**97 Things Every Programmer Should Know**

**Chapter 4: Automate Your Coding Standard**

**What are Three Things I learned today**

1. Before – **I am not familiar with the coding standard.**

After – **Coding standard is the set format for uniformity.**

- Setting a standard in terms of coding helps the developer to make the document clean and easy to understand. I also agree with it, clean code format that is readable can make things go smooth and easy to work with.

2. Before – **I have no idea about the automated coding standard**

After – **Automated coding standard makes the documentation clean.**

- Automated coding standard makes it easy to work in a project. Working in an automated way can make your work runs smoothly and can easily debug the program whenever there is a problem encountered.

3. Before – **I am not familiar with the ways to consider in producing code**

**quality reports.**

After – **Things to consider in producing quality of code reports are the ff:**

- Make sure code formatting is part of the build process, so that everybody runs it automatically every time they compile the code.

- Use static code analysis tools to scan the code for unwanted anti-patterns. If any are found, break the build.

- Learn to configure those tools so that you can scan for your own, project-specific anti-patterns.

- Do not only measure test coverage, but automatically check the results too. Again, break the build if test coverage is too low.

And with these, humans are not perfect and there is no perfect assurance that they will follow those, but if there is one that will take initiative to comply all the rules, it will produce better outcomes.

**Chapter 5: Beauty is in Simplicity**

**What are Three Things I learned today**

1. Before – **I am particular in format, consuming more time in perfection.**

After – **“Beauty of style and harmony and grace and good rhythm depends on simplicity.” - Plato**

- As developers we aim to produce a quality of code that has the following: Readability, Maintainability, Speed of development, and the elusive quality of beauty and to obtain all that is every programmer should document the code in simplest way.

2. Before – **I have no idea how to beautify my code.**

After – **Relying in simplicity in terms of coding is beautiful.**

- When you simplify your way of coding it will make your document easy to understand and make your work done in the set time.

3. Before – **I have no idea how to simplify my code.**

After – **Considering the qualities of producing beautiful code is to simplify it.**

* Simplifying your code can keep our systems maintainable over time. It can also easily allocate where the bugs came from and it is more convenient to fix.

**Chapter 6: Before you Refactor**

**What are Three Things I learned today**

1. Before – **I do not know the things to consider before refactoring.**

After – **Consider an approach first before you try to refactor.**

- An approach for restructuring begins with taking stock of the existing codebase and tests written against that code. In this case, it will help you understand its strengths and weaknesses.

2. Before – **I think it’s okay to refactor the code without prior considerations.**

After – **Before refactoring we must diligently think first.**

- If you tempt to rewrite everything, instead of fixing it, it may produce another error. I have experienced it a lot of times in my academe while working on a project. There was one time that I was tasked by one of my team to add new features, but then I forgot to focus on my task, I was hooked to adjust what I saw and tried to change it. After the scene, one of my members confronted me if I did some changes, then the program has produced complications.

3. Before – **I stick to my personal preference in coding.**

After - **When you code personal preferences and ego should not get in the way.**

- Personal preference is not a valid reason for restructuring. If you only rely to your preference, not all time your structure will work for everyone. One must consider for everyone not only for the self.

**Chapter 7: Beware the Share**

**What are Three Things I learned today**

1. Before – **I sometimes do not think about the context when I look out for codes that is useful for my task.**

After – **I should think about the context and understand it first.**

- Thinking about the context in terms of coding will help you produce your project in fastest way and can reduce bugs in the future.

2. Before – **It’s okay to copy paste the code without thinking the whole context of it.**

After – **Context is critical.**

- If you do not think about its context and understand it, it may lead to some errors. I had also an experience with this while working on a project in the academe. I searched source codes that is relevant to the problem without understanding its context, then it does not change everything; the error still exists.

3. Before **– I do not have any idea about considering the context first before using the shared code.**

After – **Some of the shared code have different context and were not dependent to a particular factor.**

- If the code were not dependent to each other or it does not align to its functionality it may produce errors. It was proven and tested while I was working in a project in my academe. I was not aware that there are some parts of the code that is not relevant to my code were included, then it affected the program that I created. It birthed to another error.

**Chapter 8: The Boy Scout Rule**

**What are Three Things I learned today**

1. Before – **I never thought about the boy scout rule.**

After – **Boy scout rule must be followed every time.**

- As programmers it is also our job to clean the mess regardless of who made the mess.

2. Before – **I only think about the task then after doing it, I leave it.**

After – **Applying the rule can help you organize and locate the document without having a hard time.**

- The rule should always be applied in all ways to make jobs easier.

3. Before – **I am too dependent on the other to clean my mess.**

After - **Teams help each other, and clean up after each other.**

- If collaboration is present in the project, and the rule is applied it will produce better outcomes.