Code Review Checklist

Biswajit Sundara | 25-Dec-2020

Code Formatting

1. Code should be easy to understand
2. Refactor lengthy codes to few lines
3. Check the code is properly formatted. (format in IDE)
4. Code lines shouldn't be too big. (avoid scrolling)
5. Ensure the alignment is correct
6. No more white spaces, unwanted spaces
7. Easy to figure out from where the code starts and ends
8. No empty lines
9. No commented out lines
10. Remove warnings
11. Remove unused imports

Naming Convention

1. Ensure naming convention is correctly followed (PascalCase, camelCase)
2. Method & variable names should be in camelCase (e.g name, employeeName, getData)
3. Class name should be in PascalCase (e.g Student,EmployeeDetails)
4. Constant names should in capital letters (e.g ORGNAME)
5. Give meaningful names to everything in the code (avoid generic names like array, i, j etc)
6. Avoid names like fake, dummy, random etc.
7. Distinguishable names, location for one location but if it's an array then locations

Comments

1. Follow 'no comments' rule. Develop the code that should be understandable without comments
2. Add comments only when its absolutely necessary. (usually for library or generic methods)
3. Do not write comments about the code but the operation you are doing
4. Good Comment: Add two numbers
5. Bad Comment: It takes two parameters and using the plus operator does the addition
6. Bad Comment:/*set the value of the age integer to 32*/int age = 32;
7. Specify about any workaround and temporary fixes.
8. Highlight warnings /*Don't change the return type*/

Framework Rules

1. Adhere to the framework rules and design
2. If methodname() { is used in your framework/project then follow it instead of putting the { next line.
3. Use framework reusable method, if the framework has click() method use it instead of driver.click() method.
4. Follow the framework folder structure, don't put things here and there. put files, codes in the designated folders
5. If you need to use custom code or enhance the framework, discuss and do it
6. In some project folks prefix I before interface e.g IDataProvider. Follow the same approach to remain consistent.
7. The framework should have logging mechanism that records the flow if enabled and helps in debugging.
8. The framework features should be changeable, we should on and off the features as per our need.

Managing Data/Files

1. Never hard code data in the code
2. Always mantian external files/separate class files
3. For example mantain test data in excel, json files
4. Application or framework properties in .properties files
5. Mantain the configurable properties in XML or properties file
6. Create class file for project constants etc.
7. Group similar values under an enumeration (enum).
8. Avoid opening and closing files too many times.
9. Any file opened should be closed.
10. Put the files in designated folder
11. Use a data type that best suits the needs.
12. Such as StringBuilder, generic collection classes for good performance.

Coding Best Practices

1. Methods/class files shouldn't be very huge.
2. Always break into logical groups that is easy to mantain/debug
3. Codes should be generic/reusable/extendable
4. Avoid multiple if/else blocks.
5. Avoid multiple, nested loops
6. Avoid recursive functions
7. Remove variables those are declared but not used.
8. Never do synchronization on Boolean (could lead to deadlock)
9. Remove dead code. Code is written but never called
10. Remove unreachable code. having code after the return statement
11. Break the loops once operation is done.
12. Never use Thread.sleep. Use Explicit wait.
13. static variables shouldn't be accessed/updated through objects
14. Handle exceptions and cleanup (dispose) resources.

Code Design

1. Direct use of implementations instead of interfaces
2. Use DRY (Do not Repeat Yourself) principle
3. Refactor duplicate/repeated codes
4. Follow Single Responsibility Design
5. Means one class, method, code block should do only one thing.

Web Development Coding

1. Use em instead of px
2. Use consistent styles for all similar elements
3. Use Responsive Web Design by implementing the code for all type of screens
4. Never hard code the height and widths, calculate it based on min, max-width
5. The application/style should work in all the browsers