**97 things every programmer should know**

**Chapter 4 Automate your Coding Standard**

**▼ What are Three Things I learned today**

1. **Make sure code formatting is part of the build process**

Before – So that no one can own this kind of code, because every developers has a code formatting for them to figure out who’s copying others code.

After – So that it can run automatically everytime they run the code.

1. **Use static Analysis tools to scan the code for unwanted anti-patterns**

Before –Using this is very important it is because, as a developer we need to make sure that our system or project is secure.

After – Using static analysis tools is to help you detect code quality and security issues early in your software's development lifecycle

**The coding standard should be dynamic rather than static**

Before – It is because the project or the system that you make is need to make a change. After - As the project evolves, the needs of the project change, and what may have seemed smart in the beginning, isn't necessarily smart a few months later. Not all the time your system is good by the time goes by, your system needs improvement that why if you code something it is good if you make it a dymanic.

**Chapter 5 Beauty is in Simplicity**

**▼ What are Three Things I learned today**

1. **What is beautiful code?**

Before – For me beautiful code is a simple code that is easy to understand.

After - Beautiful is to make it simple.

1. **Readability**

Before – It is really important that your code is readable.

After – Make sure if you code make sure that is easy and readable. Because if your code is not readable it make hard if some people to understand your code.

1. **The bottom line is that beautiful code is simple code**

Before – As what I understand this sentence is that keeping your code simple is beautiful because, it is easy to understand.

After – Keeping your code simple is easy to understand and easy to trace if you have encountered some issues. You can appreciate beauty if it is simple. Because simplicity is beauty.

**Chapter 6 Before you Refactor**

**▼ What are Three Things I learned today**

1. **Avoid the Temptation to rewrite everything**

Before – Avoid making a new code if that old code is still useful you should reuse it in order for you to avoid another bugs.

After – Reuse your old code as much as possible because that code is already been tested so it is for you to make life easier and less hassle because you don’t have to face new bugs. And this will waste a lot of time, effort, and knowledge gained over the years.

1. **Personal preferences and ego shouldn’t get in the way**

Before – Ego should not involve if you’re working in a company it because you can compare yourself to others work is not good either.

After –Don’t restructuring other’s code if it is not broken just because you feel that you can do better.

1. **Remember that humans make mistakes**

Before – Yes, because we humans are not perfect even when a professional person can make mistakes.

After - It's not always a given that restructuring will result in new code that is superior to or even on par with the old. Numerous failed restructuring initiatives have been made but we are still human we make some mistakes.

**Chapter 7 Beware the Share**

**▼ What are Three Things I learned today**

1. **Beware the share**

Before – You need to be aware of the shared code because we have a different logic.

After – Being aware of the shared is very important it is because if we copied some codes then analyse those code you have a different logic. Yes the code is running but in the long run it will meet an error.

1. **Check your context**

Before – Checking your context is the very first thing to do so that you can build your own.

After – Checking it is for you to be aware what your context are in, and it for your own good because if you copy others code you need to check the context first before you will it.

1. **Dependencies is critical**

Before – Being dependent of others work is critical because we don’t know the real logic of that code.

After – So in this dependencies of beware of share is critical that is because if you copied code from others. Yes it look like the same but it has a different logic and in the long run you might encounter a larger concerns of the system.

**Chapter 8 The Boy Scout Rule**

**▼ What are Three Things I learned today**

1. **Caring for our own code is one thing**

Before – It is easy for you to taking good care for your own code that is not a big responsibility because that is your own.

After – Making sure that you care for your own code is a good practice of taking care of everything. Because before you care for anything else you make sure that you care for yourself or for your code. Yours is the first step to care others.

1. **Caring for the team's code is quite another**

Before – Yes, because a lot of you has the responsibility of taking care your team’s code.

After – Your teams are making sure that your care for your code. Just to make no one will copyright our works or codes.

**3. Teams help each other, and clean up after each other.**

Before – Yes, in this stage your teammate needs to cooperate in order for your group to reach your teammate’s goals.

After – You need each other’s back, you help each other this is what you call team work. For example, we the Team Synergy make a mobile-based simulation game so without our team works we can’t build this mobile based called EcoSave. If you have a team work in your group mates you can accomplished things that is impossible for you to make.