**97 Things Every Programmer Should Know**

**Chapter 1: Act with Prudence**

**What are Three Things I learned today**

1. Before – Doing it right is better than doing it quick.

After - "Doing it quick is often appealing than doing it right"

- Working on a project holds different accounts that are needed to be managed well. If one fails it might affect the other. Planning a concrete time table makes it more efficient to accomplish the assignment for it holds pressure to finish the task rather than rest for a bit and will fix it later. It is good but it also carries other effect such as another unexpected bug.

2. Before – It is okay to leave bugs and fix it later.

After - "Technical Debt is NOT your FRIEND"

- "Technical debt is like a loan" I agree on that statement for it will benefit you in a bit, but it will cost you more efforts to pay. In the field of programming, creating projects has a structure that needs to be follow. There are teams assigned to it, and they must consider their responsibility in making the project possible.

In terms of coding, tracking the code is never an easy task for its process is like a chain reaction, one move and it will affect the other. It is not advisable to be too complacent about the task for it will create future complications.

3. Before – Rest first before fixing the bug.

After - "Payoff technical debt as soon as possible"

- If you will not solve the unresolved problems right away it will affect you, your team, and your project.

**Chapter 2: Apply Functional Programming Principles**

**What are Three Things I learned today**

1. Before – It is okay to not to master any kinds of programming.

After - Mastery of Functional Programming Paradigm Improves the Quality of Code

- In this age, many system infrastructures are requiring developers to adapt the modern way of coding. As a newbie learning another set of language or syntax is quite difficult at first but I believe that with enough time and training it will surely develop. Building a quality of code undergoes strict evaluation to attain

the set goals.

2. Before – I have no idea what is visibility semantics

After - Visibility semantics is helpful.

- Visibility semantics refers to the visibility and scope of variables. It is crucial for it can help reduce and identify insidious defects.

3. Before – I have no idea what is referential transparency

After - Object systems will resonate with referential transparency goodness and closer to their functional counterparts.

- Building a good system requires a good and efficient program that will reflect its set goals. Referential transparency refers to an expression wherein it can be replaced by its resulting value without affecting the program's behavior. Object systems must acquire referential transparency to make the system run smoothly.

**Chapter 3: Ask What Would the User Do**

**What are Three Things I learned today**

1. Before – I thought users also think like programmers

After - Users don't think like programmers

- As programmers it is our right to know the clients' needs. Programmers duty is to implement and create the clients' vision of a specific project. Some users don’t have a background about how we work on the project. It is a jargon for them, and our job is to make them understand how the project will work.

2. Before – I don’t know how users think

After - The best way to find out how users think is to watch one

- Empathize is one of the best strategies to win a customer.

3. Before – I have no idea how to capture requirements easily.

After - Watching users is the best way to capture requirements.

- Gathering requirements is not an easy job for you will consider different things related to the project to attain the goal, collaborating and empathizing with the users is a great help to fulfill it.