
       driver=new AndroidDriver(U,dc);
       break; //terminate from loop
     }
   catch(Exception exc)
     {
     }
  }
try
  { //swipe on app screen
    driver.context("NATIVE_APP");
    int w=driver.manage().window().getSize().getWidth();
    int h=driver.manage().window().getSize().getHeight();
```

```
TouchAction ta=new TouchAction(driver);
// swipe from right to left
    int x=(int) (w*0.9); // right
    int y=(int) (h/2);
    int temp=(int) (w*0.7);
    for(int i=1; i<=5; i++)
     {
       ta.press(x,y).moveTo(temp-x,0).release().perform();
       Thread.sleep(5000);
     }
  // swipe from left to right
      x=(int) (w*0.7);  //left
      y=(int) (h/2);
     temp=(int) (w*0.9); // right
     for(int i=1; i<=5;i++)
     {
       ta.press(x,y).moveTo(temp-x, 0).release().perform();
       Thread.sleep(5000);
     }
 // close app
   driver.closeApp();
}
catch(Exception e)
    {
      System.out.println(e.getMessage());
    }
```

// stop and close appium server

```
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

longPress():- we can use this method to get longPress effect on an element or location.

syntax

```
MobileElement e = (MobileElement) driver.findElement(By.xpath("//*[@xxx='xxx']"));
TouchAction ta = new TouchAction(driver);
ta.longPress(e).release().perform();
```

(or)

```
TouchAction ta = new TouchAction(driver);
ta.longPress(599, 866).release().perform();
```

x By of location

(or)

```
ta.longPress(e, 10, 10).release().perform();
```

location inside of element.

tap():- we can use this method to get tap or click effect on an element or specific location.

syntax

```
MobileElement e = (MobileElement) driver.findElement(By.xpath("xxx"));
TouchAction ta = new TouchAction(driver);
ta.tap(e).perform();
```

(or)

```
ta.tap(e, 10, 10).perform();
```

specific area of element

(or)

```
ta.tap(599, 866).perform();
```

specific location in app screen

waitAction():- we can use this method to maintain waiting during an operation execution.

Syntax

```
Duration d = Duration.of(10, chronoUnit.SECONDS);
```

Duration: class in JDK; chronoUnit: class in JDK

```
TouchAction ta = new TouchAction(driver);
ta.longPress(500, 600).waitAction(d).release().perform();
```

Ex:-3

→ Launch Phone app in ARD
→ Back to home
→ longPress on an element up to 10 seconds
→ tap at specific location in screen.
→ close Phone app

