סיכום שיעור אחד עשרה:

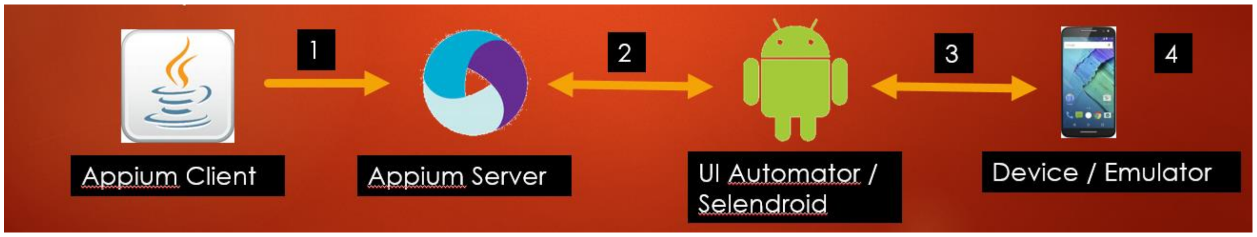
1. Appium

Appium is an open source test automation framework for use with native, hybrid and mobile web apps .It drives iOS, Android, and Windows apps using the WebDriver protocol.

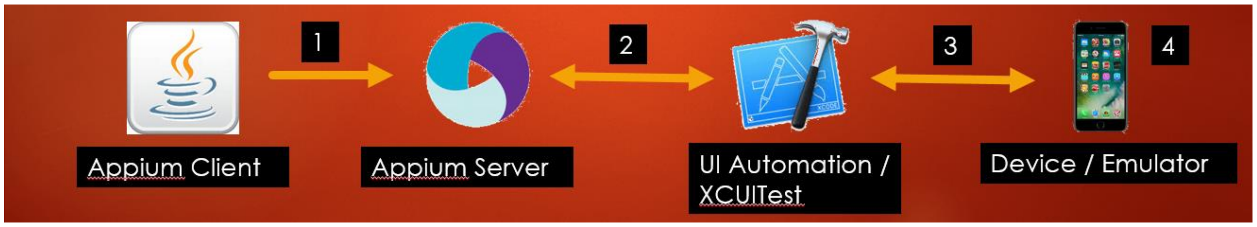
הדרך שבה Appium זה ב JSON Wire Protocol אשר נועד לשליחת/קבלת נתונים מ/אל השרת.
השלבים הם:

- Appium Server then creates a ***automation session*** for the client and also checks the desired capabilities of client and connects with respective vendor-provided frameworks like Selendroid/UIAutomator/ UIAutomation / XCUITest (depends on OS).
- Vendor-provided frameworks will then communicate with ***bootstrap.jar*** which is running in Emulator/Real device for performing client operations
- Bootstrap.jar act as a ***TCP*** server to perform action on our AUT (Application Under Test)

Android:



iOS:



לינק ל documentation:
http://appium.io/slate/en/master/?java#about-appium

2. **<u>סוגי אוטומציה:</u>**

דיברנו על שני סוגי האוטומציה – פנימי /חיצוני והשוונו בין Espresso ל Appium.

Parameters / method	**Espresso**	**Appium**
Speed	Fast (~2 seconds a test)	Slow (~1 min a test)
Setup	Easy setup	Difficult setup
Updates & support	Great (Google)	Slow (open-source)

אז למה בכל זאת בחרנו ב Appium?

- א צריך לדעת פיתוח מובייל.
- אפשרי בהרבה שפות.
- חינמי.
- ועוד..

3. **<u>מציאת אלמנטים</u>**

דיברנו על מספר אפשרויות בהן ניתן למצוא אלמנטים במובייל:

