**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 often times blame the compiler when I started to run out of ideas on how to solve the bug or problem.
2. **After**- Compiler bug or error is very rare and in order to pin down the correct reason for the error, a thorough search must be conducted.
3. **Before**- Patience is a virtue especially in debugging a certain module.
4. **After**- Don’t waste time on hypothetical guesses, put more effort in extensive debugging and testing.
5. **Before**- I don’t trust any debugging tools when I developed my own code. I really on my own instincts and faith.
6. **After**- I should use different debugging tools to have an ease of pinpointing the real cause of the problem.

**97 THINGS EVERY PROGRAMMER SHOULD KNOW- CHAPTER 10: CHECK YOUR TOOLS WITH CARE**

**WHAT ARE THREE THINGS I LEARNED TODAY**

1. **Before**- I only use tools that are familiar to me and doesn’t want to explore other any other tools
2. **After**- I will try out many tools that is applicable to the system that I will develop.
3. **Before**- I intend to create everything from the scratch meaning that I developed using the native version of different languages.
4. **After**- I started using frameworks to save time and to have a more reliable code because frameworks are proven and tested especially the mainstream ones.
5. **Before**- I mix and match different tools without inspecting them properly and the whole code ends up being messy and other tools are being conflicted to one another.
6. **After**- I read their documentation thoroughly and decide whether it is applicable to the code that I am developing and if it’s futureproof.

**97 THINGS EVERY PROGRAMMER SHOULD KNOW- CHAPTER 11: CODE IN THE LANGUAGE OF THE DOMAIN**

**WHAT ARE THREE THINGS I LEARNED TODAY**

1. **Before**- I code in a very complex structure which is not readable by other programmer or developer.
2. **After**- I will code in a way that the next programmer will thank me.
3. **Before**- I don’t care how my code looks in the perspective of another programmer
4. **After**- I will code in a much understandable manner where the next programmer can gather the intent of the code much more easily.
5. **Before**- I don’t give any importance on different codebase.
6. **After**- Having a clean codebase is very important and essential in development because it can affect the reusability , readability and etc. of the code.

**97 THINGS EVERY PROGRAMMER SHOULD KNOW- CHAPTER 12: CODE IS DESIGN**

**WHAT ARE THREE THINGS I LEARNED TODAY**

1. **Before**- The quicker it is done the better
2. **After**- In coding, it is a good practice to have an efficient and bug-free code. Development takes time and sometimes the pressure on the deadline makes the code less efficient and incomplete.
3. **Before**- Becoming complacent on the quality of the code because of the thinking that “it can be fix later”.
4. **After**- We should eradicate the pressure of using incomplete code for the sake of faster development.
5. **Before**- The code quality doesn’t matter because the client won’t even inspect the source code.
6. **After**- We should ensure the quality of our code to avoid design crisis.

**97 THINGS EVERY PROGRAMMER SHOULD KNOW- CHAPTER 13: CODE LAYOUT MATTERS**

**WHAT ARE THREE THINGS I LEARNED TODAY**

1. **Before**-
2. **After**-
3. **Before**-
4. **After**-
5. **Before**-
6. **After**-