A white ornate corner with black background

Description automatically generatedA white ornate corner with a black background

Description automatically generated

**BỘ CÔNG THƯƠNG**

**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white sign with a red and white logo

Description automatically generated**……** 🙦 🕮🙤 **……**

ĐỒ ÁN MÔN HỌC KIỂM THỬ PHẦN MỀM

***Đề tài***

Kiểm Thử Phần Mềm Calculator (Android) bằng Appium

## Giáo viên hướng dẫn: NGUYỄN THỊ NGỌC

## A white corner with a black background Description automatically generatedA white ornate corner with a black background Description automatically generatedSinh viên thực hiện:

|  |  |
| --- | --- |
| **1. VƯƠNG TÂM** | **0306221067** |
| **2. TRẦN QUỐC BẢO** | **0306221003** |
| **3. HOÀNG MINH HIẾU**  **4. MAI QUỐC TRUNG**  **5. TRẦN HOÀNG TIẾN**  **LỚP: CĐ TH 22DĐ D**  **KHÓA: 2022 – 2025** | **0306221020**  **0306221087**  **0306221080** |

**TP. HCM, tháng 10 năm 2024**

# MỤC LỤC

[CHƯƠNG 1. KIỂM THỬ TRÊN THIẾT BỊ DI ĐỘNG 1](#_Toc180646300)

[1. Kiểm thử trên thiết bị di động 1](#_Toc180646301)

[1.1. Các khái niệm cơ bản về ứng dụng di động 1](#_Toc180646302)

[1.2. Các đặc điểm của kiểm thử di động 3](#_Toc180646303)

[2. Kiểm thử giao diện, chức năng, tự động và đơn vị 4](#_Toc180646304)

[2.1. Kiểm thử giao diện (UI Testing) 5](#_Toc180646305)

[2.2. Kiểm thử chức năng (Functional Testing) 5](#_Toc180646306)

[2.3. Kiểm thử đơn vị (Unit Testing) 6](#_Toc180646307)

[2.4. Kiểm thử tự động 7](#_Toc180646308)

[2.5. Mục tiêu của kiểm thử tự động 7](#_Toc180646309)

[2.6. Nguyên tắc kiểm thử tự động 8](#_Toc180646310)

[2.7. Quy trình kiểm thử tự động 10](#_Toc180646311)

[2.8. So sánh giữa kiểm thử di động và các môi trường khác 11](#_Toc180646312)

[3. Một số khái niệm khác 12](#_Toc180646313)

[4. Một số công cụ kiểm thử tự động 13](#_Toc180646314)

[CHƯƠNG 2. THỰC NGHIỆM SỬ DỤNG APPIUM CHO KIỂM THỬ TỰ ĐỘNG TRÊN ANDROID 15](#_Toc180646315)

[1. Giới thiệu phần mềm Appium 15](#_Toc180646316)

[1.1. Công cụ Appium 15](#_Toc180646317)

[1.2. Nguyên lý hoạt động của Appium 15](#_Toc180646318)

[1.3. Phần mềm Appium 15](#_Toc180646319)

[2. Lợi ích kiểm thử tự động hoá thiết bị bằng Appium 16](#_Toc180646320)

[2.1. Tăng độ bao phủ của bài kiểm thử: 16](#_Toc180646321)

[2.2. Nâng cao hiệu quả kiểm thử: 16](#_Toc180646322)

[2.3. Đảm bảo chất lượng sản phẩm: 16](#_Toc180646323)

[2.4. Linh hoạt và mở rộng: 16](#_Toc180646324)

[2.5. Tích hợp với các công cụ khác: 17](#_Toc180646325)

[3. Thực hiện Appium bằng IntelliJ IDEA 17](#_Toc180646326)

[3.1. Cài đặt Android Studio 18](#_Toc180646327)

[3.2. Cài đặt JDK 23 18](#_Toc180646328)

[3.3. Cài đặt Nodejs 19](#_Toc180646329)

[3.4. Cài đặt Appium 20](#_Toc180646330)

[3.5. Cài đặt Uiautomator2 20](#_Toc180646331)

[3.6. Cài đặt Appium Inspector 21](#_Toc180646332)

[3.7. Cài đặt Intellij IDEA 22](#_Toc180646333)

[3.8. Tải Apache maven 23](#_Toc180646334)

[3.9. Xây dựng bộ ca kiểm thử cho ứng dụng cần kiểm thử 25](#_Toc180646335)

[3.10. Tạo dự án kiểm thử 26](#_Toc180646336)

[3.11. Báo Cáo 31](#_Toc180646337)

[TÀI LIỆU THAM KHẢO 33](#_Toc180646338)

# DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU

[Hình 1-1. Quy trình Kiểm thử tự động trong mỗi quan hệ với Kiểm thử phần mêm 16](file:///C:\Users\USER\Desktop\Báo%20Cáo%20Đồ%20Án%20Kiểm%20Thử%20Phần%20Mềm%20-%20Nhóm%20Cánh%20Én.docx#_Toc180477645)

[Hình 2-1.Thêm vào JAVA\_HOME 22](#_Toc180477649)

[Hình 2-2. Xem java version 23](#_Toc180477650)

[Hình 2-3. Kiểm tra phiên bản node 24](#_Toc180477651)

[Hình 2-4. Cài đặt Appium bằng cmd 24](#_Toc180477652)

[Hình 2-5. Cài đặt uiautomator2 25](#_Toc180477653)

[Hình 2-6. Cài đặt appium inspector 26](#_Toc180477654)

[Hình 2-7. Cài đặt intellij IDEA Community Edition 27](#_Toc180477655)

[Hình 2-8. Thêm MAVEN\_HOME 27](#_Toc180477656)

[Hình 2-9. Thêm đường dẫn Path 28](#_Toc180477657)

[Hình 2-10. Xem maven version 29](#_Toc180477658)

[Hình 2-11. Màn hình ứng dụng test 29](#_Toc180477659)

[Hình 2-12. Tạo Project 31](#_Toc180477660)

[Hình 2-13. Thêm appium java 32](#_Toc180477661)

[Hình 2-14. Thêm testng 32](#_Toc180477662)

[Hình 2-15. Tạo class viết code 33](#_Toc180477663)

[Hình 2-16. Tạo DataReader 33](#_Toc180477664)

[Hình 2-17. Thông tin cấu hình 34](#_Toc180477665)

[Hình 2-18. Lấy dữ liệu 34](#_Toc180477666)

[Hình 2-19. Code đọc thông tin từ file 35](#_Toc180477667)

[Hình 2-20. Code đóng ứng dụng 35](#_Toc180477668)

[Hình 2- 21. Kết quả kiểm thử 35](#_Toc180477669)

# KIỂM THỬ TRÊN THIẾT BỊ DI ĐỘNG

Chương này sẽ giới thiệu sơ lược về lĩnh vực kiểm thử trên di động, đặc biệt là về kiểm thử tự động trên di động. Ngoài ra, sự khác nhau giữa kiểm thử trên di động so với trên các nền tảng khác và ưu, nhược điểm của kiểm thử tự động cũng sẽ được đề cập đến.

## Kiểm thử trên thiết bị di động

Kiểm thử phần mềm trên thiết bị di động (Mobile Application Testing) là quá trình kiểm tra chất lượng và tính ổn định của các ứng dụng di động để đảm bảo rằng chúng hoạt động đúng cách trên nhiều thiết bị, hệ điều hành và mạng khác nhau. Đây là một phần quan trọng trong quá trình phát triển ứng dụng, giúp đảm bảo rằng người dùng có trải nghiệm tốt nhất với ứng dụng trên các thiết bị di động như điện thoại thông minh, máy tính bảng.

### Các khái niệm cơ bản về ứng dụng di động

#### Giới thiệu

**Ứng dụng di động** là các chương trình phần mềm được thiết kế để chạy trên các thiết bị di động như điện thoại thông minh, máy tính bảng hoặc đồng hồ thông minh. Các ứng dụng này được phát triển để tận dụng các tính năng phần cứng và phần mềm đặc trưng của thiết bị di động, nhằm cung cấp các chức năng hoặc dịch vụ cho người dùng. Ứng dụng di động có thể được tải xuống và cài đặt từ các kho ứng dụng như **Google Play** (Android), **App Store** (iOS), hoặc từ các nguồn khác tùy thuộc vào hệ điều hành.

#### Phân loại ứng dụng trên thiết bị di động

* *Ứng dụng gốc (Native applications)*

Ứng dụng được phát triển riêng cho một nền tảng di động cụ thể (Android hoặc iOS). Native apps tận dụng tối đa các tính năng của hệ điều hành và phần cứng, giúp tối ưu hóa hiệu suất và giao diện người dùng. Tuy nhiên, phải phát triển hai phiên bản khác nhau cho Android và iOS, đa số các trò chơi trên thiết bị di động đều là ứng dụng gốc.

* *Ứng dụng Web (Web applications)*

Là các ứng dụng chạy trực tiếp trên trình duyệt di động mà không cần cài đặt. Mặc dù không thể tận dụng tối đa các tính năng của thiết bị di động, web apps dễ bảo trì và không yêu cầu phát triển riêng cho từng nền tảng.. Các ứng dụng web phù hợp cho trình duyệt trên điện thoại thông minh hoặc máy tính bảng và người dùng cần một trình duyệt và kết nối Internet để sử dụng chúng.

* *Ứng dụng lai (Hybrid applications)*

Ứng dụng sử dụng công nghệ web (HTML, CSS, JavaScript) nhưng được đóng gói như một ứng dụng di động native. Hybrid apps có thể chạy trên cả Android và iOS, nhưng không có hiệu suất tốt bằng ứng dụng native. Sự khác biệt giữa ứng dụng lai và ứng dụng web là ứng dụng lai có gói ứng dụng gốc và nó thường được cài đặt từ nền tảng AppStore riêng.

