“97 Things Every Programmer Should Know”

Chapter 71-80

Chapter 71: Reinvent the Wheel Often:

Trying new ways to solve problems is important because it helps us learn and discover better solutions. It's like experimenting with different study methods to find what works best for us. By being open to new ideas and approaches, we can foster creativity and innovation in problem-solving, leading to more effective solutions and personal growth.

Chapter 72: Resist the Temptation of the Singleton Pattern:

Using the same solution for every problem might limit our options and creativity. It's like always using the same tool, even if there's a better one available. Instead, being flexible and open to different approaches can lead to more effective solutions tailored to each specific problem, ultimately improving our problem-solving skills and the quality of our work.

Chapter 73: The Road to Performance Is Littered with Dirty Code Bombs:

Writing messy code can slow down our progress and make future tasks harder. It's like leaving obstacles in our path when we're trying to reach a destination quickly. Therefore, keeping our code clean and organized is crucial for working efficiently and avoiding problems down the road. By prioritizing code cleanliness and adhering to best practices, we can streamline our development process and improve the performance and maintainability of our software.

Chapter 74: Simplicity Comes from Reduction:

Keeping things simple by focusing on what's important makes it easier to understand and work with them. It's like cleaning up clutter in our room and only keeping what we really need. Simplifying our code helps us avoid confusion and find solutions more easily, ultimately leading to cleaner, more maintainable codebases and more efficient development processes.

Chapter 75: The Single Responsibility Principle:

Giving each part of our code one job to do makes it easier to manage and understand. It's like assigning specific tasks to different group members for a project. By following this principle, we can create cleaner, more organized code that is easier to maintain, debug, and extend, ultimately improving the overall quality and reliability of our software systems.

Chapter 76: Start from Yes:

Being open to new ideas and opportunities can lead to exciting discoveries and personal growth. It's like saying "yes" to trying a new hobby or activity—we might find something we really enjoy! Embracing new experiences with a positive attitude can lead to positive outcomes and new possibilities, ultimately enriching our lives and broadening our horizons.

Chapter 77: Step Back and Automate, Automate, Automate:

Letting technology handle repetitive tasks for us saves time and effort. It's like using a robot to do chores around the house—it frees us up to do more enjoyable things. Automation helps us work more efficiently and focus on tasks that require our creativity and attention, ultimately improving our productivity and enhancing the quality of our work.

Chapter 78: Take Advantage of Code Analysis Tools:

Using tools that help us find and fix mistakes in our code improves its quality and reliability. It's like having a tutor who checks our homework and helps us improve our skills. Embracing code analysis tools helps us write better code and become better programmers, ultimately leading to more robust and maintainable software systems.

Chapter 79 : Testing Is the Engineering Rigor of Software Development:

Testing software is like making sure a bridge is strong enough to hold cars. Just like engineers test bridges to ensure they're safe, developers test software to make sure it works correctly. Testing helps find and fix problems before they cause accidents or errors in the software.

Chapter 80: Thinking in States

Thinking about software in terms of different states is like understanding the different moods or situations people can be in. Just like people can be happy, sad, or angry, software can be in different states based on what's happening. By understanding and managing these states, developers can create more reliable and adaptable software.