Prgm

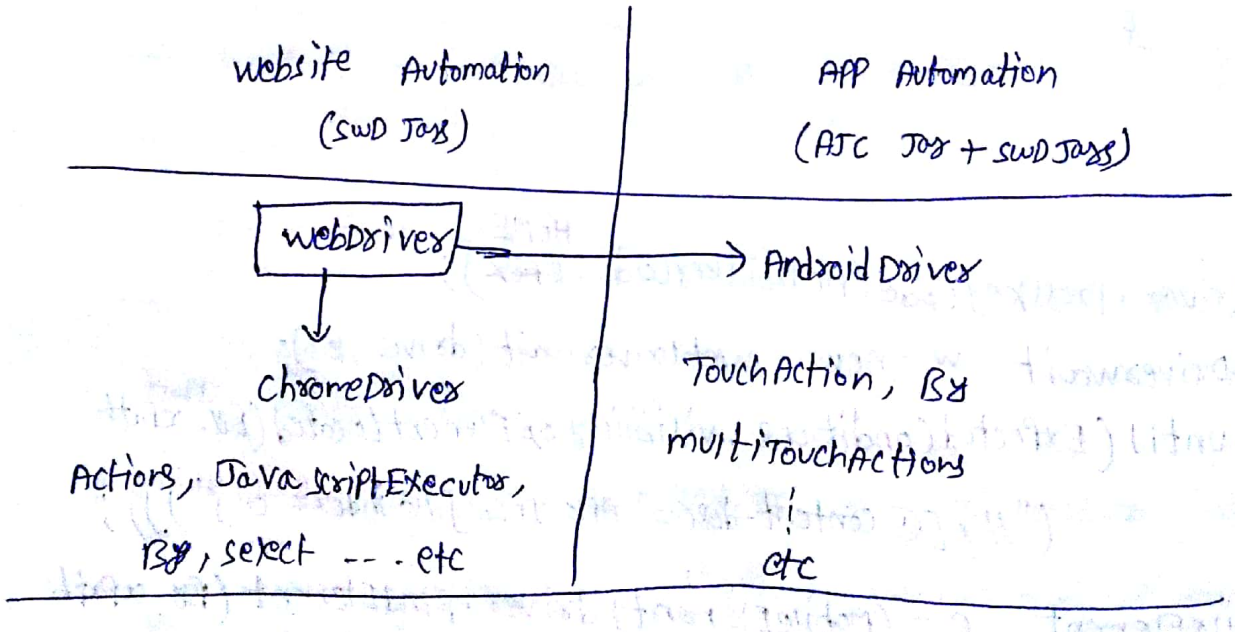
```
// start appium server
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                \"appium -a 0.0.0.0 -p 4723\"");
URL u = new URL("http://0.0.0.0:4723/wd/hub");
// Details of ARD and AUT (app under testing)
DesiredCapabilities dc = new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
```

```
dc.setCapability("deviceName", "xxxx");
dc.setCapability("PlatformName", "android");
dc.setCapability("PlatformVersion", "6.0.1");
dc.setCapability("appPackage", "com.android.contacts");
dc.setCapability("appActivity", "com.android.dialer.DialtactsActivity");
// create driver object to launch app in ARD
AndroidDriver driver;
while (2>1)
    {
      try
       {
          driver = new AndroidDriver(u, dc);
          break;
       }
      catch (Exception e)
       {
       }
    }
try
  {
    driver.pressKeyCode(AndroidKeyCode.HOME);
WebDriverWait w = new WebDriverWait(driver, 100);
w.until(ExpectedConditions.visibilityOfElementLocated(By.xPath
          ("//*[@content-desc='APK Info'][@index='0']")));
MobileElement e = (MobileElement) driver.findElement(By.xPath
          ("//*[@content-desc='APK Info'][@index='0']"));
TouchAction ta=new TouchAction(driver);
Duration d = Duration.of(10, ChronoUnit.SECONDS);
ta.longPress(e).waitAction(d).release().perform();
```

```
        ta.tap(599,872).perform();

    //close app
        driver.launchApp();
        driver.closeApp();
}
catch (Exception e)
    {
        system.out.println(e.getMessage());
    }
// stop and close appium server.
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

Note 1:-

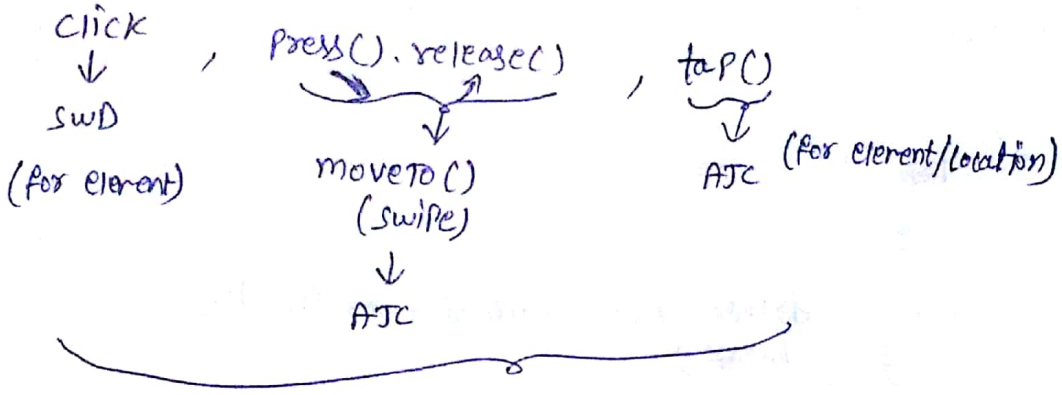


Note 2:- WebDriverwait and Fluentwait classes based wait concepts are also called as Explicit wait (or) conditional wait.

Note 3:- To get specific area coordinates in app screen, we need to use "show pointer location" in developer options in settings of mobile.

Note 4:- In AJC and SWD Java classes, we can get methods to perform similar operation.

Ex:-



g) MultiTouchAction class methods

we can use this class methods to perform parallel touch operations on app screen elements. This class is providing 2 methods such as add() and perform().

Prgm

