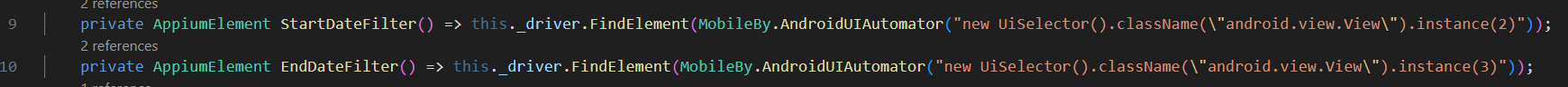
Prerequisites:

* Playwright installation
  + Dotnet new nunit
  + Dotnet add package Microsoft.Playwright.Nunit
  + Dotnet build
  + Pwsh bin/Debug/net9.0/playwright.ps1 install
  + <https://playwright.dev/dotnet/docs/intro>

Automation Strategy:

* Automation framework follows the Page Object Model design pattern
  + Each page within mobile application is modelled as a C# class as a Page Object
  + Components within a page may be modelled as a C# class as a Component Object
* For more complex pages, we may use class Composition to combine component classes within a page object class
  + Example:
    - Given a component is expected to be present on multiple pages within mobile application, it makes sense to model the component as a separate class, allowing page objects to add component as a member
* The structure of a page object classes include:  
  Example: HomePage.cs

1. Page Locators, via XPath, CSS Selector, AccessibilityId, AndroidUIAutomator, ClassName, Id, etc.



1. Component class members



1. Getters/Setters

A computer screen with colorful text

AI-generated content may be incorrect.

1. Playwright Actions

A screenshot of a computer screen

AI-generated content may be incorrect.

* Additional Helper classes are stored within the **Util** directory
  + Example:
    - RTOCalculator.cs, provides static helper methods to calculate and format the output required for validation within test script

Abstracting Page Objects:

* Page objects will represent different parts of the mobile application’s user interface. This includes:
  + Page
  + Dialog
  + Components (Calendar, HamburgerMenu…)
* To maintain modular and scalable test automation, each part of the UI has been abstracted into its own base class. These base classes will factor out common functionality, such as handling the initialization of the Appium session (Android Driver)
  + BasePage
  + BaseDialog
  + BaseComponent

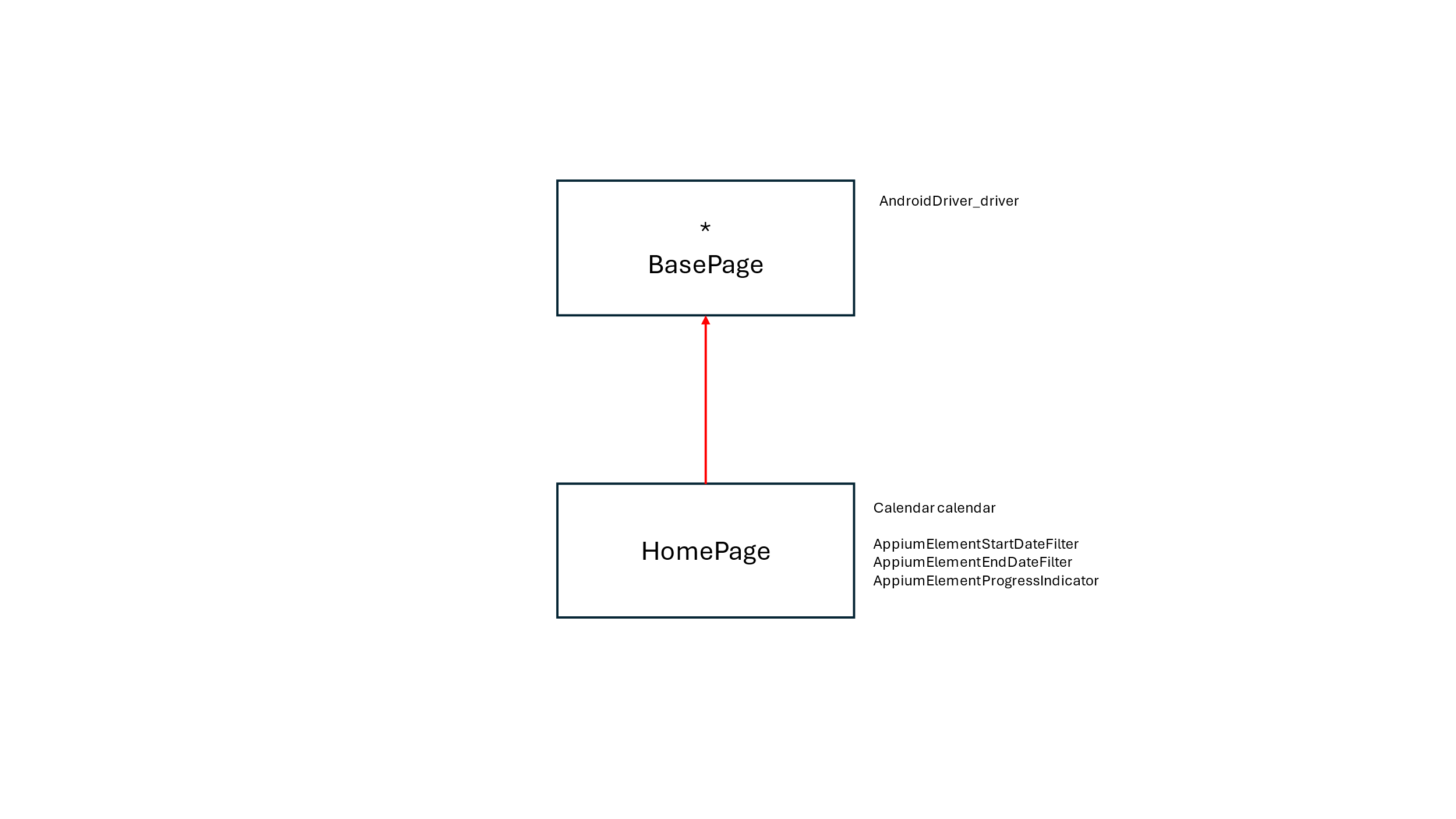
Concrete Page Object classes:

* When implementing page objects, we create concrete classes that inherit from their respective base page classes
  + Ex: HomePage inherits from BasePage
* Each page object class should be focused and cohesive, such that the class only contains element locators and methods directly related to the specific part of the mobile application this class represents

Locators

* In this automation framework, locators are returned through getter methods which follow a lazy instantiation pattern
* Locators are not resolved during page object instantiation, instead, the getter returns a reference to the locator, and the element lookup only happens when an action is performed
  + - Click()
    - Fill()
* Lazy Instantiation ensures Playwright will only search for elements at the time of interaction during test execution
* By deferring element lookup, this approach avoids unnecessary computation and risk of stale locators, since each locator call returns a fresh element

Inheritance Hierarchy Conceptually:



Data-Driven Testing:

TODO.