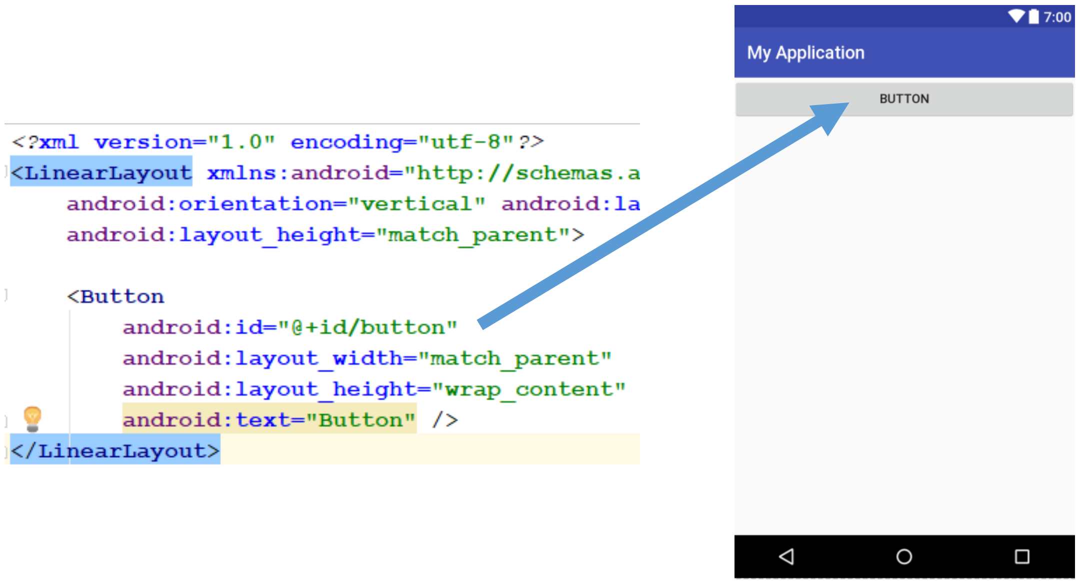
שימוש ב driver.getPageSource() על מנת לקבל את תכולת המסך:

```
System.out.println(driver.getPageSource());
```

וזו התוצאה:

```
<android.widget.EditText index="1" text="" class="android.widget.EditText"
checkable="false" checked="false" clickable="true" enabled="true" focusable="true"
focused="false" scrollable="false" long-clickable="true" password="true" selected="false"
bounds="[48,484][1032,620]" resource-id="com.app:id/userPassword" instance="1" />
```

ראינו איך נראים ה locators מצד מפתחי האפליקציות ובאיזה קלות ניתן להוסיף:



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.a
    android:orientation="vertical" android:la
    android:layout_height="match_parent">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
</LinearLayout>
```

4. **<u>UIautomatorviewer:</u>**

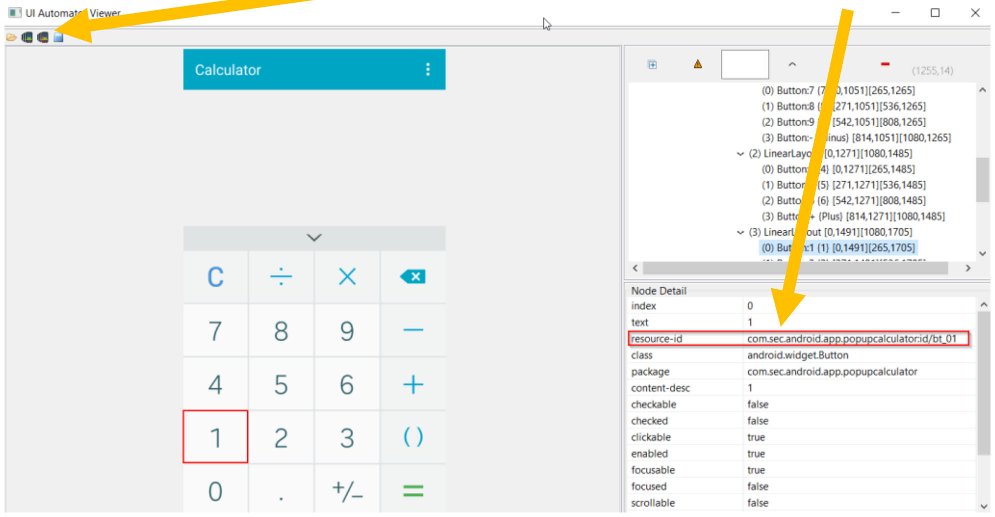
UIautomatorviewer משמש אותנו למציאת אלמנטים באנדרואיד:

- UIautomatorviewer is a GUI tool to scan and analyze the UI components of an android application.
- With UIautomatorviewer, you can inspect the UI of an android application in order to find out the hierarchy and view different properties (id, text…) of the element.
- Uiautomatorviewer comes with the Android SDK

המיקום שלו הוא ב = C:\Users\<user>\AppData\Local\Android\sdk\tools\bin

לתוכנה קוראים **UIautomatorviewer.bat**.

בלחיצה על האייקון של הטלפון נקבל snapshot של אותו המסך בו נוכל למצוא אלמנטים ו locators בקלות.