```
// start appium server
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                    \"appium -a 0.0.0.0 -p 4723\"");
URL u = new URL("http://0.0.0.0:4723/wd/hub");
// Details of ARD and AUT (App under Testing)
DesiredCapabilities dc= new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
dc.setCapability("deviceName", "xxxx");
dc.setCapability("PlatformName", "android");
```

```
dc.setCapability("PlatformVersion", "6.0.1");
dc.setCapability("appPackage", "com.android.contacts");
dc.setCapability("appActivity", "com.android.dialer.DialtactsActivity");
// create driver object to launch app in ARD
AndroidDriver driver;
while(2>1)
    {
      try
        {
          driver=new AndroidDriver(u, dc);
        }  break;
      catch(Exception e)
        {
        }
    }
// Define multiple touch action
WebElement e1=driver.findElement(By.xpath("//*[@text='5']"));
TouchAction ta1=new TouchAction(driver).press(e1).release();
WebElement e2=driver.findElement(By.xpath("//*[@text='9']"));
TouchAction ta2=new TouchAction(driver).press(e2).release();
// create object to MultiTouchAction to perform.
MultiTouchAction ma=new MultiTouchAction(drives);
ma.add(ta1).add(ta2).perform();
Thread.sleep(10000);
// close app
driver.closeApp();
```

// stop and close appium server.

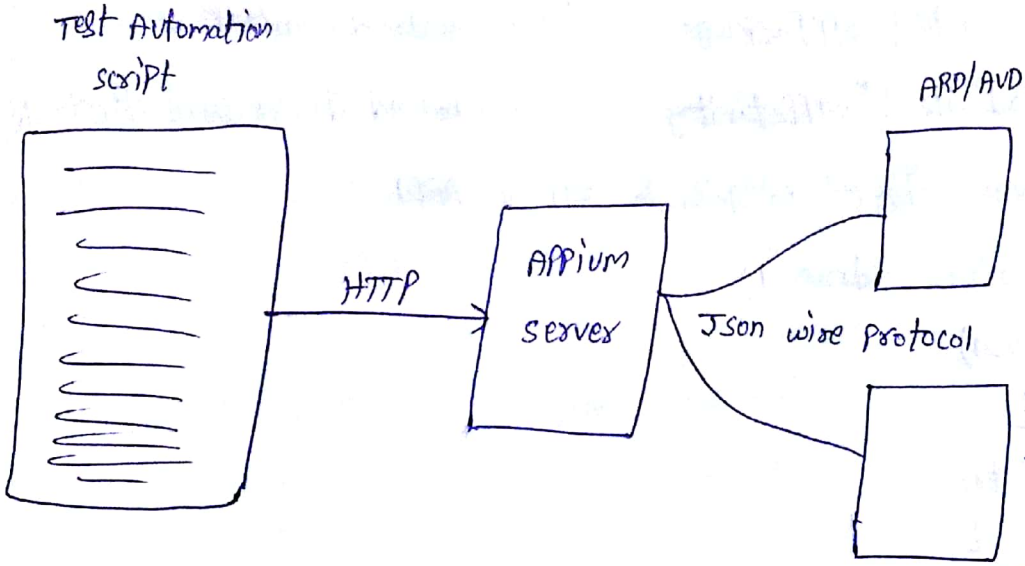
Runtime.getRuntime().exec("taskkill /F /IM node.exe");

Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");

b).

b) cross mobile testing

while automating some apps, we need to take more than one mobile device in execution



json → Javascript object notation

http → Hypertext transfer protocol

- → checking for lock/unlock device
- → sending appium settings to device.
- → verify SSL (secure socket layer) certificates on app.
- → Launch app
- → further code execution on app screen.

Ex:- (Phone call in b/w two mobiles)

Prgm

