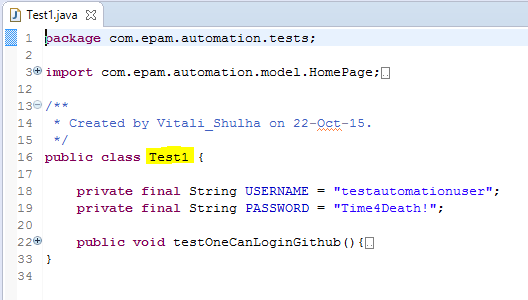
For every violation please describe the following:

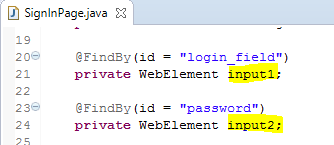
\* why this part of code is violating Clean Code practices? what is wrong with this code?

\* how this can be fixed? what you can suggest?

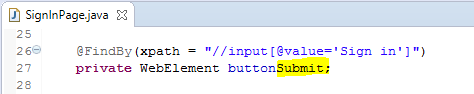
**Naming:**



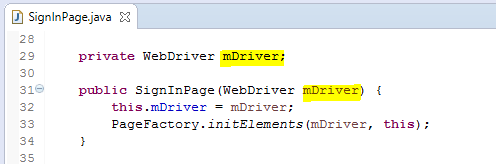
* The class name ‘Test1’ can be more meaningful. For example, it should have been ‘TestGithubLogin’ or ‘LoginTests’



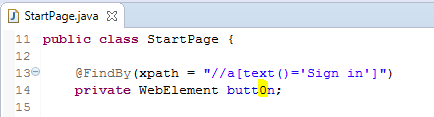
* The instance variable name ‘input1’ is not meaningful. It should have rather been ‘username’, which reveals its purpose.
* The instance variable name ‘input2’ not meaningful. It should have rather been ‘password’, which reveals its purpose.



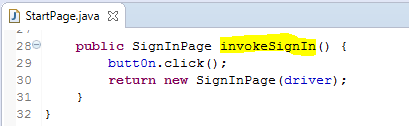
* The name ‘buttonSubmit’ can better reflect the name as visually seen in the web page, i.e. ‘Sign in’. Though the name seems meaningful, ‘buttonSubmit’ can better be renamed to ‘buttonSignIn’ to reflect functional flow as seen in GUI.



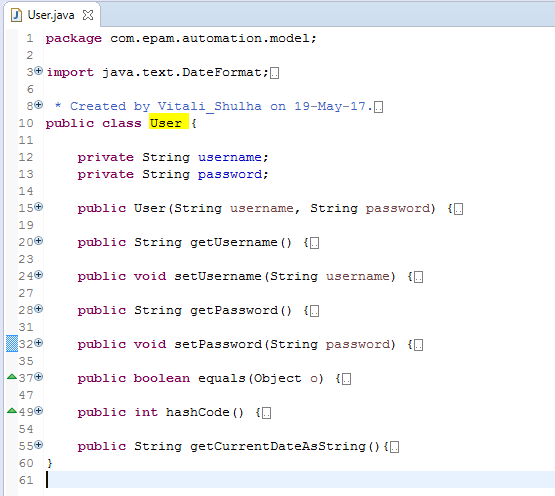
* Do not use Hungarian notation in Java. ‘mDriver’ is an obvious member of the class, and its accessibility is defined by the access modifier. So, no need for ‘m’ prefix here. The instance variable should have been named as ‘webDriver’. If the code were copied from other programming languages, do refactor for meaning names.
* So as with the argument in the constructor, ‘webDriver’ instead of ‘mDriver’.



* Typo ‘0’ crept in. Also, ‘buttonSignIn’ instead of only ‘button’.

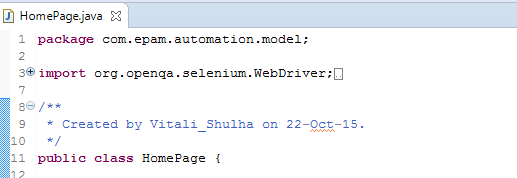
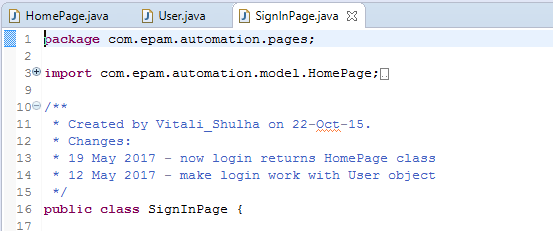