```
(0) Button:7 {7  0,1051][265,1265]
(1) Button:8 {  [271,1051][536,1265]
(2) Button:9    [542,1051][808,1265]
(3) Button:-  inus} [814,1051][1080,1265]
(2) LinearLayo   [0,1271][1080,1485]
    (0) Button  4} [0,1271][265,1485]
    (1) Button  {5} [271,1271][536,1485]
    (2) Butto  {6} [542,1271][808,1485]
    (3) Butt  + {Plus} [814,1271][1080,1485]
(3) LinearL  out [0,1491][1080,1705]
    (0) But  n:1 {1} [0,1491][265,1705]
```

Node Detail

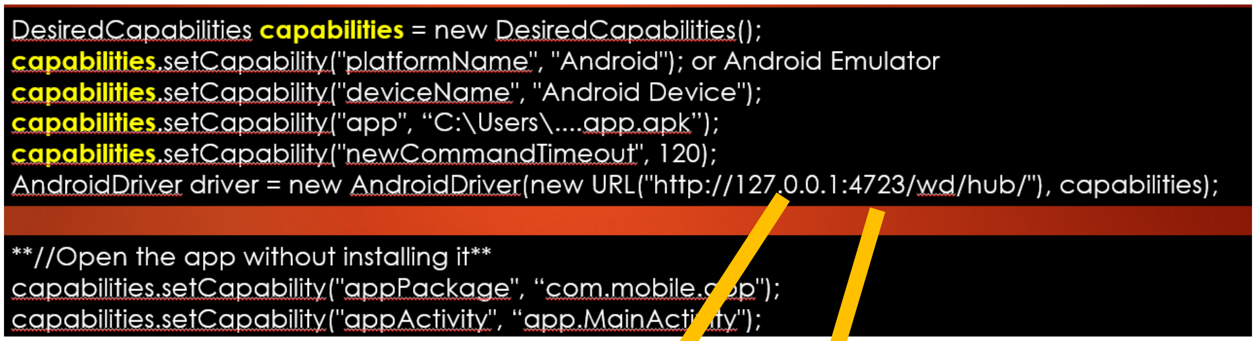
index	0
text	1
resource-id	com.sec.android.app.popupcalculator:id/bt_01
class	android.widget.Button
package	com.sec.android.app.popupcalculator
content-desc	1
checkable	false
checked	false
clickable	true
enabled	true
focusable	true
focused	false
scrollable	false

בלחיצה על כל אלמנט במסך נקבל את כל המאפיינים שלו (id, text, class…).
העבודה מול האלמנטים מתבצעת בדרך זהה למה שעשינו בסלניום רק שהפעם נעבוד מול AndroidDriver ולא ChromeDriver:

```
driver.findElement(By.id("com.sec.android.app.popupcalculator:id/bt_01")).click();
```

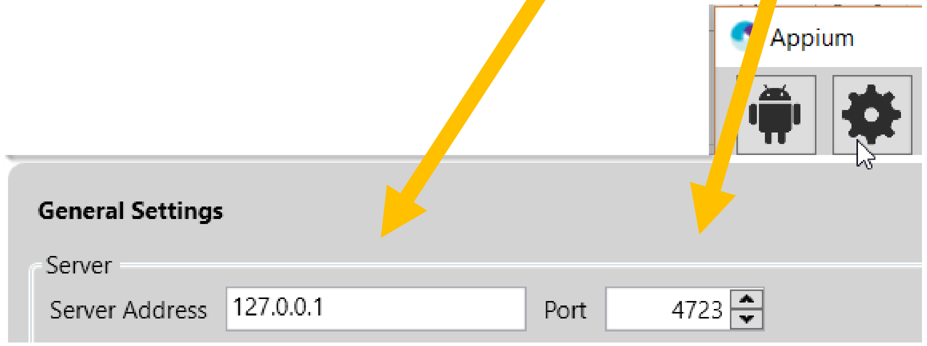
:DesiredCapabilities .5

Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session we're interested in starting up. There are also various capabilities which can modify the behavior of the server during automation. For example, we might set the platformName capability to iOS to tell Appium that we want an iOS session, rather than an Android or Windows one. Or we might set the safariAllowPopups capability to true in order to ensure that, during a Safari automation session, we're allowed to use JavaScript to open up new windows. See the capabilities doc for the complete list of capabilities available for Appium.