```
// Get mobile number from keyboard
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter a mobile number");
    String x=sc.nextLine();
// Start appium server.
Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                          \"appium -a 0.0.0.0 -p 4723\" ");
```

```
URL u = new URL("http://0.0.0.0:4723/wd/hub");
//Details of ARD1 and app under testing
DesiredCapabilities dc1 = new DesiredCapabilities();
dc1.setCapability(CapabilityType.BROWSER_NAME, "");
dc1.setCapability("deviceName", "xxxx");
dc1.setCapability("PlatformName", "android");
dc1.setCapability("PlatformVersion", "6.0.1");
dc1.setCapability("appPackage", "com.android.contacts");
dc1.setCapability("appActivity", "com.android.dialer.DialtactsActivity");
//create driver object to launch app in ARD1
AndroidDriver driver1;
while(2>1)
{
try
{
driver1 = new AndroidDriver(u, dc1);
break;
}
catch(Exception e)
{
}
}
//Details of ARD2 and app
DesiredCapabilities dc2 = new DesiredCapabilities();
dc2.setCapability(CapabilityType.BROWSER_NAME, "");
dc2.setCapability("devicename", "xxxx");
```

```
dc2.setCapability("PlatformName", "xxxx");
dc2.setCapability("PlatformVersion", "xxx");
dc2.setCapability("appPackage", "xx");
dc2.setCapability("appActivity", "xxx");