#### Các hệ điều hành trên thiết bị di động

* *Hệ điều hành Android*

Android là hệ điều hành phát triển nhanh nhất và phổ biến rộng rãi nhất trong các hệ điều hành di động. Android được sở hữu và quản lý bởi Open Handset Alliance - tập đoàn công nghiệp tạo phần cứng, phần mềm và viễn thông tiêu chuẩn mở cho thiết bị di động. Tập đoàn này được dẫn dắt bởi Google. Kể từ khi Android là dự án mã nguồn mở dựa trên Linux, hầu hết các nhà sản xuất và các hãng di động tận dụng điều đó và sửa đổi hệ điều hành cho phù hợp với phần cứng của họ, tăng sự phức tạp của các hệ thống. Thực tế này làm cho hệ điều hành điện thoại di động Android bị phân mảnh nhất, làm tăng chi phí kiểm tra và phức tạp. Tuy nhiên, nhiều công cụ kiểm thử khác nhau và mục tiêu các framework mục chủ yếu là cho Android do nó là hệ điều hành phổ biến nhất.

* *Hệ điều hành iOS*

IOS là hệ điều hành di động phát triển và sở hữu bởi Apple Inc. Đó là mã nguồn đóng, hệ thống vận hành giống Unix dựa trên Darwin (BSD) và OS X. IOS được phát triển cho iPhone, nhưng bây giờ iOS chạy trên iPad, iPod Touch hoặc Apple TV. Các nhà sản xuất khác không được cấp phép sử dụng iOS. Do đó, chỉ các thiết bị của Apple có thể chạy nó.

Hầu hết người dùng iOS có phiên bản iOS 10 hay mới nhất 11. Kết quả là, thử nghiệm trên tất cả các thiết bị có sẵn không phải là khó khăn như trên Android. Hệ điều hành phân mảnh được trình bày chủ yếu bởi các tính năng

thiết lập hệ điều hành khác nhau (một số tính năng không có sẵn trên một số thiết bị). Mặt khác, các thiết bị của Apple là một trong những thiết bị rất tốn kém.

* *Hệ điều hành Windows & Windows Phone*

Với Windows 8, Microsoft đã chuyển hệ điều hành Windows đến với các thiết bị di động. Windows 8.1, là phiên bản hiện tại, có thể chạy trên máy tính cá nhân và máy tính bảng. Hơn nữa, Microsoft đã có hệ điều hành Windows Phone đặc biệt cho điện thoại thông minh. Hai nền tảng được hội tụ trong Windows 10.

### Các đặc điểm của kiểm thử di động

#### Sự đa dạng các thiết bị di động

* Đa dạng các hãng sản xuất thiết bị như HTC, SamSung, Apple, Nokia, v.v. với các kích thước màn hình và cấu hình phần cứng khác nhau.
* Đa nền tảng: Việc chắc chắn rằng ứng dụng di động chạy được trên tất cả các loại thiết bị (smartphone, tablet hay phablet được cung cấp bởi một số các nhà cung cấp lớn (như Samsung, Sony, Nokia, HTC, Apple…) và trên tất cả các hệ điều hành (iOS, Android, Windows, Blackberry…) thực sự là một thách thức.
* Các thiết bị di động có thời gian chạy ứng dụng khác nhau.

#### Thách thức phần cứng của thiết bị

* Khả năng thích nghi và không gian giới hạn khiến kích thước màn hình thay đổi liên tục: Với mỗi một dòng máy, mỗi loại thiết bị sẽ có nhiều kích thước khác nhau để đáp ứng tính cạnh tranh trên thị trường nên việc phát triển làm sao để ứng dụng có thể thích nghi với kích thước của các màn hình khác nhau cũng là một thách thức.
* Giới hạn tốc độ xử lý: các thiết bị di động đang ngày càng được nâng cấp về tốc độ xử lý để phù hợp với nhu cầu ngày càng cao của người dùng.
* Giới hạn dung lượng bộ nhớ của thiết bị: không giống như trên máy tính, dung lượng bộ nhớ của các thiết bị di động gần như không thể thay thế hay nâng cấp.
* Sự khác biệt về giao thức của thiết bị WAP/HTTP.

#### Thách thức về đường truyền mạng

* Đa dạng các loại mạng (GSM/GPRS/WIFI/3G).
* Không dự đoán được thời gian cho truyền tải dữ liệu.
* Khác biệt về tốc độ kết nối.
* Đa dạng các nhà mạng với những tính năng mạng khác nhau.

#### Các ca kiểm thử đặc biệt cho kiểm thử các ứng dụng di động

* Sự hao tổn pin: Việc theo dõi sự hao tổn pin khi chạy ứng dụng trên thiết bị di động rất quan trọng.
* Tốc độ chạy ứng dụng: Theo dõi thời gian phản hồi trên các thiết bị khác nhau với các dung lượng bộ nhớ khác nhau, tốc độ mạng khác nhau.
* Yêu cầu bộ nhớ: Khi tải, cài đặt và chạy ứng dụng
* Kiểm tra tính tương thích của ứng dụng: Để đảm bảo ứng dụng không bị crash khi mất kết nối mạng hoặc các tác động ngoại vi khác.

**Ngoài ra kiểm thử ứng dụng di động cũng bao gồm các dạng kiểm thử sau:**

* Kiểm thử giao diện (UI Testing): Kiểm tra màu sắc, phong cách Menu, nhất quán của giao diện người dùng trên các thiết bị khác nhau.
* Kiểm thử chức năng (Function Testing): Kiểm tra các chức năng chính của ứng dụng di động theo đặc điểm kĩ thuật của thiết bị.
* Kiểm thử hiệu suất và chịu tải (Performance and Load Test): Kiểm tra hành vi của ứng dụng di động trong các nguồn tài nguyên thấp (Bộ nhớ/ Không gian lưu trữ), hành vi của trang web điện thoại di động khi nhiều người sử dụng điện thoại di động cùng truy cập vào trang app di động.
* Kiểm tra khả năng sử dụng (Usability Testing): Kiểm tra các khía cạnh khả năng sử dụng các ứng dụng di động.
* Thử nghiệm tương thích (Compatibility Testing): Kiểm tra khả năng tương thích của ứng dụng của bạn với các tính năng thiết bị gốc để đảm bảo rằng ứng dụng của bạn không cản trở các ứng dụng khác trong thiết bị.
* Kiểm tra gián đoạn: Vì lí do các thiết bị di động có bộ nhớ thấp hơn nhiều so với desktop nên phải đảm bảo rằng khi có cuộc gọi thoại, tin nhắn SMS, cắm sạc, thông báo bộ nhớ thấp trong khi ứng dụng đang chạy không gây ra bất cứ xung đột nào.

## Kiểm thử giao diện, chức năng, tự động và đơn vị

### Kiểm thử giao diện (UI Testing)

Kiểm thử giao diện (User Interface Testing) trong môi trường di động là quá trình kiểm tra xem các thành phần của giao diện người dùng (UI) của ứng dụng có hoạt động và hiển thị đúng trên các thiết bị khác nhau hay không. Điều này đặc biệt quan trọng trong môi trường di động, nơi có nhiều loại thiết bị với kích thước màn hình, độ phân giải và hệ điều hành khác nhau.

**Các yếu tố cần kiểm tra trong kiểm thử giao diện di động:**

* Hiển thị đúng trên mọi kích thước màn hình: Đảm bảo rằng ứng dụng hiển thị chính xác và tương thích trên các thiết bị khác nhau (điện thoại, máy tính bảng) và hệ điều hành khác nhau (iOS, Android, Windows Phone).
* Độ tương tác của người dùng: Kiểm tra các yếu tố như nút, thanh cuộn, menu thả xuống, và các thao tác như chạm, kéo, vuốt, và cử chỉ chạm hai lần.
* Khả năng phản hồi: Ứng dụng cần phải phản hồi nhanh và mượt mà với các thao tác của người dùng mà không có độ trễ đáng kể.
* Tính nhất quán: Kiểm tra xem giao diện người dùng có tuân thủ theo nguyên tắc thiết kế của nền tảng hay không (Material Design cho Android, Human Interface Guidelines cho iOS).
* Tính thân thiện với người dùng: Đảm bảo giao diện dễ hiểu và thân thiện với người dùng.

**Công cụ hỗ trợ:**

* **Appium:** Một công cụ kiểm thử giao diện di động tự động hỗ trợ nhiều hệ điều hành như Android và iOS.
* **Ranorex**: Một công cụ mạnh mẽ cho kiểm thử giao diện đa nền tảng.

### Kiểm thử chức năng (Functional Testing)

Kiểm thử chức năng tập trung vào việc kiểm tra tính chính xác của các chức năng của ứng dụng di động theo yêu cầu đã được định sẵn. Đây là quá trình đảm bảo rằng các chức năng cụ thể của ứng dụng hoạt động đúng như mong đợi trên các thiết bị và hệ điều hành khác nhau.

**Các yếu tố trong kiểm thử chức năng di động:**

* Điều hướng: Đảm bảo rằng người dùng có thể điều hướng giữa các màn hình và chức năng khác nhau mà không gặp sự cố.
* Thao tác trên dữ liệu: Kiểm tra các thao tác liên quan đến dữ liệu, chẳng hạn như nhập liệu, gửi dữ liệu, lưu dữ liệu và xử lý thông tin.
* Khả năng tích hợp: Kiểm tra khả năng tương thích của ứng dụng với các dịch vụ bên ngoài như thanh toán, chia sẻ, hoặc sử dụng các API khác.
* Thông báo (Push Notifications): Kiểm tra xem các thông báo của ứng dụng có hiển thị và hoạt động đúng cách không.
* Hiệu suất mạng: Đảm bảo ứng dụng hoạt động trơn tru với nhiều loại kết nối mạng khác nhau (Wi-Fi, 3G, 4G, 5G) và trong điều kiện không có kết nối mạng.

**Công cụ hỗ trợ:**

* **Robotium:** Công cụ hỗ trợ kiểm thử chức năng tự động trên ứng dụng Android.
* **Selendroid:** Một công cụ kiểm thử tự động khác dành cho ứng dụng Android, hỗ trợ Selenium.

### Kiểm thử đơn vị (Unit Testing)

Kiểm thử đơn vị là quá trình kiểm tra các thành phần nhỏ nhất của phần mềm, thường là các hàm, phương thức hoặc các module riêng biệt. Trong môi trường di động, kiểm thử đơn vị là rất quan trọng để đảm bảo rằng các khối chức năng nhỏ của ứng dụng hoạt động đúng.