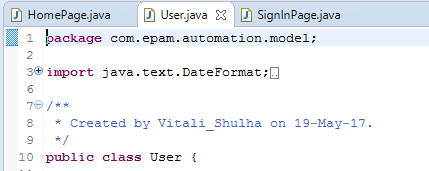
* It would have been more meaningful if the method were named ‘navigateToSignInPage()’ instead of ‘invokeSignIn()’

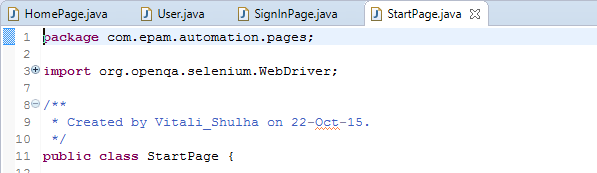
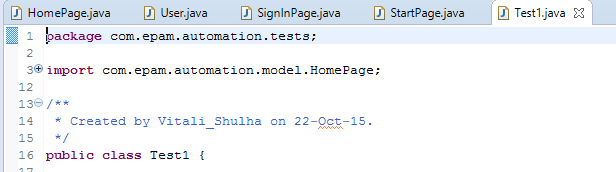


* Need not be a violation, but the name of the class here, ‘User’, could have been more intention revealing here. The model java class may not exactly represent a user entity here, rather the credentials of a user. May be UserCredentials is more meaningful name for the class here. [User may have a broader responsibility; UserCredentials has single and meaningful responsibility]

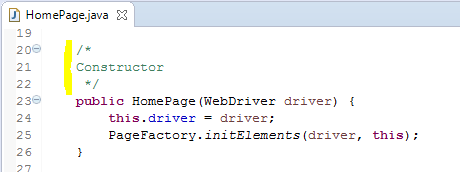
**Comments:**



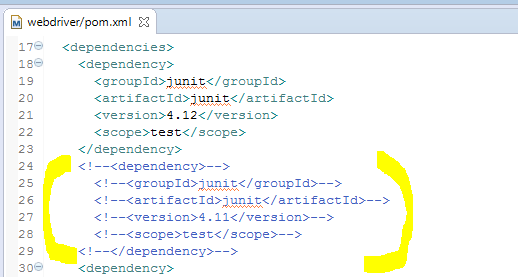


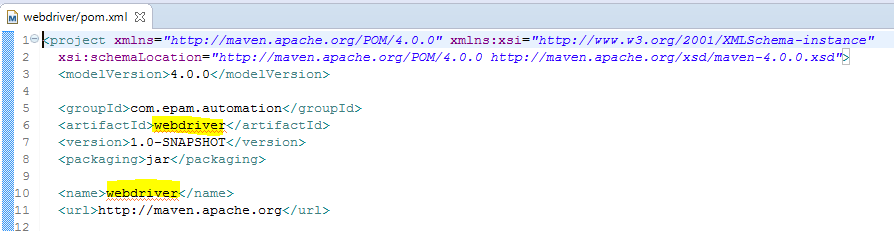
* No need of the comments with information about modifications made and dates; these are generally tracked in version control systems.
* While VCS also tracks the author/modifier of the class, author name can sometimes be allowed as part of auto generated code, however, with Java, it is more cleaner to use @author javadocs annotation.
* Also, one can provide short information on what this class represents if the class name doesn’t reveal the intended responsibility. In this case, the name ‘HomePage’ is meaningful and one need not write an explicit information that this class represents or models a home page.



* Redundant comment. No need of writing a comment for a constructor which is self-explanatory; one can understand from the syntax and semantics of this code



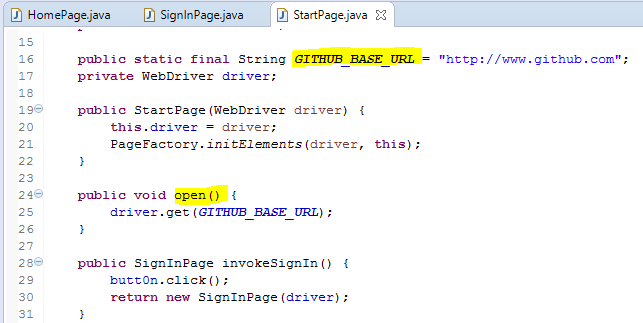
* Remove commented out code. Commented out code reduces the readability of the code and increases the lines of code for no good reason. Old code snippets can always be retrieved from the version control system.