```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("platformName", "Android"); or Android Emulator
capabilities.setCapability("deviceName", "Android Device");
capabilities.setCapability("app", "C:\Users\....app.apk");
capabilities.setCapability("newCommandTimeout", 120);
AndroidDriver driver = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub/"), capabilities);

**//Open the app without installing it**
capabilities.setCapability("appPackage", "com.mobile.app");
capabilities.setCapability("appActivity", "app.MainActivity");
```

** We need to make sure that the same address and port are configured in the Appium server:



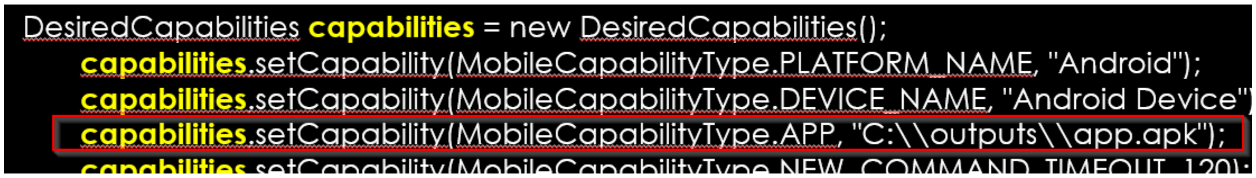
General Settings

Server

Server Address: 127.0.0.1   Port: 4723

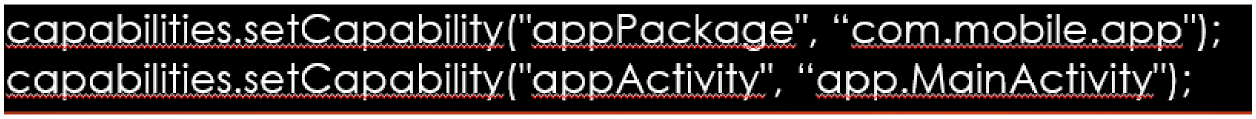
6. הרצה נקייה (התקנת קובץ האפליקציה) מול הרצה רגילה:

- כשנרצה להריץ את הטסט לאחר התקנה נקייה של האפליקציה, נוכל להשתמש ב capability בשם app ולציין את הנתיב של האפליקציה במחשב:



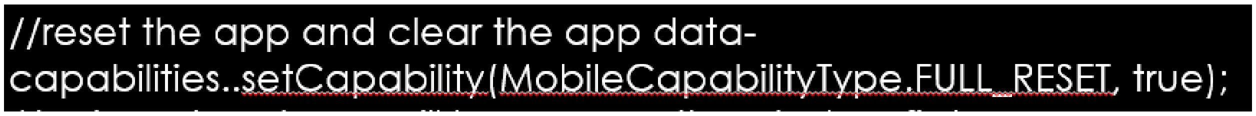
```
DesiredCapabilities capabilities = new DesiredCapabilities();
    capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, "Android");
    capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "Android Device")
    capabilities.setCapability(MobileCapabilityType.APP, "C:\\outputs\\app.apk");
    capabilities.setCapability(MobileCapabilityType.NEW_COMMAND_TIMEOUT, 120);
```

- במידה ונרצה להריץ את הטסט על אפליקציה שכבר מותקנת, נשתמש ב appPackage כמזהה האפליקציה וב appActivity ע"מ שהטסט ידע מאיפה לרוץ:



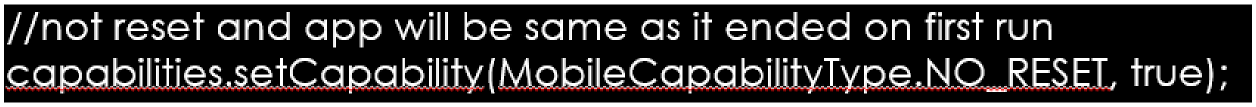
```
capabilities.setCapability("appPackage", "com.mobile.app");
capabilities.setCapability("appActivity", "app.MainActivity");
```

במידה ונרצה לנקות את כל המידע של האפליקציה, נוסיף את השורה הבאה:



```
//reset the app and clear the app data-
capabilities..setCapability(MobileCapabilityType.FULL_RESET, true);
```

במידה ונרצה שהאפליקציה תתחיל מאותה הנקוד בה היא נמצאת – זאת אומרת לא ממסך ההתחלה, נוכל להשתמש ב:



```
//not reset and app will be same as it ended on first run
capabilities.setCapability(MobileCapabilityType.NO_RESET, true);
```

7. שימוש ב Android UI Selector וב MobileBy למציאת אלמנטים

A. <u>Android UI Selector &</u>

- Specifies the elements in the layout hierarchy for tests to target, filtered by properties such as text value, content-description, class name, and state information .
- You can also target an element by its location in a layout hierarchy.
- We saw many locators we can use using Android UI Selector:

```
 WebElement element = driver.findElementByAndroidUIAutomator("new UiSelector()
.index(0).clickable(true)");
```

  We can find the full list at:
  https://developer.android.com/reference/android/support/test/uiautomator/UiSelector.html

- Apple has dropped UIAutomation support in iOS 9.3 and above. Fortunately, Apple introduced a new automation technology, called XCUITest, beginning with iOS 9.3. For iOS 10 and up.

```
 MobileElement e = (MobileElement)
driver.findElement(MobileBy.ByIosUIAutomation.IosUIAutomation(""));
```

B. <u>MobileBy</u>

MobileBy is optimized for MobileUIs mainly for Android and IOS instead of the generic By for mainly web locators, but will do the same.