**Các yếu tố trong kiểm thử đơn vị di động:**

* Kiểm tra phương thức và hàm: Đảm bảo rằng các phương thức hoặc hàm nhỏ nhất của ứng dụng, chẳng hạn như tính toán hay thao tác dữ liệu, hoạt động đúng với mọi đầu vào.
* Kiểm tra logic nghiệp vụ: Xác minh rằng logic của ứng dụng hoạt động chính xác trong mọi trường hợp.
* Phạm vi nhỏ và nhanh: Do kiểm thử đơn vị thường thực hiện kiểm tra các thành phần nhỏ và cụ thể, việc kiểm thử đơn vị nên thực hiện nhanh chóng và có thể chạy tự động liên tục (Continuous Integration).

**Công cụ hỗ trợ:**

* JUnit: Một framework phổ biến để thực hiện kiểm thử đơn vị trên Android.
* XCTest: Framework kiểm thử đơn vị cho iOS do Apple phát triển.

### Kiểm thử tự động

Kiểm thử tự động là quá trình thực hiện một cách tự động các bước trong một ca kiểm thử. Nó sử dụng một công cụ kiểm thử tự động nào đó để rút ngắn thời gian kiểm thử. Kiểm thử tự động hỗ trợ các kiểm thử viên rất nhiều tùy vào công cụ và các nội dung kiểm thử có thể thực hiện bằng tay hay không. Đối với những nhiệm vụ kiểm tra khó mà thực hiện bằng tay hoặc yêu cầu chi phí về nhân công là quá lớn thì sử dụng công cụ hỗ trợ là điều hết sức cần thiết.

### Mục tiêu của kiểm thử tự động

Phần mềm có khiếm khuyết là thông thường và gây ra thiệt hại về kinh tế theo thời gian. Chính vì vậy các tổ chức về phần mềm dành nhiều thời gian và nguồn lực để phân tích và kiểm thử phần mềm.

Ngày nay ứng dụng tự động hóa vào các ngành đa dạng, trong đó có ngành kiểm thử. Trước đây kiểm thử viên kiểm thử bằng tay thực hiện và ghi lại kết quả trên giấy, nhưng với ứng dụng công nghệ thông tin thì các công cụ kiểm thử cũng phát triển từ rất sớm hỗ trợ kiểm thử viên rất nhiều và đặc biệt là các trường hợp kiểm thử đặc biệt mà kiểm thử bằng tay không thể thực hiện được hoặc rất khó khăn để thực hiện. Các tổ chức càng quan tâm đến chất lượng phần mềm thì càng có nhiều chức năng và nội dung được kiểm tra đòi hỏi kiểm thử viên phải thực hiện nhiều công việc hơn vì vậy kiểm thử với sự hỗ trợ của công cụ là rất cần thiết.

Để giúp các kiểm thử viên có thể kiểm thử tự động đó là các Test Tool, tuy nhiên không phải trong bất cứ trường hợp nào cũng có thể kiểm thử tự động. Vậy kiểm thử thử tự động khi nào?

* Kiểm thử tự động trong các tình huống sau:
* Không đủ tài nguyên:
* Khi số lượng tình huống kiểm tra quá nhiều mà các kiểm thử viên không thể hoàn tất bằng tay trong thời gian cụ thể nào đó.

Có thể lấy một dẫn chứng là khi thực hiện kiểm tra chức năng của một website. Website này sẽ được kiểm tra với 6 môi trường gồm 4 trình duyệt và 2 hệ điều hành. Tình huống này đòi hỏi số lần kiểm tra tăng lên và lặp lại 6 lần so với việc kiểm tra cho một môi trường cụ thể.

* Kiểm tra hồi quy:

Trong quá trình phát triển phần mềm, nhóm lập trình thường đưa ra nhiều phiên bản phần mềm liên tiếp để kiểm tra. Thực tế cho thấy việc đưa ra các phiên bản phần mềm có thể là hàng ngày, mỗi phiên bản bao gồm những tính năng mới, hoặc tính năng cũ được sửa lỗi hay nâng cấp. Việc bổ sung hoặc sửa lỗi mã nguồn cho những tính năng ở phiên bản mới có thể làm cho những tính năng khác đã kiểm tra tốt chạy sai mặc dù phần mã nguồn của nó không hề chỉnh sửa. Để khắc phục điều này, đối với từng phiên bản, kiểm thử viên không chỉ kiểm tra chức năng mới hoặc được sửa, mà phải kiểm tra lại tất cả những tính năng đã kiểm tra tốt trước đó. Điều này khó khả thi về mặt thời gian nếu kiểm tra thủ công.

* Kiểm tra vận hành phần mềm trong môi trường đặc biệt:

Đây là kiểm tra nhằm đánh giá xem vận hành của phần mềm có thỏa mãn yêu cầu đặt ra hay không. Thông qua đó kiểm thử viên có thể xác định được các yếu tố về phần cứng, phần mềm ảnh hưởng đến khả năng vận hành của phần mềm.

* Mục tiêu của kiểm thử tự động:
* Giảm bớt công sức và thời gian thực hiện.
* Tăng độ tin cậy.
* Giảm sự nhàm chán.
* Giảm chi phí tổng cho quá trình kiểm thử.
* Ưu điểm của kiểm thử tự động:
* Kiểm thử phần mềm không cần can thiệp của kiểm thử viên.
* Giảm chi phí khi thực hiện kiểm tra số lượng lớn ca kiểm thử hoặc ca kiểm thử lặp lại nhiều lần.
* Giả lập được các tình huống khó có thể thực hiện bằng tay.

### Nguyên tắc kiểm thử tự động

Thực sự là sai lầm khi nghĩ tự động là đơn giản chụp lại, ghi lại một tiến trình kiếm thử thủ công. Thực tế, kiểm thử tự động có những điểm khác với kiểm thử thủ công. Nó có những lỗi và khả năng dự đoán.

Vì thế, những cơ hội thành công với kiểm thử tự động sẽ được cải thiện đáng kể trong trường hợp bạn thực sự hiểu nó.

Kiểm thử tự động tuân theo đầy đủ những nguyên tắc kiểm thử nói chung, đó là các nguyên tắc sau:

* *Nguyên tắc 1 – Kiểm thử đưa ra lỗi:*

Kiểm thử có thể cho thấy rằng phần mềm đang có lỗi, nhưng không thể chứng minh rằng phần mềm không có lỗi. Kiểm thử làm giảm xác suất lỗi chưa tìm thấy vẫn còn trong phần mềm, thậm chí là không còn lỗi nào, nó vẫn không phải là bằng chứng của sự chính xác.

* *Nguyên tắc 2 – Kiểm thử mọi thứ là không thể*

Kiểm thử mọi thứ (tất cả các tổ hợp của điều kiện input đầu vào) là không thể thực hiện được, trừ khi nó chỉ bao gồm một số trường hợp bình thường (ít trường hợp tổ hợp thì có thể test toàn bộ được). Thay vì kiểm thử toàn bộ, việc phân tích rủi ro và dựa trên sự mức độ ưu tiên chúng ta có thể tập trung việc kiểm thử vào một số điều cần thiết.

* *Nguyên tắc 3— Kiểm thử sớm*

Đề tìm được bug sớm, các hoạt động kiểm thử nên được bắt đầu càng sớm càng tốt trong quy trình phát triển (vòng đời phát triển) phần mềm hoặc hệ thống, và nên tập trung vào các hoạt động đã định trước.

* *Nguyên tắc 4— Sự tập trung của lỗi*

Nỗ lực kiểm thử nên tập trung một cách cân đối vào mật độ lỗi dự kiến và lỗi phát hiện ra sau đó trong các mô-đun. Một số ít các mô-đun thường chứa nhiều lỗi không phát hiện ra trong lúc kiểm thử trước khi phát hành, hoặc chịu trách nhiệm cho hầu hết các lỗi hoạt động của phần mềm.

* *Nguyên tắc 5 — Nghịch lý thuốc trừ sâu*

Nếu việc kiểm thử tương tự nhau được lặp đi lặp lại nhiều lần, thì cuối cùng sẽ có một số trường hợp kiểm thử sẽ không còn tìm thấy bất kỳ lỗi nào mới. Để khắc phục "nghịch lý thuốc trừ sâu" này, các trường hợp kiểm thử cần phải được xem xét và sửa đổi thường xuyên, và cần phải viết các ca kiểm thử mới và khác nhau để thực hiện nhiều phần khác nhau của phần mềm hoặc hệ thống để tìm ra lỗi tiềm ẩn nhiều hơn nữa.

Nguyên tắc này giống như việc trừ sâu trong nông nghiệp, nếu chúng ta cứ phun một loại thuốc với nồng độ giống nhau trong một khoảng thời gian dài thì có một số con sâu sẽ quen dần và cuối cùng việc phun thuốc giống như là tắm chúng vậy (bị lờn thuốc) => lúc đó chúng ta không thể diệt sạch chúng được. Do vậy, để diệt sạch sâu một cách hiệu quả, người ta thường thay đôi loại thuốc trừ sâu, mỗi loại chỉ dùng trong khoảng thời gian ngắn.

* *Nguyên tắc 6 — Kiểm thử theo các ngữ cảnh độc lập*

Nguyên tắc này là việc testing phụ thuộc vào ngữ cảnh, test trong nhiều ngữ cảnh khác nhau.

* **Để hiểu rõ hơn chúng ta xem ví dụ sau:**

Ví dụ: Cùng với một chương trình calculator nhưng có rất nhiều phạm

vi ứng dụng:

* Nếu test chương trình này cho cấp 1 thì chỉ cần có các chức năng cộng

trừ nhân chia.

* Nếu test chương trình này cho cấp 2 thì phải thêm chức năng tính toán

với đa thức

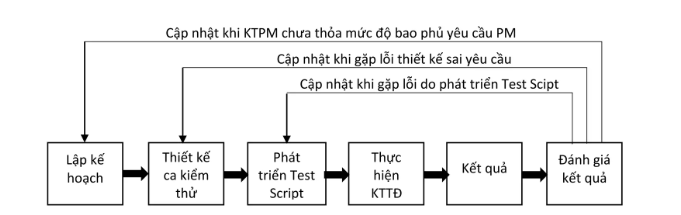
* Nếu test chương trình này cho cấp 3 thì cần thêm tính toán tích phân, đạo

hàm, v.v.

