**Mini Project Documentation**

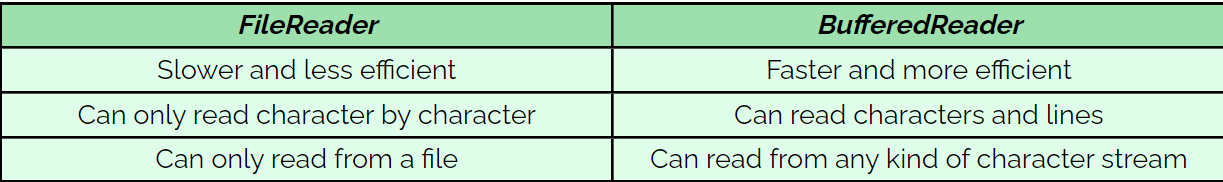
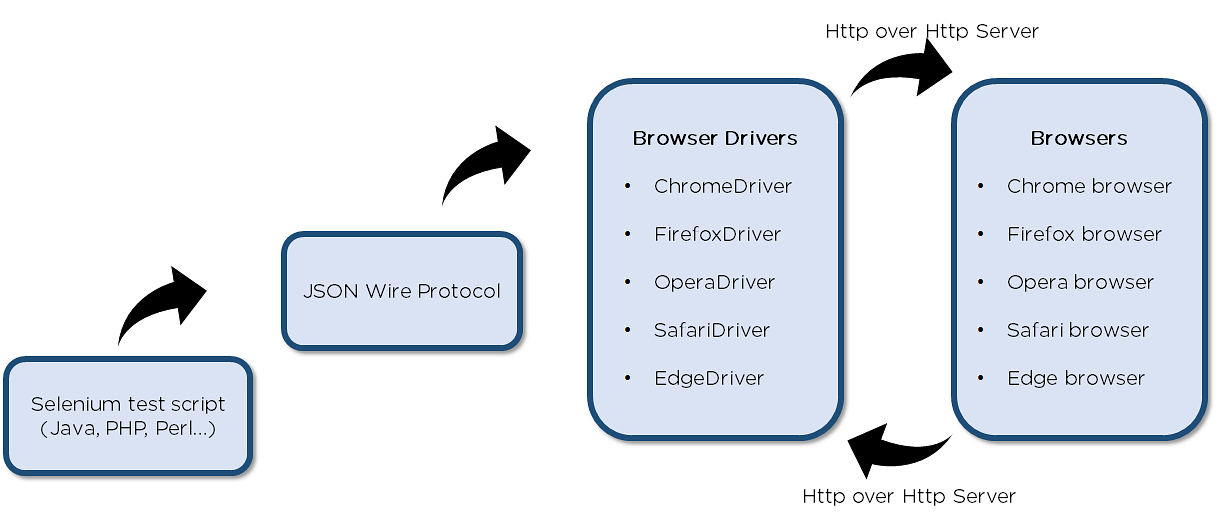
App.java

FileReader fr = **new** FileReader("C:\\Users\\2282058\\Downloads\\eclipse-java-2023-03-R-win32-x86\_64 (1)\\eclipse\\BookSearchAutomation\\Resources\\config.properties");

Properties p = **new** Properties();

p.load(fr);

String url = p.getProperty("url");

* **FileReader** class is used to read data from the give file. It returns data in byte format like FileInputStream class.
  + Constructors of FileReader class:
    - FileReader(String file): It gets the file name as a string. It opens in read mode. If any error occurs, will throw FileNotFoundException.
    - FileReader(File file): It gets the file name as file instance. It opens in read mode. If any error occurs, will throw FileNotFoundException.
  + Methods in FileReader class:
    - Int read(): It is used to return a character in ASCII form. It returns -1 at the end of file.
    - Void close(): It is used to close the FileReader class.
* BufferReader class is used to create a buffer which holds the data from the input character stream.
* 
* **Properties:**
  + .properties is the extension for files to store configurable parameters of an application. It is used for storing strings for localization.
  + It is used to maintain a list of key and value pair, where both key and value are of string type.
  + Advantages: If any data is changed from the properties record, we don’t have to recompile the java class. It is utilized to store data that is to be changed habitually.
  + Properties.load() is used to read (key, value) pair present in the properties file.
  + getProperty(key) is used to access the value from the key of the properties file.
  + If fails, it throws IOException.
* **What is System.setProperty(key, value) ?**
  + System is a Java class that belongs to the **java.lang** package.
  + System is a final class and does not provide any public constructors; hence, it cannot be instantiated.
  + This means it cannot be inherited to override its methods.
  + The system setProperty method has two attributes – “propertyName” and “value.” The propertyName represents the name of the browser-specific driver, and the value points to the path of that browser driver.
* **What is Selenium WebDriver ?**
  + Selenium WebDriver was the first cross-platform testing framework that could configure and control the browsers on the OS level
  + 
  + Selenium test script - Selenium test script is the test code written in any of the mentioned programming languages that are interpreted by the driver.
  + JSON Wire Protocol - JSON Wire Protocol provides a transport mechanism to transfer data between a server and a client. JSON Wire Protocol is the industry standard for various web services.
  + Browser drivers - Selenium uses drivers, specific to each browser to establish a secure connection with the browser.
  + Browsers - Selenium WebDriver supports multiple web browsers to test and run applications on.
  + WebDriver is an interface class and ChromeDriver and EdgeDriver are the implementation class.

PrintCategories.java

* navigate() allows us the browser to move backwards or forward in the browser’s history.
* **Difference between navigate() and get()**
  + ***driver.get()***method is used to open an URL and it will wait till the whole page gets loaded. WebDriver will wait until the page has fully loaded before returning control to your test or script.
  + While **driver.navigate.to()**method navigates to an URL and It will not wait till the whole page gets loaded. It maintains the browser history and cookies, so we **can use forward and backward button to navigate between the pages** during the coding of Testcase.
* **implicitlyWait**(): This timeout is used to specify the time the driver should wait while searching for an element if it is not immediately present.
* **Difference between implicitly.wait() and Thread.sleep()**:
  + implicitly.wait() sets a global timeout for the webdriver to wait for elements to be present, while thread.sleep pauses the execution of the entire thread for a specified duration, regardless of the element’s availability.
  + implicitly.wait() is more efficient and flexible for handling dynamic web elements.
* **Why to use timeouts?**
  + Using frameworks such as React, Angular, or any other which takes a certain time for the web elements to load on the page, whenever that page is loaded or refreshed. Hence, in case you tend to locate an element in your script which is yet to load on the page, selenium will throw you ‘**ElementNotVisibleException**’ message.
* **Why using thread.sleep() is not good?**
  + Unnecessary wait times: As it will pause the execution of some time for the element to appear or action to complete.
  + Unpredictable delays: It can cause unpredictable delays in test execution, especially in scenario where the application’s responsiveness varies. This can lead to inefficient test runs and longer execution time.
* // start with locators