//create driver object to launch app in ARD 2
AndroidDriver driver2;
while(2>1)
    {
     try
      {
          driver2 = new AndroidDriver(u, dc2);
          break;
      }
     catch(Exception e)
      {
      }
  }
// Dial number and activate call in ARD 1
 try
  {
   driver1.manage().timeouts().implicitlyWait(100, TimeUnit.SECONDS);
   for(int i=0; i<x.length; i++)
    {
        char d=x.charAt(i);
     driver1.findElement(By.xPath("//*[@Class='android.widget.TextView']
                    [@text='"+d+"']@index='0']")).click();
    }
```

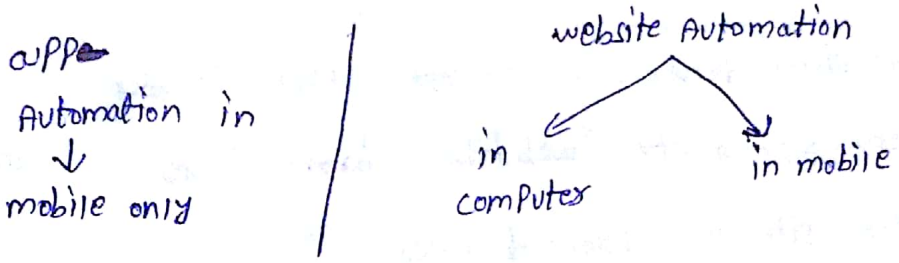
```
driver1.findElement(By.xpath("//*[@content-desc='Call'][@index='0']")).click();
webDriverwait w1=new webDriverwait(driver1,100);
w1.until(ExpectedConditions.visibilityof(driver1.findElementByAndroidUIAutomator(
        "new UiSelector().description(\"End call\")")));
//Lift call in ARD2
TouchAction ta=new TouchAction(driver2);
ta.press(375,1150).moveTo(0,-300).release().perform();
driver2.manage().timeouts().implicitlywait(100,TimeUnit.SECONDS);
driver2.findElement(By.xpath("//*[@content-desc='End'][@index='1']")).click();
//Get and display call cost value in ARD1
webDriverwait w2=new webDriverwait(driver1,100);
w2.until(ExpectedConditions.visibilityofElementLocated(By.xpath(
        "//*[@resource-id='android:id/message']")));
string m=driver1.findElement(By.xpath("//*[@resource-id='android:id/
        message']")).getAttribute("text");
System.out.println(m);
driver1.findElement(By.xpath("//*[@text='Ok']")).click();
//close apps in ARD1 and ARD2
    driver1.closeAPP();
    driver2.closeAPP();
}
catch(Exception e)
    {
        system.out.println(e.getmessage());
    }
```

// stop and close appium server

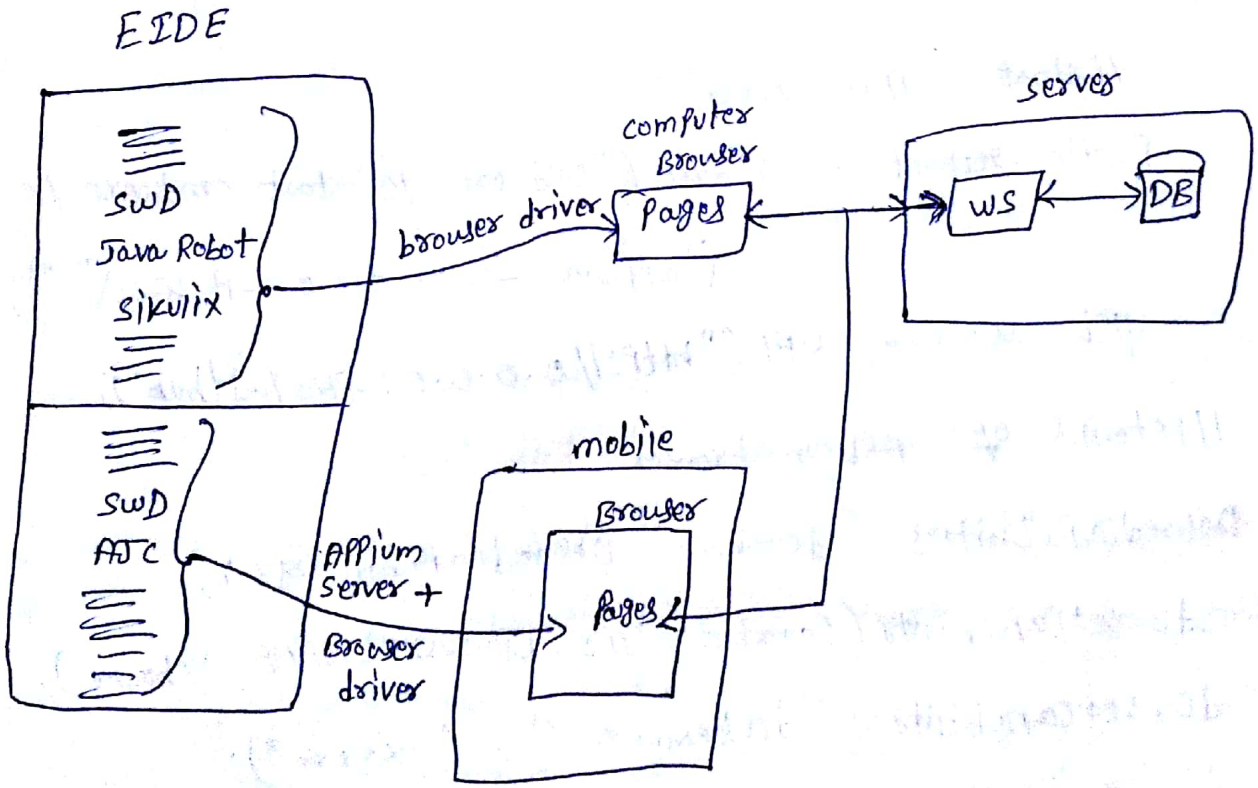
```
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
```

i) Website Automation in Android Mobile

In general people are using websites in computers and mobiles. In this situation, we need to (SDET) automate corresponding websites in computer and mobile



while automating website pages in computers and mobiles, we can get below like execution flow;



Ex-1 Agm