* *Nguyên tắc 7 — Sự sai lầm về việc không có lỗi*

Việc tìm và sửa chữa lỗi sẽ không giúp được gì nếu hệ thống được xây dựng xong nhưng không thể dùng được và không đáp ứng được nhu cầu và sự mong đợi của người dùng. (Nghĩa là nếu sau khi code, test rồi fix bug, làm đủ tất cả các trường hợp và cuối cùng cho ra một sản phẩm không như mong đợi hoặc không đáp ứng được nhu cầu của khách hàng thì dự án phần mềm đó coi như thất bại mặc dù đã được test xong).

### Quy trình kiểm thử tự động

Quy trình kiểm thử tự động phần mềm cũng giống như quy trình thực hiện kiểm thử thủ công chỉ khác ở chỗ kiểm thử tự động có hỗ trợ của công cụ ít hoặc nhiều như tạo test script (có thể bằng tay hoặc công cụ), công cụ hỗ trợ về ghi lại kết quả và lưu trữ kết quả trong máy tính. Quy trình này cũng gần] tương tự với quy trình phát triển phần mềm, được thực hiện qua nhiều bước, được tiến hành rất sớm trong quy trình phát triển phần mềm và đội kiểm thử tiến hành gần như song song cùng đội phát triển phần mềm.

Hình 1-1. Quy trình Kiểm thử tự động trong mỗi quan hệ với Kiểm thử phần mêm

Để kiểm thử tự động thì công cụ là thành phần không thể thiếu trong tiến trình này, việc kiểm thử viên thành thạo các công cụ kiểm thử đảm bảo cho quy trình kiểm thử tự động được hiệu quả.

### So sánh giữa kiểm thử di động và các môi trường khác

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Kiểm thử di động** | **Kiểm thử trên các môi trường khác (Web/desktop)** |
| **Giao diện (UI)** | - Đa dạng kích thước màn hình và độ phân giải, nhiều thiết bị (điện thoại, tablet). | - Kích thước màn hình ổn định hơn (web: trình duyệt khác nhau, desktop: cố định). |
| - Chạy trên nhiều hệ điều hành khác nhau (Android, iOS, Windows Phone). | - Web: Chạy trên các trình duyệt (Chrome, Firefox); Desktop: trên hệ điều hành cố định. |
| - Phải tương thích với các thao tác cảm ứng (chạm, vuốt, kéo). | - Tương tác chủ yếu qua chuột và bàn phím. |
| **Chức năng** | - Tương tác trực tiếp với phần cứng (GPS, camera, cảm biến, mạng di động). | - Web: Hạn chế tương tác với phần cứng; Desktop: tương tác với phần cứng cơ bản (bàn phím, chuột). |
| - Kiểm tra trong nhiều tình huống mạng (3G, 4G, 5G, Wi-Fi). | - Web: Thường trên mạng ổn định (Wi-Fi, Ethernet); Desktop: Mạng cục bộ hoặc offline. |
| - Phải kiểm tra tương tác với ứng dụng nền (thông báo đẩy, tin nhắn). | - Không có hoặc ít thông báo nền. |
| **Kiểm thử đơn vị** | - Phức tạp do có sự phụ thuộc vào phần cứng và hệ điều hành. | - Ít phụ thuộc vào phần cứng, dễ kiểm tra độc lập. |
| - Cần mô phỏng các thành phần phần cứng để kiểm tra. | - Các lớp backend hoặc giao diện dễ kiểm tra do ít phụ thuộc phần cứng. |

## Một số khái niệm khác

* **Security Test (Kiểm thử bảo mật):**

Đây là quá trình kiểm tra ứng dụng để tìm ra các điểm yếu hoặc lỗ hổng có thể bị khai thác bởi kẻ tấn công. Mục tiêu của kiểm thử bảo mật là đảm bảo rằng các hệ thống và dữ liệu của ứng dụng di động không bị xâm phạm. Kiểm thử bảo mật bao gồm kiểm tra các vấn đề như xác thực, ủy quyền, mã hóa, và khả năng chịu đựng của hệ thống trước các cuộc tấn công.

* **Parameter Tampering (Giả mạo tham số):**

Đây là loại tấn công mà kẻ tấn công thay đổi các tham số (parameters) trong ứng dụng di động, thường là những tham số gửi từ máy khách đến máy chủ. Ví dụ, thay đổi giá trị của tham số "giá sản phẩm" trong một ứng dụng mua sắm để mua sản phẩm với giá rẻ hơn. Điều này đặc biệt quan trọng khi kiểm thử các ứng dụng có giao tiếp giữa máy khách và máy chủ.

* **SQL Injection (Chèn SQL):**

Đây là một lỗ hổng bảo mật xảy ra khi kẻ tấn công chèn các câu lệnh SQL độc hại vào ứng dụng thông qua các trường đầu vào. Ứng dụng không lọc kỹ các đầu vào này và chạy các câu lệnh SQL độc hại trên cơ sở dữ liệu, dẫn đến việc kẻ tấn công có thể truy cập, sửa đổi hoặc xóa dữ liệu không được phép.

* **Cross-Site Scripting (XSS - Tấn công kịch bản chéo trang):**

Trong tấn công XSS, kẻ tấn công chèn mã JavaScript độc hại vào trang web hoặc ứng dụng di động, sau đó mã này được thực thi trên trình duyệt của người dùng khác. Điều này có thể dẫn đến việc đánh cắp thông tin người dùng, cookies, hoặc gây rối các phiên hoạt động.

* **Operating System Commanding (Tấn công thông qua lệnh hệ điều hành):**

Kẻ tấn công gửi các lệnh hệ điều hành độc hại qua ứng dụng để điều khiển hệ thống đích. Điều này thường xảy ra khi ứng dụng cho phép người dùng thực thi các lệnh hệ điều hành, nhưng không kiểm tra hoặc hạn chế đầu vào một cách chính xác.

* **Directory Traversal Attack (Tấn công xuyên qua thư mục):**

Đây là một lỗ hổng bảo mật mà kẻ tấn công cố gắng truy cập các tệp hoặc thư mục không được phép bằng cách thay đổi đường dẫn tập tin trong ứng dụng. Ví dụ: kẻ tấn công có thể sử dụng chuỗi "../" để di chuyển lên các thư mục cha và truy cập vào các tệp nhạy cảm như tập tin cấu hình hệ thống hoặc cơ sở dữ liệu.

* **Buffer Overflow (Tràn bộ đệm):**

Tràn bộ đệm xảy ra khi kẻ tấn công gửi dữ liệu lớn hơn kích thước bộ đệm mà chương trình có thể xử lý, khiến bộ đệm bị tràn và ghi đè lên các phần khác của bộ nhớ. Điều này có thể dẫn đến sự cố hệ thống hoặc kẻ tấn công có thể thực thi mã độc trên hệ thống mục tiêu.

**Source Code Disclosure (Lộ mã nguồn):**

Đây là một lỗ hổng bảo mật mà kẻ tấn công có thể truy cập hoặc tiết lộ mã nguồn của ứng dụng di động. Lộ mã nguồn có thể cho phép kẻ tấn công tìm ra các lỗ hổng hoặc khai thác các đoạn mã yếu để tấn công hệ thống.

## Một số công cụ kiểm thử tự động

* **Selenium**

Selenium là một công cụ kiểm tra phần mềm được sử dụng đề kiểm tra hồi quy. Đây là một công cụ kiểm tra mã nguồn mở cung cấp chức năng phát lại và thu âm để kiểm tra hồi quy. Các Selenium IDE chỉ hỗ trợ trình duyệt web Mozilla Firefox.

* **QTP (HP UFT)**

QTP được sử dụng rộng rãi để kiểm thử chức năng và hồi quy, giải quyết các ứng dụng phần mềm và môi trường. Đề đơn giản hóa việc tạo và bảo trì thử nghiệm, nó sử dụng khái niệm kiêm tra từ khóa.

* **NaHonal Function Tester**

Là 1 công cụ kiểm tra tự động hướng đối tượng có khả năng tự động Kiểm tra dữ liệu, kiểm tra giao diện, và kiểm thử hồi quy.

* **Watir**

Là một phần mềm kiểm tra mã nguồn mở để kiểm thử hồi quy. Watir chỉ hỗ trợ khám phá Internet trên các cửa số trong khi Watir webdriver hỗ trợ Chrome, Firefox, Opera, v.v.

* **Appium Studio**

Sẽ được đề cập đến trong phần thực nghiệm.

# THỰC NGHIỆM SỬ DỤNG APPIUM CHO KIỂM THỬ TỰ ĐỘNG TRÊN ANDROID

Chương cuối này sẽ là kết quả của quá trình thực nghiệm khi sử dụng một phần mềm điển hình cho việc kiểm thử tự động trên Appium.

## Giới thiệu phần mềm Appium

### Công cụ Appium

* **Appium:** là một công cụ mã nguồn mở, đa nền tảng, được sử dụng rộng rãi để tự động hóa các test case cho các ứng dụng di động (native, hybrid, mobile web) trên iOS và Android.
* **Tại sao sử dụng Appium:**
* Đa nền tảng: Viết một bộ test case, chạy được trên cả iOS và Android.
* Không cần root/jailbreak: Có thể kiểm thử ứng dụng mà không cần can thiệp sâu vào hệ thống thiết bị.
* Hỗ trợ nhiều ngôn ngữ lập trình: Java, Python, Ruby, C#,...
* Cộng đồng lớn: Có nhiều tài liệu, thư viện hỗ trợ.

### Nguyên lý hoạt động của Appium

* Appium hoạt động dựa trên nguyên tắc client-server. Client là một thư viện mà bạn sử dụng để viết các kịch bản kiểm thử, trong khi server là một tiến trình chạy trên máy tính của bạn để điều khiển các thiết bị di động. Khi bạn chạy một kịch bản kiểm thử, client sẽ gửi các lệnh tới server, server sẽ dịch các lệnh này thành các lệnh tương ứng trên thiết bị và thực hiện các hành động.