```
 WebElement checkBox =
driver.findElement(MobileBy.id("com.app:id/long_click_button"));
```

Using UI automator for scrolling:

By id:

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector()."
+"scrollable(true)).scrollIntoView(new UiSelector().resourceId("resourceId"))"));

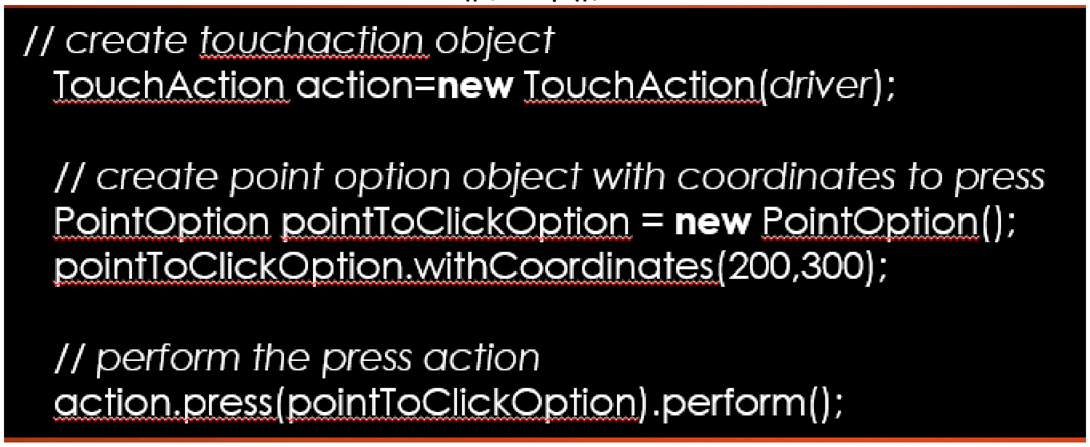
By element text:

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true))"
+ ".scrollIntoView(new UiSelector().text("textValue"))"));

Touch action:

- As we learned Appium libraries are an “extension” for Selenium libraries.
- Therefore we are able to use the same methods we learned in web automation (e.g. click(), sendKeys(), clear() etc.).
- We will concentrate on mobile devices actions we can perform.
- The class we are going to use is **TouchAction**

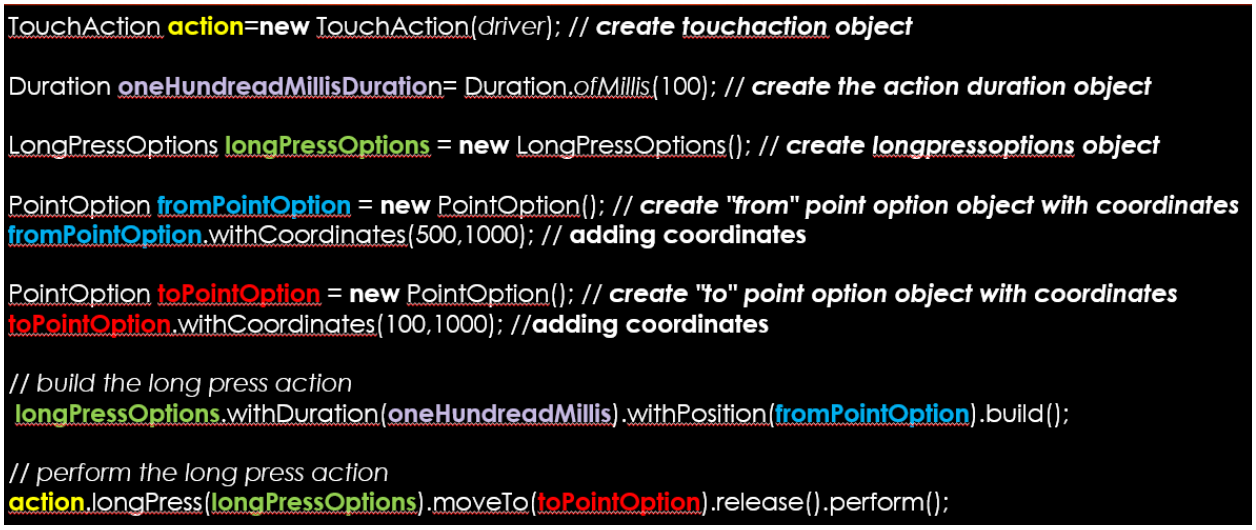
Press():
Press will do the same as Click() / Tap();



```
// create touchaction object
  TouchAction action=new TouchAction(driver);

  // create point option object with coordinates to press
  PointOption pointToClickOption = new PointOption();
  pointToClickOption.withCoordinates(200,300);

  // perform the press action
  action.press(pointToClickOption).perform();
```

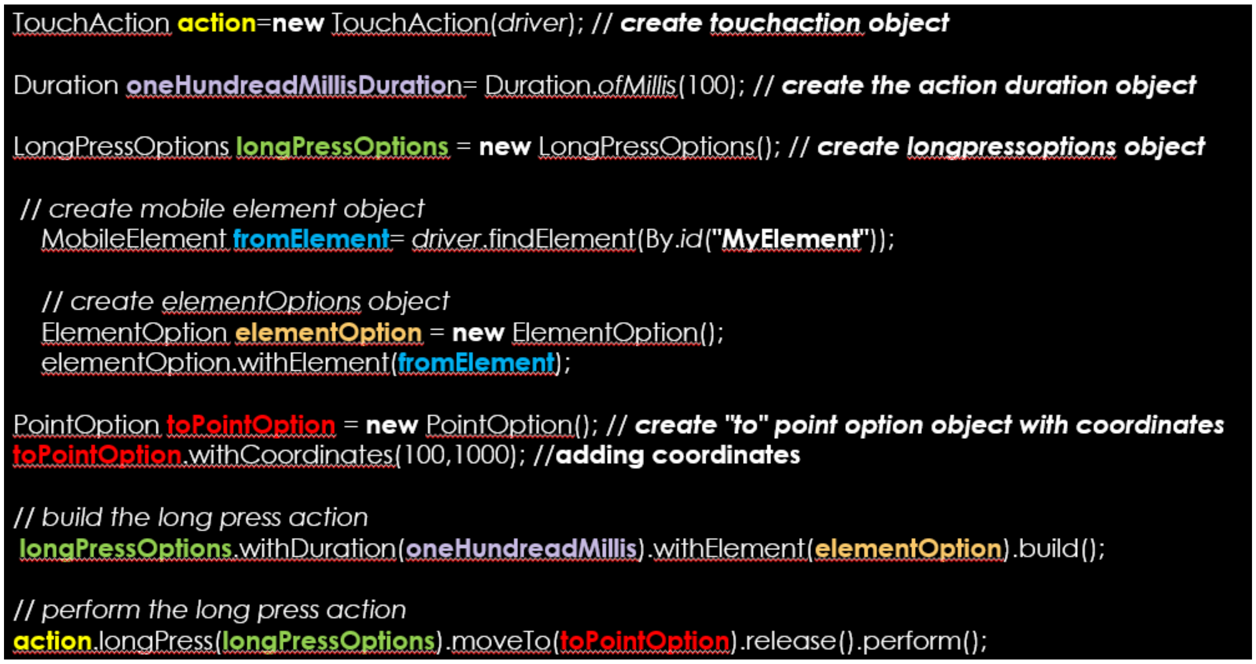
Swipe

Example of swiping using coordinates:



```
TouchAction action=new TouchAction(driver); // create touchaction object

Duration oneHundreadMillisDuration= Duration.ofMillis(100); // create the action duration object

LongPressOptions longPressOptions = new LongPressOptions(); // create longpressoptions object

PointOption fromPointOption = new PointOption(); // create "from" point option object with coordinates
fromPointOption.withCoordinates(500,1000); // adding coordinates

PointOption toPointOption = new PointOption(); // create "to" point option object with coordinates
toPointOption.withCoordinates(100,1000); //adding coordinates

// build the long press action
 longPressOptions.withDuration(oneHundreadMillis).withPosition(fromPointOption).build();

// perform the long press action
action.longPress(longPressOptions).moveTo(toPointOption).release().perform();
```

Example of swiping using An element:



```
TouchAction action=new TouchAction(driver); // create touchaction object

Duration oneHundreadMillisDuration= Duration.ofMillis(100); // create the action duration object

LongPressOptions longPressOptions = new LongPressOptions(); // create longpressoptions object

 // create mobile element object
   MobileElement fromElement= driver.findElement(By.id("MyElement"));

   // create elementOptions object
   ElementOption elementOption = new ElementOption();
   elementOption.withElement(fromElement);

PointOption toPointOption = new PointOption(); // create "to" point option object with coordinates
toPointOption.withCoordinates(100,1000); //adding coordinates

// build the long press action
 longPressOptions.withDuration(oneHundreadMillis).withElement(elementOption).build();

// perform the long press action
action.longPress(longPressOptions).moveTo(toPointOption).release().perform();
```

תזכורת ממה שעשינו בכיתה:

1. Set DesiredCapabilities with parameters.
2. Setting server URL for AndroidDriver (same as the one we set in Appium settings)
3. Finding elements.
4. Clicking elements.
5. Closing session.

```
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;

import java.io.IOException;
import java.net.URL;
import java.util.concurrent.TimeUnit;

public class AppsExamples {
    private static AndroidDriver<MobileElement> driver;

    @BeforeClass
    public static void setUp() throws IOException {

        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("platformName", "Android");
        capabilities.setCapability("deviceName", "Android Device");
  1     capabilities.setCapability("appPackage", "com.android.chrome");
        capabilities.setCapability("appActivity", "com.google.an-
droid.apps.chrome.Main");
        capabilities.setCapability("newCommandTimeout", 120);
        capabilities.setCapability("unicodekeyboard", true);
        capabilities.setCapability("resetkeyboard", true);
        driver = new AndroidDriver(new URL("http://0.0.0.0:4723/wd/hub/"), capabili-   2
ties);
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

    }

    @Test
    public void test1_skipIntro() throws InterruptedException, IOException {
        WebElement introSkipButton = driver.findEle-
ment(By.id("il.co.mintapp.buyme:id/skipButton"));   3
        introSkipButton.click();
    }                                          4

    @AfterClass
    public static void tearDown() throws IOException, InterruptedException {

        driver.quit();   5
    }
}
```