```
//Get Platform
Scanner sc=new Scanner(System.in);
System.out.Println("Enter Platform as computer or mobile");
string P=sc.nextLine();
webDriver driver;
if(P.equals("computer"))
 {
    //Run chromedriver.exe to get chrome browser.
    System.setProperty("webdriver.chrore.driver", "path of Chromedriver");
    //Open site in chrore browser.
      driver =new ChroreDriver();
 }
else
 {
    //start appium server
   Runtire.getRuntire().exec("cmd.exe /c start cmd.exe /k
                              \"appium -a 0.0.0.0 -P 4723\"");
   URL u=new URL("http://0.0.0.0:4723/wd/hub");
  //Details of ARD in browser.
  DesiredCapabilities dc=new DesiredCapabilities();
   dc.setCapability(CapabilityType.BROWSER_NAME, "chroore");
  dc.setCapability("deviceNare", "xxxxx");
  dc.set capability("PlatformNare", "android");
  dc.set capability("PlatformVersion", "6.0.2");
```

```
// create driver object and launch chrome browser
while(x>1)
{
    try
    {
        driver = new AndroidDriver(u, dc);
        break;
    }
    catch(Exception e)
    {
    }
}
}
// Launch site.
driver.get("http://www.google.co.in");
Thread.sleep(3000);
// close site
driver.close();
if(p.equals("mobile"))
{
    // stop and close appium server.
    Runtime.getRuntime().exec("taskkill /F /IM node.exe");
    Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
}
```

Note 1:- while running above like scripts in computer, we can get compatability issues in b/w browser driver and browser. To solve this we need to download and use latest browser driver.

Note 2:- while running above like scripts in mobile, we can get compatability issues in b/w browser and browser driver inbuilt with appium server. To solve these issues we need to download latest browser driver and attach that driver to appium server. Path like shown below:

Path

~~c:\users\kesanam\node_modules\appium-chromedriver\chromedriver\win~~

c:\users\kesanam\node_modules\appium-chromedriver\chromedriver\win

Ex:-2 (Non-Html Elements in webpages)

Prgm

```
// Get Platform
Scanner sc = new Scanner(System.in);
System.out.println("Enter Platform as computer/mobile");
String P = sc.nextLine();
if(P.equals("computer"))
{
// selenium webdriver with sikulix
System.setProperty("webdriver.chrome.driver", "Path of browser");
WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("http://www.youtube.com");
Thread.sleep(20000);
```

```
// search video link (SWD)

driver.findElement(By.name("search_query")).sendKeys("kalam sir speech",
                                                     Keys.ENTER);
Thread.sleep(5000);

driver.findElement(By.partialLinkText("Dr. A.P.J Abdul Kalam")).click();

Thread.sleep(10000); //time to start & to get skipadd

//sikuli code automate video icons

Screen s=new Screen();

if(s.exists("skipadd.png")!=null)
  {
      s.click("skipadd.png");
  }
Thread.sleep(5000);

//Move mouse pointer to video body to get icons

  Location l= new Location(300,200);
  s.wheel(1,Button.LEFT,0);

  // click Pause Icon
   s.click("Pause.png");
   Thread.sleep(5000);

// move mouse pointer to video body to get icons
  s.wheel(1,Button.LEFT,0);