### Phần mềm Appium

* **Appium:** là một IDE (Integrated Development Environment) được xây dựng trên nền tảng Appium. Nó cung cấp một giao diện trực quan, dễ sử dụng hơn so với việc viết code thuần túy bằng các ngôn ngữ lập trình.
* **Ưu điểm của Appium:**
* Giao diện trực quan: Dễ dàng tạo, chỉnh sửa và thực thi các test case.
* Tích hợp với nhiều framework: Hỗ trợ nhiều framework kiểm thử phổ biến như Selenium, JUnit, TestNG.
* Đơn giản hóa quá trình thiết lập: Cài đặt và cấu hình Appium Studio nhanh chóng.
* Debug trực quan: Giúp tìm và sửa lỗi trong test case dễ dàng hơn.
* **Nhược điểm của Appium:**
* Khả năng tùy biến hạn chế: Không linh hoạt bằng việc viết code thuần túy, khó thực hiện các thao tác phức tạp, phụ thuộc vào các tính năng có sẵn của công cụ.
* Hiệu năng có thể thấp hơn: Do lớp trừu tượng và các tính năng bổ sung, hiệu năng có thể bị ảnh hưởng, không phù hợp với các dự án yêu cầu hiệu năng cao.

## Lợi ích kiểm thử tự động hoá thiết bị bằng Appium

### ****Tăng độ bao phủ của bài kiểm thử:****

* **Kiểm tra toàn diện các tính năng:** Appium cho phép chúng ta tương tác với nhiều loại tính năng tự động hóa khác nhau như Bluetooth, GPS, camera, micro,... giúp đảm bảo rằng tất cả các tính năng đều hoạt động đúng như mong đợi.
* **Phát hiện lỗi sớm:** Bằng cách tự động hóa các tương tác với các tính năng này, chúng ta có thể phát hiện ra các lỗi tiềm ẩn ngay từ giai đoạn sớm của quá trình phát triển.

### ****Nâng cao hiệu quả kiểm thử:****

* **Tự động hóa các tác vụ lặp đi lặp lại:** Thay vì phải thực hiện thủ công các tác vụ như kết nối Bluetooth, bật GPS, chụp ảnh,... Appium giúp tự động hóa các tác vụ này, tiết kiệm thời gian và công sức cho người kiểm thử.
* **Tiết kiệm chi phí:** Việc giảm thiểu thời gian kiểm thử và tăng hiệu quả sẽ giúp tiết kiệm chi phí cho doanh nghiệp.

### ****Đảm bảo chất lượng sản phẩm:****

* **Kiểm tra tính tương thích:** Appium giúp kiểm tra tính tương thích của ứng dụng với các phiên bản hệ điều hành khác nhau và các loại thiết bị khác nhau.
* **Cải thiện trải nghiệm người dùng:** Bằng cách đảm bảo các tính năng tự động hóa hoạt động trơn tru, chúng ta góp phần nâng cao trải nghiệm người dùng.

### ****Linh hoạt và mở rộng:****

* **Hỗ trợ đa nền tảng:** Appium hỗ trợ cả iOS và Android, giúp chúng ta kiểm thử trên nhiều nền tảng khác nhau.
* **Khả năng tùy biến cao:** Appium cung cấp nhiều tùy chọn để tùy chỉnh các kịch bản kiểm thử, phù hợp với các yêu cầu cụ thể của từng dự án.

### ****Tích hợp với các công cụ khác:****

* **CI/CD:** Appium có thể dễ dàng tích hợp vào các pipeline CI/CD để tự động hóa quá trình xây dựng và kiểm thử.
* **Báo cáo:** Appium hỗ trợ tạo các báo cáo chi tiết về kết quả kiểm thử, giúp dễ dàng theo dõi và phân tích.

## Thực hiện Appium bằng IntelliJ IDEA

Phần này sẽ hướng dẫn cách thiết lập và sử dụng **Appium** trong **IntelliJ IDEA** để kiểm thử một chương trình chạy trên hệ điều hành Android. Quá trình này bao gồm việc cài đặt các công cụ cần thiết, cấu hình môi trường kiểm thử và xây dựng dự án kiểm thử tự động.

* Trước tiên, bạn cần cài đặt các thành phần quan trọng để thiết lập môi trường kiểm thử:
* **Cài đặt Android Studio**: Android Studio sẽ cung cấp Android SDK và máy ảo Android (AVD) để kiểm thử ứng dụng.
* **Cài đặt JDK 23**: Java Development Kit (JDK) phiên bản 23 là công cụ cần thiết để phát triển và chạy các ứng dụng Java. Sau khi cài đặt, hãy đảm bảo thiết lập biến môi trường JAVA\_HOME để tích hợp với các công cụ khác.
* **Cài đặt Node.js**: Node.js là runtime được sử dụng bởi Appium server. Việc cài đặt Node.js là bắt buộc để khởi động và quản lý Appium server.
* **Cài đặt Appium**: Appium là công cụ kiểm thử tự động cho các ứng dụng di động. Bạn có thể cài đặt Appium thông qua **npm** (Node Package Manager) với lệnh:

|  |
| --- |
| *npm install -g appium* |

* Điều này sẽ giúp bạn khởi động Appium server để giao tiếp với thiết bị Android trong quá trình kiểm thử.
* **Cài đặt Uiautomator2**: Đây là framework cần thiết để Appium có thể tương tác với các thành phần giao diện người dùng trên thiết bị Android. Nó được tích hợp sẵn trong Appium và sẽ tự động sử dụng trong quá trình kiểm thử Android.
* **Cài đặt Appium Inspector**: Công cụ này giúp bạn kiểm tra và xác định các phần tử giao diện người dùng trong ứng dụng Android, từ đó có thể xây dựng các trường hợp kiểm thử chính xác.
* **Cài đặt IntelliJ IDEA**: Đây là IDE được sử dụng để viết mã kiểm thử tự động. IntelliJ IDEA hỗ trợ nhiều công cụ như **Maven** hoặc **Gradle** để quản lý các thư viện phụ thuộc trong dự án kiểm thử.
* **Tải Apache Maven**: Maven là công cụ quản lý dự án giúp tự động hóa việc xây dựng, quản lý và giải quyết các thư viện phụ thuộc trong dự án kiểm thử. Bạn sẽ sử dụng Maven để quản lý các thư viện như **Appium Java Client** và **Selenium**.

=> Sau khi các công cụ trên đã được cài đặt, tiến hành **xây dựng bộ ca kiểm thử** cho ứng dụng cần kiểm thử, bao gồm các trường hợp kiểm tra các chức năng quan trọng của ứng dụng. Cuối cùng, bạn sẽ **tạo dự án kiểm thử** trong IntelliJ IDEA và bắt đầu viết mã kiểm thử tự động sử dụng Appium.

### Cài đặt Android Studio

**Bước 1:** Truy cập đường dẫn: [https://developer.android.com/studio?](https://developer.android.com/studio?gad_source=1&gclid=Cj0KCQjwsc24BhDPARIsAFXqAB1angMhNZYmvzRR2rwOnQRxLxPRzdMtNP5E4rQJqnmfcQ3G-mMs_p4aAokUEALw_wcB&gclsrc=aw.ds&hl=vi)

**Bước 2:** Tiến hành tải và cài đặt Android Studio

**Bước 3:** Chọn và cài đặt SDK

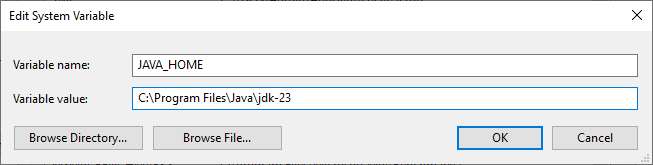
### Cài đặt JDK 23

**Bước 1:** Truy cập đường dẫn: [**https://www.oracle.com/vn/java/technologies/downloads/**](https://www.oracle.com/vn/java/technologies/downloads/)

