# **97 Things Every Programmer Should Know**

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

1. **Before -** Once I build the application or the application system and there are some issues arises, I usually blame my code that it is not working.

**After -** The fault is likely within your program code, and therefore, the responsibility lies with you as the author.

1. **Before** - It is rare for a compiler to have bugs.

**After** - I do agree that it is rare to have bugs with the compiler, unless it is in its first version or they confirm the bug with a statement.

1. **Before** - As issues arises, I usually check the line of code or file that is not working, and fixed it directly. Blaming the compiler doesn’t come to my mind if this situation happened.

**After** - If you believe your code is written correctly but it isn't working on your current compiler, try it on another compiler or device. If it works on another compiler or device, there's something wrong with your current compiler. However, if it still doesn't work, then the issue is with your code.

## **Chapter 10: Choose your Tools with Care**

1. **Before -** Just like buying a phone, we look first to its specifications, features, and costs.

**After -** When choosing programming tools, we need to ensure that the tool is appropriate for the project system and within budget.

1. **Before** - I only use the necessary tools (free version) for a project system, as having excessive or inappropriate tools in a project can slow down my device due to limited storage capacity.

**After** - Using only the necessary tools for your project helps you optimize your workflow and saves you time, resources and money. It is important to choose the tools that fit the needs and requirements of the project, and avoid using tools that are not needed, as they can slow down the process and lead to unnecessary expenses.

1. **Before** - Choosing the right mix of tools is quite easy to decide.

**After** - Choosing the right mix of tools is a tricky business, as you need to consider various factors such as available documentation, configuration, cost, and efficiency for your project.

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

1. **Before -** When making a name, make sure that it has intent.

**After -** Using domain terms will greatly build the intent of a variable, method, function or class. This leads to code that is more readable and understandable.

1. **Before** - I don’t usually use domain terms. As long as I understand the terms that I am using, I’m okay with it.

**After** - I do agree that if you don’t code using domain terms, the code will be less understandable, making it more difficult for others to follow, maintain, modify, and creating a technical debt. This can result in confusion for other programmers who might need to review your code in the future.

1. **Before** - What I understand with domain terms, is that a specific term for that certain project. For example in an e-commerce program, instead we use “customers”, we say “shopper”.s

**After** - It’s good to know that making domain concepts or using terms that is are related to the subject matter of the project will help build the intent of the code. This can improve code maintenance and reduce confusion among other programmers who may need to work with the code

## **Chapter 12: Code is Design**

1. **Before -** With the advancement of the technology that we have today, there are a lot of resources that are available for everyone.

**After -** However, this advancement has also led to the automation of many tasks, which may not require human intervention as the projects can be completed without any cost.

1. **Before** - Distributing the project as soon as possible the better.

**After** - Having a well-designed and completed project quickly can give an advantage in the market by providing the solution to the market needs before the competitors.

1. **Before** - Coding is a technical process.

**After** - It is a way to bring imagination and creativity to life, similar to other forms of art such as literature or poetry. This means that coding has both technical and artistic aspects, making it a unique and dynamic field.

## **Chapter 13: Code Layout Matters**

1. **Before -** When making a project, I make sure that there is consistency with my layout, as it irritates my eyes and challenging to navigate the code.

**After -** I agree that code layout matters. Having well-laid-out code can improve readability and make it easier to track and understand.

1. **Before** - Code layout should be consistent and is uniform with other layouts.

**After** - Having a consistent layout in your code will make it easier to read and navigate through the program. This can improve the overall efficiency of the project, as you will be able to quickly locate the specific sections of code that you need to work on, without having to waste time searching through the code. Additionally, a uniform pattern will also help to minimize errors and bugs in the program, as it will be easier to identify any inconsistencies or unexpected results when the code is well-structured and easy to follow.

1. **Before** - Coding is simply just coding.

**After** - Coding is a creative and expressive art form, similar to literature or poetry. A well-structured, concise, and understandable code will help us grasp the intended ideas more easily.