  // click Play Icon
   s.click("Play.png");
   Thread.sleep(5000);
   // close site
```

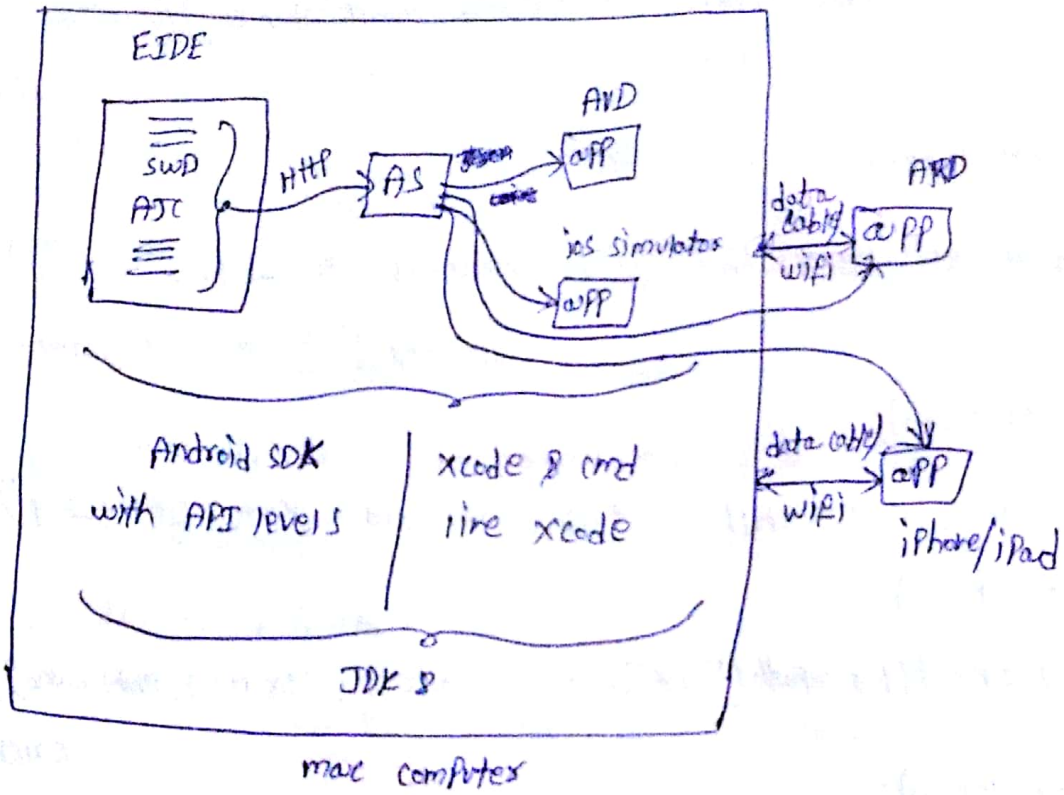
```
    driver.close();
}
else
  {
    // start appium server
    Runtime.getRuntime().exec("cmd.exe /c start cmd.exe /k
                    \"appium -a 0.0.0.0 -p 4723\"");
    URL u=new URL("http://0.0.0.0:4723/wd/hub");
    // Details of ARD with Browser
    DesiredCapabilities dc=new DesiredCapabilities();
    dc.setCapability("CapabilityType.BROWSER_NAME, "Chrome");
    dc.setCapability("deviceName", "xxxx");
    dc.setCapability("PlatformName", "android");
    dc.setCapability("PlatformVersion", "6.0.1");
// create driver object and launch chrome browser.
    AndroidDriver driver;
    while(2>1)
      {
        try
         {
            driver = new AndroidDriver(u, dc);
            break;
         }
        catch(Exception e)
        {
        }
      }
}
```

[In AVD's Browser name is "browser"]