**Bước 2:** Tiến hành tải và cài đặt

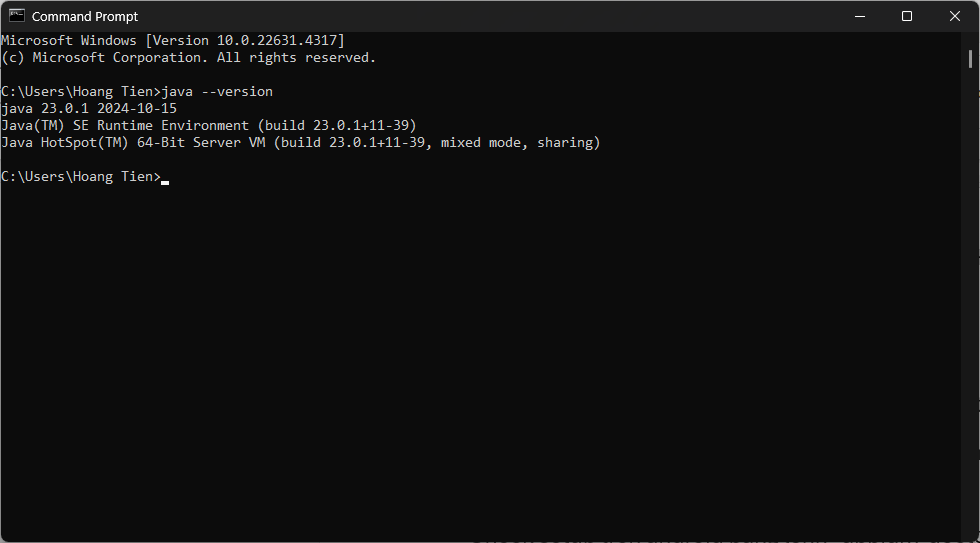
**Bước 3:** Truy Environment Variables

**Bước 4:** Thêm JAVA\_HOME và đường dẫn đến thư mục jdk-23



Hình 2-1.Thêm vào JAVA\_HOME

**Bước 5:** Mở cmd

**Bước 6:** Nhập “java –version” 

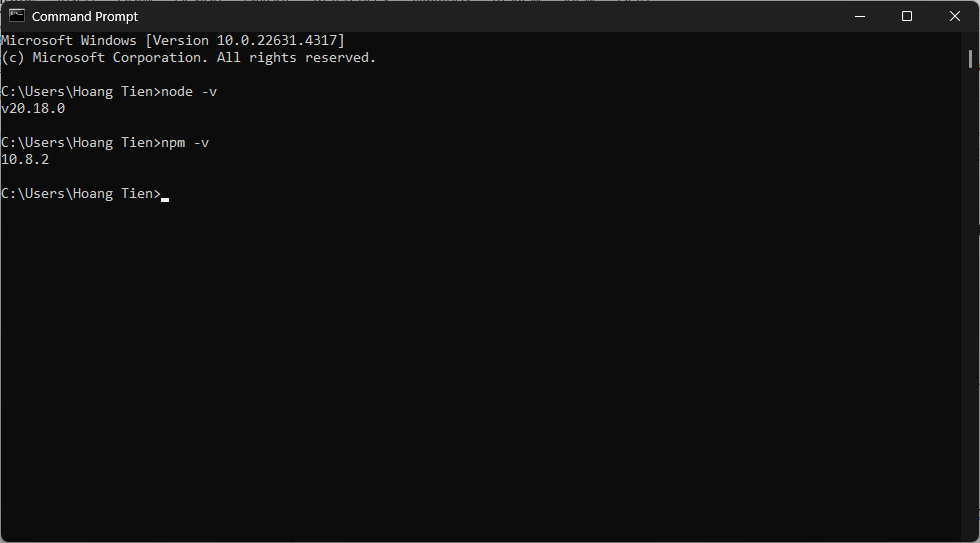
Hình 2-2. Xem java version

### Cài đặt Nodejs

**Bước 1:** Truy cập: <https://nodejs.org/en>

**Bước 2:** Tải và cài đặt

**Bước 3:** Mở cmd

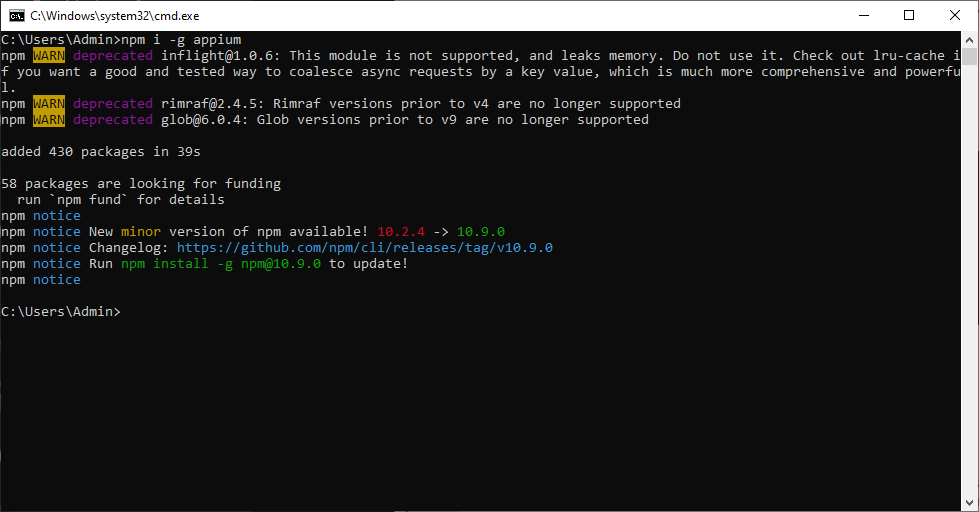
**Bước 4:** Nhập “node -v” và “npm -v” để kiểm tra phiên bản 