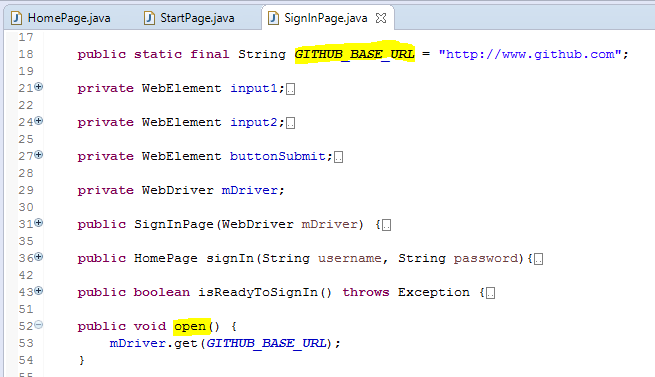
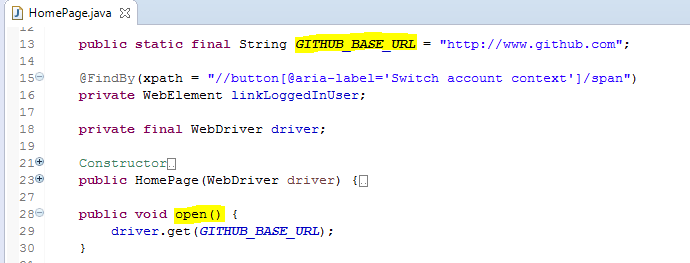


* Project name/artifactId can be more meaningful. ‘webdriver’ could be a misleading name, where as ‘webdriver-demo’ or ‘github-tests’ or ‘clean-code-demo’ might have been more meaningful.

**Code:**

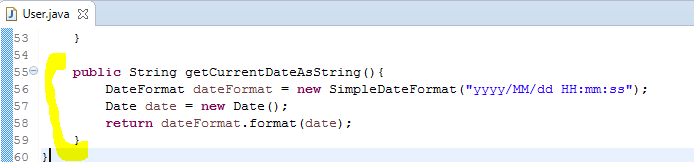
**Repeated code:**



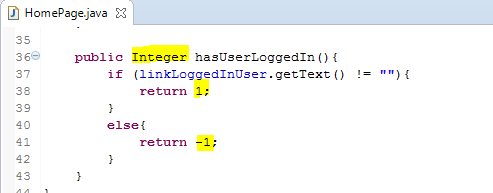
 

* DRY violation; The base URL and the open() method is only used for the first time through the StartPage instance. Hence, it is unnecessarily repeated in the other two classes.
* Just in case, if there were a functional need to use the base URL and open() from any Page, then it is cleaner to move this code to a base class and let all the page object classes extend the base class.

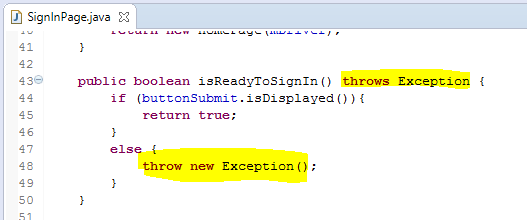
**SRP:**



* SRP violation. The method getCurrentDateAsString has no relevance in the User class. This method can be extracted out in a Utility class as static method.
* For instance, DateUtil. getCurrentDateAsString();



* Avoid Error Codes. Instead, use symbolic string constants with meaningful names. Use switch case statement to hop on the error caused in the handling code.
* For more readability and ease in maintenance, enums can be used.
* In this particular case, as the method returns only two possible types, return type can be boolean rather than Integer.
  + If it were to return boolean type, true or false, it can directly be used in condition expressions in the calling method. More readable.
  + However, if there were more than two possibilities to return, like 1, 0, or -1, then symbolic constants can be used.

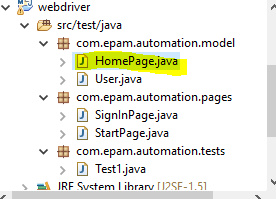


* Avoid throwing checked exceptions in the code. Throw unchecked exceptions instead, for a cleaner and maintainable code. Create a custom exception, say ‘LoginException’ by extending RuntimeException (hence, making it an unchecked exception) and define constructors for passing appropriate message.

For instance, throw new LoginException(“Sign In button is not present…”);

* With LoginException being defined as an unchecked exception, this method’s code and any other code that uses this method directly or indirectly need not change as per any change in this code in future.

**Others:**



* HomePage.java should be been placed in com.epam.automation.pages package.