```
driver.get("http://www.youtube.com");
Thread.sleep(10000);
driver.context("NATIVE_APP");
Thread.sleep(20000);
driver.findElement(By.xpath("//*[@text='search youtube'][@index='0']")).
                                                        click();
Thread.sleep(10000);
driver.findElement(By.xpath("//*[@resource-id='koya_elem_1_4']"))
                                        .sendkeys("kalam sir speech");
Thread.sleep(10000);
driver.findElement(By.xpath("//*[@resource-id='koya_elem_1_2']")).click();
Thread.sleep(5000);
driver.findElement(By.xpath("//*[contains(@text, 'Dr.APJ Abdul Kalam')]")).
                                                        click();
Thread.sleep(10000);
driver.findElementByAndroidUIAutomator("new UiSelector().index(0).
                        text(\"video\")").click();
driver.findElement(By.xpath("//*[@text='Pause']")).click();
Thread.sleep(10000);
driver.findElement(By.xpath("//*[@text='Play']")).click();
Thread.sleep(15000);
driver.closeApp();
// stop appium server
Runtime.getRuntime().exec("taskkill /F /IM node.exe");
Runtime.getRuntime().exec("taskkill /F /IM cmd.exe");
}
```

II. Automate Android & ios based apps & sites using "mac" computer:

SWD Jars + AJC Jars
+ selenium standalone server Jars
+ ----- etc



while using mac computer, we are able to automate android and ios based apps and sites. During this process:

a) Installation in mac system:

for Android	for ios
→ JDK 8	→ JDK 8
→ Android SDK (bundle 145)	→ xcode
→ API levels from 17 (4.2.2) for AVD's	→ command line xCode for IOS simulators
→ Eclipse IDE with Java Project, Package & classes with main().	→ Eclipse IDE with Java Project, Package & classes with main().

→ SWD Jars, ATC Jars, selenium standalone Jar, - - - - etc

→ Install nodejs for mac

→ install Appium server using "npm" command.

→ SWD Jars, ATC Jars, selenium standalone Jar - - - etc

→ Install nodejs for mac

→ install Appium server using "npm" command

[npm - node package manager]

b) start appium server

while automating android and ios based apps & sites using mac computer, we need to use below code to start appium server programmitically.

Ex- Runtime.getRuntime().exec("\"appium -a 0.0.0.0 -p 4723\"");

(code to start cmd prompt in mac system)

Runtime → static class in java.awt package in JDK

URL u = new URL("http://0.0.0.0:4723/wd/hub");

URL → instance class in java.net package in JDK

c) DesiredCapabilities

for android:-

```
DesiredCapabilities dc = new DesiredCapabilities();
dc.setCapability(CapabilityType.BROWSER_NAME, "");
dc.setCapability("deviceName", "xxxxx");
```

"" → null

xxxxx → get using adb devices ↵ command

dc.setCapability("PlatformName", "android");

dc.setCapability("PlatformVersion", "x.x");

setting → About
Phone → s/w info
→ observe version number.

dc.setCapability("appPackage", "xxxx.xxxx.xxxx");

dc.setCapability("appActivity", "xxxx.xxxx.Xxxxx");

using aapt cmd / APK info app / adb shell dumpsys cmd
etc

AndroidDriver driver = new AndroidDriver(u, dc);

for ios :-

DesiredCapabilities dc = new DesiredCapabilities();

dc.setCapability(CapabilityType.BROWSER_NAME, "");
Nil

dc.setCapability("deviceName", "xxxx");
simulator/Real iphone

dc.setCapability("PlatformName", "ios");
visible name

dc.setCapability("PlatformVersion", "ios xxxx");

dc.setCapability("app", "xxxx");
visible name of app

IosDriver driver = new IosDriver(u, dc);
instance class
in ATC Jar

d) Inspect Element

for android:-

"uiautomatorviewer" cmd → It was installed in mac computer while installing Android SDK

for ios:- "inspector" cmd

It was installed in mac computer during installation of appium server using "npm".

e) Automating Elements

for Android:-

1) driver.findElement(By.xPath("xxxxx")).operation();

To automate static element in app screen/site page

2) → driver.findElementByAccessibilityId("xxxx").operation();

→ driver.findElementByAndroidUIAutomator("xxxx").operation();

To automate dynamic element (or) static element in dynamic screen/page.

for ios:-

1) driver.findElement(By.xPath("xxxx")).operation();

To automate static element in app screen/site page

2) driver.findElementByAccessibilityId("xxxx").operation(); → using Inspector

To automate dynamic element (or) static element in dynamic screen/page

Note:- while automating websites in mobiles using mac computer, Browser is :

browser → AVD

Chrome → ARD

safari → Ios simulator

safari → IPhone/IPad