Hình 2-3. Kiểm tra phiên bản node

### Cài đặt Appium

**Bước 1:** Mở cmd

**Bước 2:** Nhập lệnh “npm i -g appium”

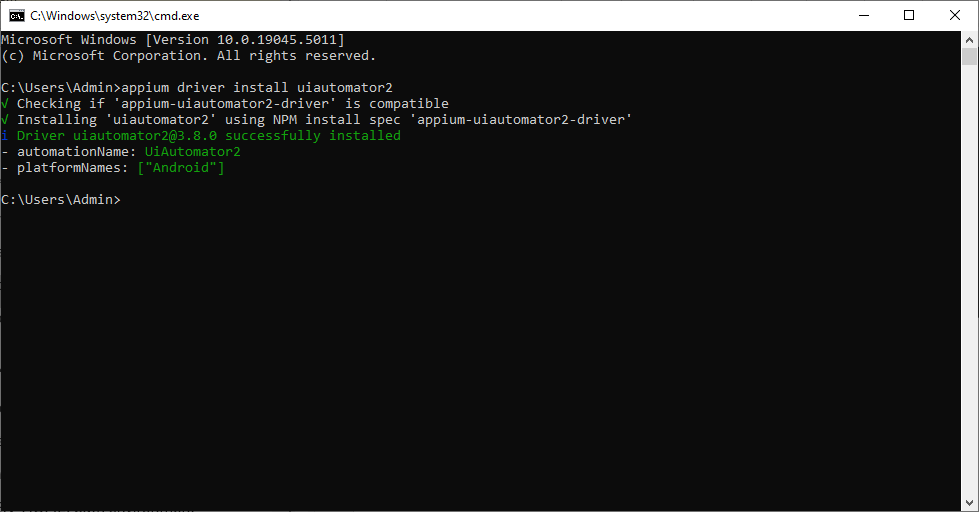


Hình 2-4. Cài đặt Appium bằng cmd

### Cài đặt Uiautomator2

**Bước 1:** Mở cmd

**Bước 2:** Nhập lệnh “appium driver install uiautomator2”

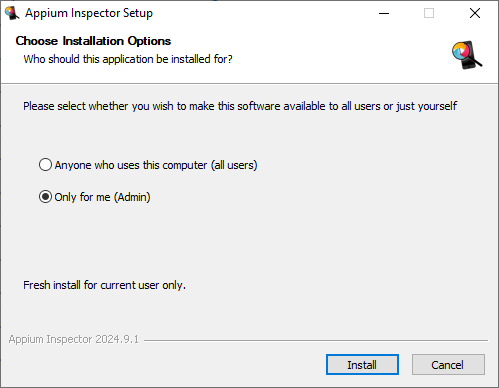


Hình 2-5. Cài đặt uiautomator2

### Cài đặt Appium Inspector

**Bước 1:** Truy cập: <https://github.com/appium/appium-inspector/releases>

**Bước 2:** Tải và cài đặt Appium-Inspector-2024.9.1-win-x64.exe

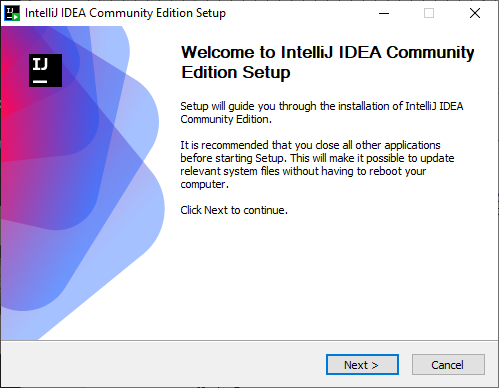


Hình 2-6. Cài đặt appium inspector

### Cài đặt Intellij IDEA

**Bước 1:** Truy cập: <https://www.jetbrains.com/idea/download/?section=windows>

**Bước 2:** Tải và cài đặt intellij IDEA Community Edition



Hình 2-7. Cài đặt intellij IDEA Community Edition

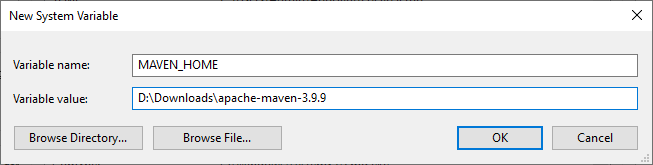
### Tải Apache maven

**Bước 1:** Truy cập : <https://maven.apache.org/download.cgi>

**Bước 2:** Tải và giải nén “apache-maven-3.9.9-bin.zip”

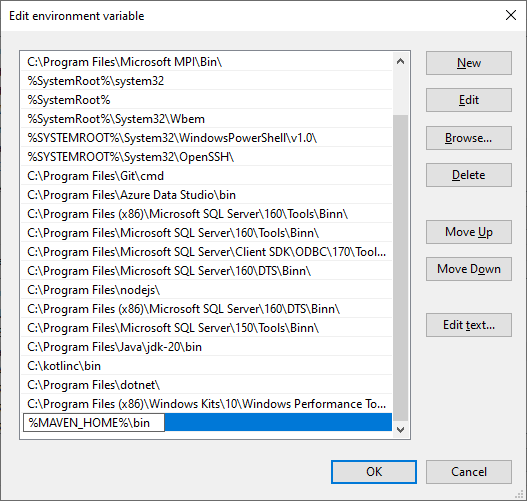
**Bước 3:** Truy Environment Variables

**Bước 4:** Thêm MAVEN\_HOME và đường dẫn đến thư mục apache-maven-3.9.9

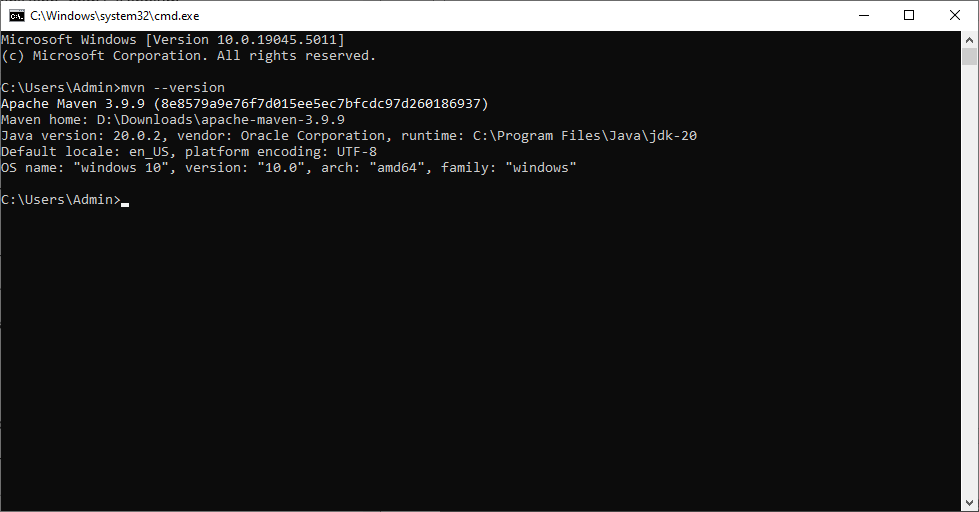


Hình 2-8. Thêm MAVEN\_HOME

**Bước 5:** Thêm đường dẫn bin vào Path



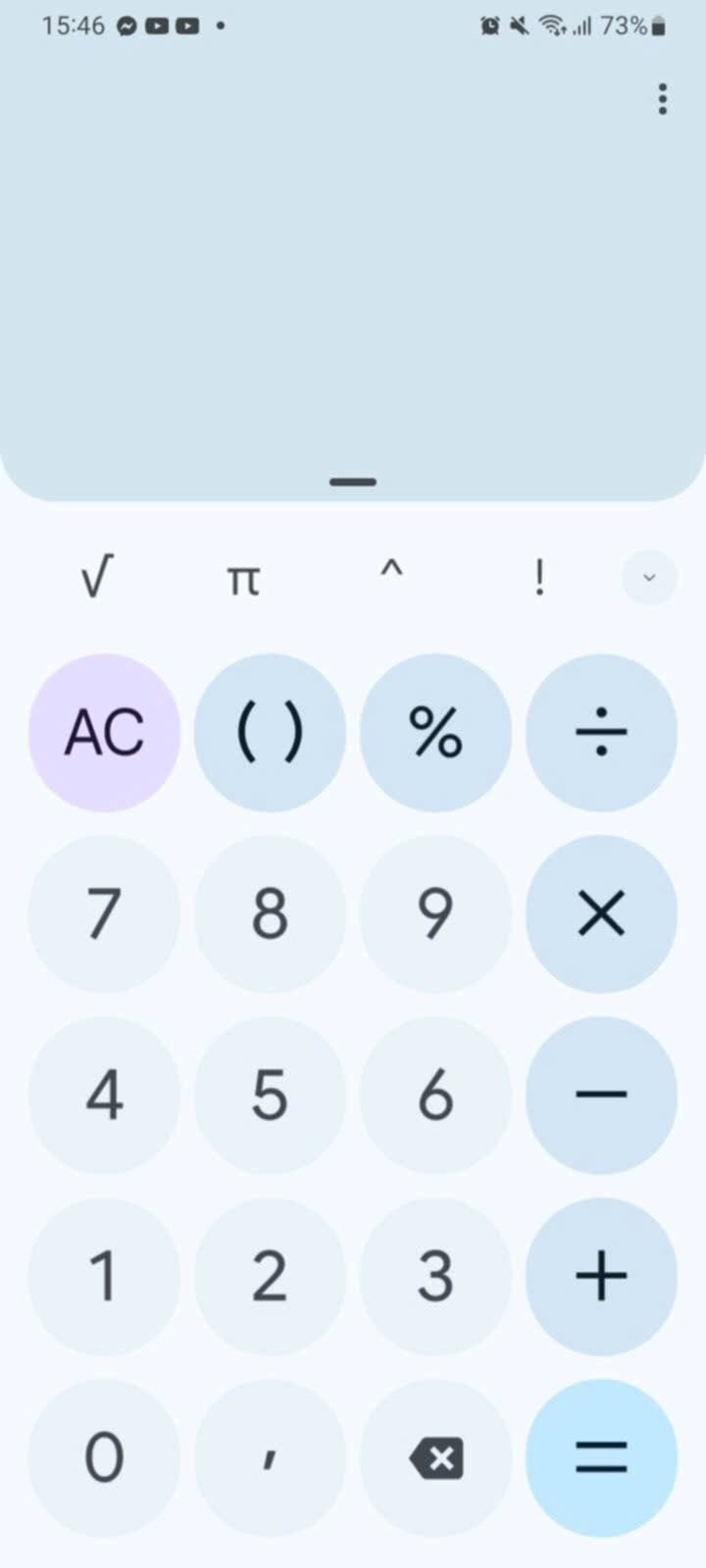
Hình 2-9. Thêm đường dẫn Path

**Bước 6:** Truy cập cmd để kiểm tra phiên bản maven bằng lệnh “mvn - -version” 

Hình 2-10. Xem maven version

### Xây dựng bộ ca kiểm thử cho ứng dụng cần kiểm thử

Ở đây ta sẽ tiến hành kiểm thử một chương trình máy tính đơn giản và dùng kiểm thử hộp đen để kiểm tra các chức năng cho phần mềm.



Hình 2-11. Màn hình ứng dụng test

**Yêu cầu của phần mềm:**

* Có đầy đủ các chức năng +, -, \*, /.
* Tính toán được các số tự nhiên và các số thập phân.
* Tính toán được với số âm và số dương.
* Hiện kết quả ERROR với những lỗi toán học cơ bản.

Áp dụng phương pháp phân vùng tương đương để xây dựng các ca kiểm thử:

1. *Xác định các lớp tương đương:*

Phân vùng 1: Số tự nhiên âm.

Phân vùng 2: Số tự nhiên dương.

Phân vùng 3: Số thập phân âm.

Phân vùng 4: Số thập phân dương.

Phân vùng 5: Số 0.

1. *Xây dựng các ca kiểm thử:*

Ta sẽ có số kiểm thử là: (5 \* 4) / 2 \* 3 \* 4 = 120 (ca)

Với (5 \* 4) / 2 là số ghép cặp các phân vùng, 3 là số ca kiểm thử với mỗi cặp (chưa tính phép toán và hoán vị trong phép toán).

Để kiểm tra ta sẽ có một bộ các ca kiểm thử đại diện như sau:

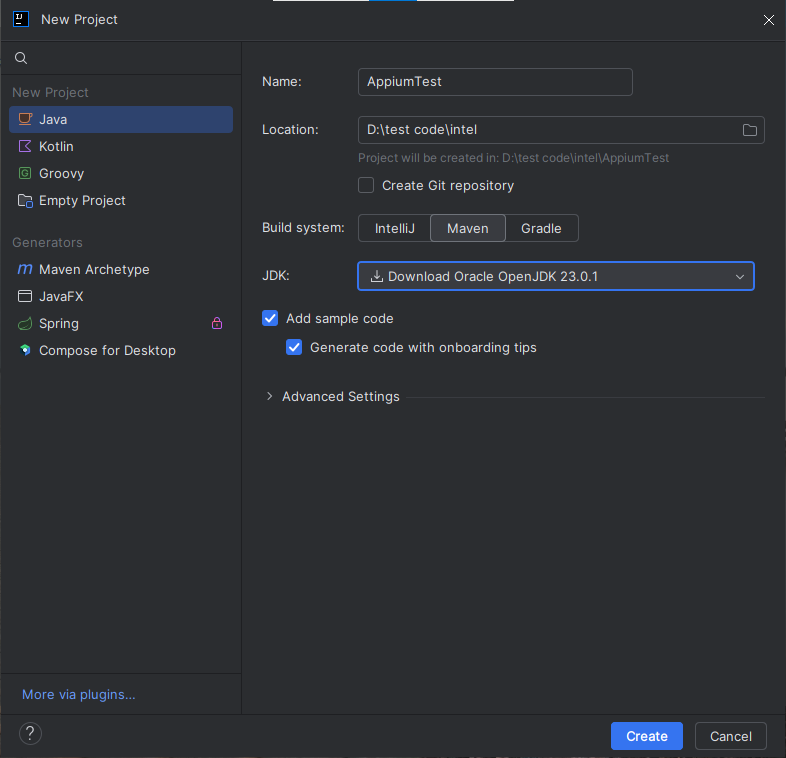
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Testcase Description** | **Test Data** | **Expected Result** |
| 1 | Kiểm thử với phép cộng số nguyên dương và số thập phân âm | 5 + (-1.6) | 3.4 |
| 2 | Kiểm thử với phép trừ số nguyên âm và số thập phân dương | -3 – 5.2 | -8.2 |
| 3 | Kiểm thử với phép nhân số nguyên âm và số thập phân âm | -2 \* (-4.2) | 8.4 |
| 4 | Kiểm thử với phép chia số thập phân dương và số nguyên dương | 6.3/8 | 0.7875 |
| 5 | Kiểm thử phép chia cho 0 | 9/0 | ERROR |

### Tạo dự án kiểm thử

#### Tạo dự án intellij IDEA

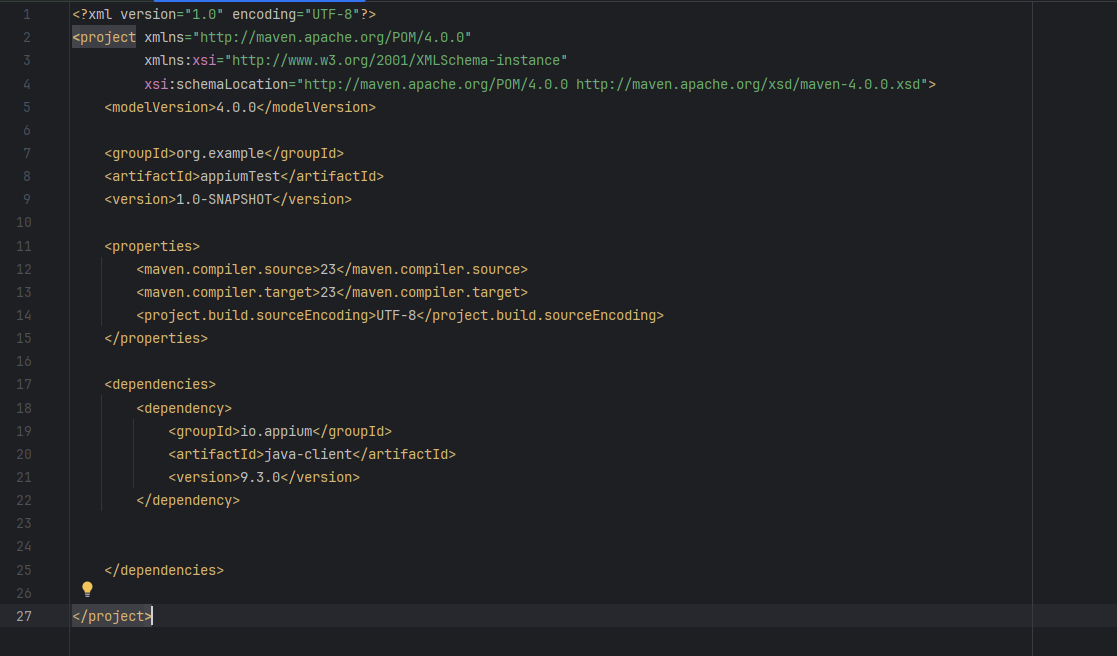
**Bước 1:** Mở intellij IDEA

**Bước 2:** Tạo Project java (build system maven)



Hình 2-12. Tạo Project

**Bước 3:** Thêm appium java client vào test framework



Hình 2-13. Thêm appium java

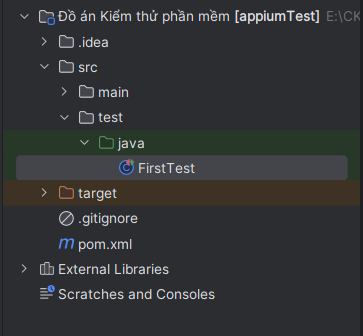
**Bước 4:** Thêm testng vào test framework

