**97 Things Every Programmer Should Know**

**Chapter 9: Check Your Code First before Looking to Blame Others**

**What are Three Things I learned today**

1. Before – **I sometimes blame the compiler for unwanted errors.**

After – **Compiler was never really the problem.**

* Oftentimes when we experienced problems instead of focusing on the possible solution, we tend to put much time and effort to seek why does it happen, what was the reason, and looking for something or someone to blame. Thus, we tend to waste time, but the truth is it is us is the problem.

2. Before – **I believe that compiler bug is the root of being unable to finish the project on time.**

After – **I realized that the way I write the code is really the problem.**

- Considering a clean and concise documentation helps me solve bugs easily. Instead of putting more energy on blaming the compiler, I will now fix my eyes to my code.

3. Before – **I put much effort on thinking of a perfect design.**

After – **Simplicity of design is paramount.**

* Making the code simple and easy to understand will help the programmer track and solve bugs easily. Unorganized code will produce complications that will affect the progress of the specific project.

**Chapter 10: Choose Your Tools with Care**

**What are Three Things I learned today**

1. Before – **I thought that all applications were made from scratch.**

After – **Modern applications are rarely built from scratch.**

* Applications are assembled using existing tools — components, libraries, and frameworks for the reasons of the factors that will help programmers develop an efficient system.

2. Before – **Choosing the right mix of tools is easy.**

After – **Choosing the right mix of tools is tricky.**

* I never thought that choosing appropriate tools is a tricky business that I need to face, I thought that all the tools in the requirements are available and sufficient.

3. Before – **I have no idea what are the appropriate tools for a project**

After – **In creating an efficient project, one must consider tools that will help the programmers develop the project. Before choosing a tool, the following are the things to consider:**

* Different tools may rely on different assumptions about their context.
* Different tools have different lifecycles, and upgrading one of them may become an extremely difficult and time-consuming task.
* Some tools require quite a bit of configuration.
* Cost

**Chapter 11: Code in the Language of the Domain**

**What are Three Things I learned today**

1. Before – **I have no idea what is the relationship between the code and the language of the domain.**

After – **The relationship between the code and the language of the domain has something to do with organization.**

- In coding if you want to make its process go smooth and free from complications, might as well apply domain terms for it can help you identify a certain thing that is a part of the project easier.

2. Before – **I have no idea what is the essence of the domain terms.**

After – **Domain terms can help you easily identify and understand your work.**

- Using domain terms or concepts can help others easily understand the intent of the code.

3. Before – **I have a lesser understanding about the domain and its relevance to the code.**

After – **Domain is basically an organization that specifies a group that will help programmers understand the intent of the code.**

- With the help of the domain, the flow of the project will be much easier to work with.

**Chapter 12: Code is Design**

What are Three Things I learned today

1. Before – **I never thought that code is a design too. I thought that code is just a structure of functions to make the system work.**

After – **Code is like a blueprint that undergoes procedure to set it well.**

* If you are too complacent about what you do for the sake of complying, it will surely affect your project's performance and status.

2. Before – **Once the timeline of the design is set, it is set to be done accordingly.**

After – **I realized that it does not work all the time.**

* Sometimes the design of what we thought is done is good to go, but it's not for it requires several tests to know its status and capability.

3. Before – **I thought that when you design something, and you are satisfied, it is done. I am wrong.**

After – **When you design you must dedicate yourself upon the mastery of your work, in that sense it will mirror the output.**

* Being passionate about something will greatly affect the output of what you are doing.

**Chapter 13: Code Layout Matters**

**What are Three Things I learned today**

1. Before – **I never considered a layout when coding.**

After – **To make a perfect documentation, one must set a layout.**

* A layout helps a document when it comes to organizing. In terms of coding, a layout will help the programmer set their code easy to understand.

**2. Before – I really do not mind of how or what my code looks like as long as it runs, then I am good.**

**After – I realized that I should take layout and organization into consideration because that's what make a code more readable and easy to understand.**

* When the code is readable and easy to understand by the other programmers, it will make the task run smoothly and be done fast.

3. Before – **I have no idea how to layout the code.**

After – **To attain a perfect layout, one must consider the following:**

* Easy to scan: Organizing it accordingly to easily locate and compile what are needed.
* Expressive layout: Every line must reflect the intention of the code not just the syntax of the used language.
* Compact format: It will the programmer trace the code without having complications with the help of the layout.