A screen shot of a computer

Description automatically generated

Hình 2-14. Thêm testng

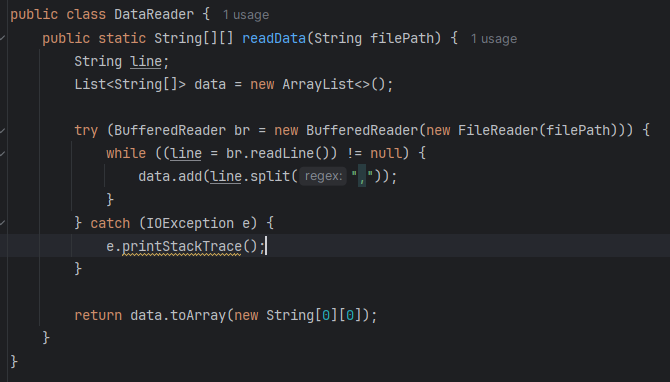
**Bước 5:** Tạo class trong thư mục test/java



Hình 2-15. Tạo class viết code

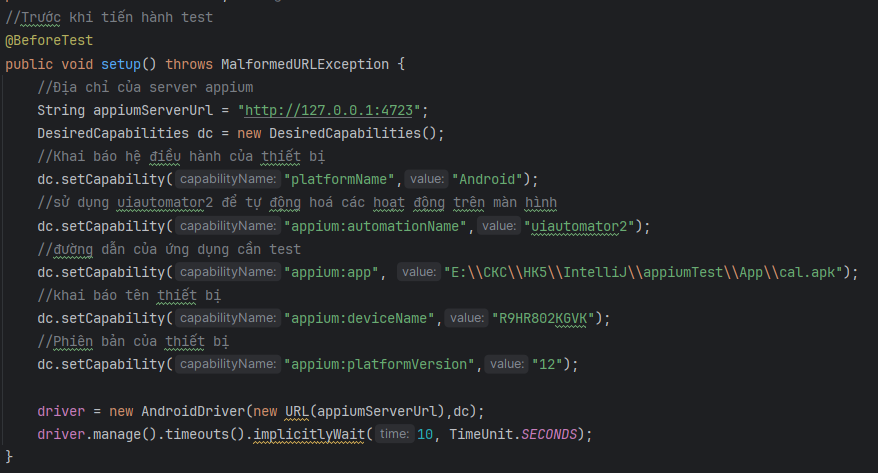
#### Viết kịch bản kiểm thử

**Bước 1:** Tạo một java class đặt tên là DataReader



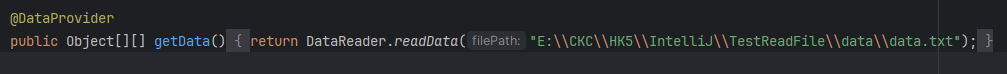
Hình 2-16. Tạo DataReader

**Bước 2:** Viết cấu hình các thông tin cần thiết



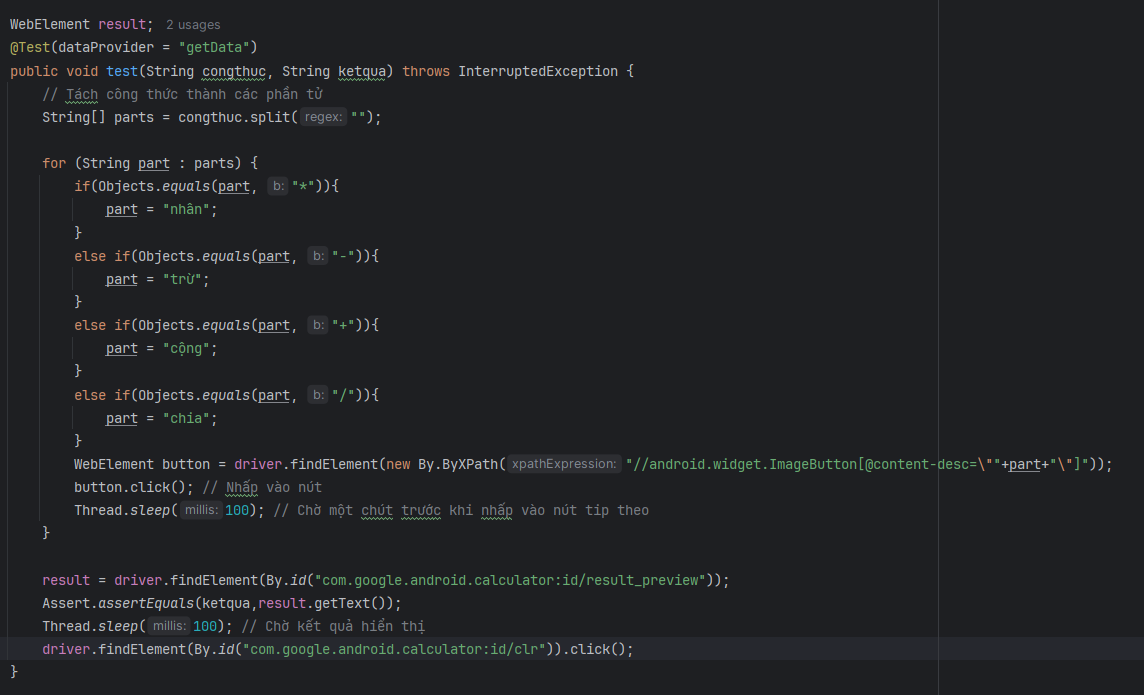
Hình 2-17. Thông tin cấu hình

**Bước 3:** Lấy dữ liệu test từ file data.txt



Hình 2-18. Lấy dữ liệu

**Bước 4:** Tiến hành test theo Test case đã đọc được từ file data.txt



Hình 2-19. Code đọc thông tin từ file

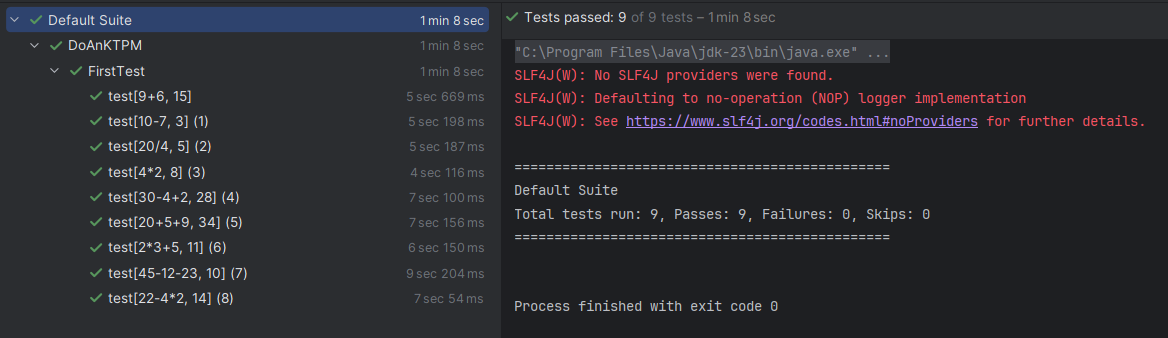
**Bước 5:** Sau khi test xong thoát ứng dụng

A computer screen shot of text

Description automatically generated

Hình 2-20. Code đóng ứng dụng

**Bước 6:** Kết quả kiểm thử



Hình 2- 21. Kết quả kiểm thử

### Báo Cáo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected Result** | **Actual Result** | **Result** |
| 1 | 5 + (-1.6) | 3.4 | 3.4 | PASS |
| 2 | -3 – 5.2 | -8.2 | -8.2 | PASS |
| 3 | -2 \* (-4.2) | 8.4 | 8.4 | PASS |
| 4 | 6.3/8 | 0.7875 | 0.7875 | PASS |
| 5 | 30 – 4 + 2 | 28 | 28 | PASS |
| 6 | 20 + 5 + 9 | 34 | 34 | PASS |
| 7 | 2 \* 3 + 5 | 11 | 11 | PASS |
| 8 | 45 – 12 - 23 | 10 | 10 | PASS |
| 9 | 9/0 | ERROR | ERROR | PASS |
| 10 | ? | ? | ? | ERROR |

# TÀI LIỆU THAM KHẢO

[1] Slide môn Kiểm Thử Phần Mềm của cô Nguyễn Thị Ngọc

[2] Mobile Testing tham khảo [TẠI ĐÂY](https://www.geeksforgeeks.org/what-is-mobile-application-testing/)

[3] Công cụ hỗ trợ Appium tham khảo [Click vào đây](https://viblo.asia/p/gioi-thieu-ve-appium-va-huong-dan-cai-dat-appium-phan-1-1Je5EYN15nL)

[4] Wikipedia, bách khoa toàn thư mở [https://vi.wikipedia.org/wiki](https://vi.wikipedia.org/wiki/Th%E1%BB%AD_nghi%E1%BB%87m_ph%E1%BA%A7n_m%E1%BB%81m)