By Rakesh Shekhawat
August 24, 2018

Clean Code Principles: Be a Better Programmer

"My code is working well, the website I built is looking great, and my client is happy. So why would I still care about writing clean code?"

If this sounds like you, then read on.

A little while ago, I was having a discussion with one of my friends, Kabir. Kabir is an experienced programmer. He was working on a complex project, and he was discussing a problem with me. When I asked to see the code for that problem, he said, sounding proud, "I built this project so we are the only ones who can understand the code."

I was pretty horrified. I asked him if he deliberately wrote dirty code.

"The client didn't give me enough time," my friend told me. "He is always in a hurry and pushing for deliveries, so I did not have time to think about cleaning it up."

This is almost always the excuse I hear when I ask about dirty code.

Some programmers write dirty code because they plan to release the first working version and then work to make it clean. But it does not work; no client gives you time to clean code. Once the first version is released, they will push you for the second. So, make it a habit to write code as clean as you can from the first line of code.

I've always learned that using clean code principles has many benefits down the line, and this post will show you why.

It is the job of the project manager, sales head, or client to get the project done in minimum time so they can control the cost of the project. But producing quality, clean code is your duty as the programmer.

Writing clean code is not a big or time-consuming task, but making it your routine, and committing to it, will go a long way toward advancing your career and improving your own time management.

Clean code always looks like it was written by someone who cares.

> *"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."—Martin Fowler*

You've probably read this far for two reasons: First, you are a programmer. Second, you want to be a better programmer. Good. We need better programmers.

Keep reading to learn why clean code matters, and you'll become a better programmer.

Why Should We Strive for Clean Code?

Clean code is readable and easy to understand by everyone whether the reader is the author of the code or a new programmer.

Writing clean code is a necessary mindset. It takes practice to write clean and structured code, and you will learn to do it over time. But you need to start with the mindset of writing this way. And you'll get used to reviewing and revising your code so it's the cleanest it can be.

No one is perfect, and so you are not either. You always will find some opportunity to improve or refactor the code when you come back to review your code after a few days or weeks. So, start writing the code as clean as you can from the first line of code so later you can work more on performance and logic improvement.

Benefits of Clean Code

“Why should I care about writing clean code?” you may still be asking yourself.

There are many reasons to get into the clean code mindset I described above. Some of the most important reasons are:

Better Use of Your Time

The first beneficiary of clean code is the programmer themselves. If you are working on a project for months, it’s easy to forget things you did in the code, especially when your client comes back with changes. Clean lines of code make it easier to make changes.

Easier Onboarding for New Team Members

Using clean code principles helps to get a new programmer onboard. There is no need for documentation to understand the code; the new programmer can directly jump into it. This also saves time for both training the new programmer as well as the time it takes for the new programmer to adjust to the project.

Easier Debugging

Whether you write dirty or clean code, bugs are inevitable. But clean code will help you to debug faster, regardless of how much experience or expertise you have.

And it’s not uncommon for your colleagues or managers to help you solve the problem. If you’ve written clean code, no problem: They can jump in and help you out. But if your manager has to work through your dirty code, well, you might end up like my friend Kabir.

More Efficient Maintenance

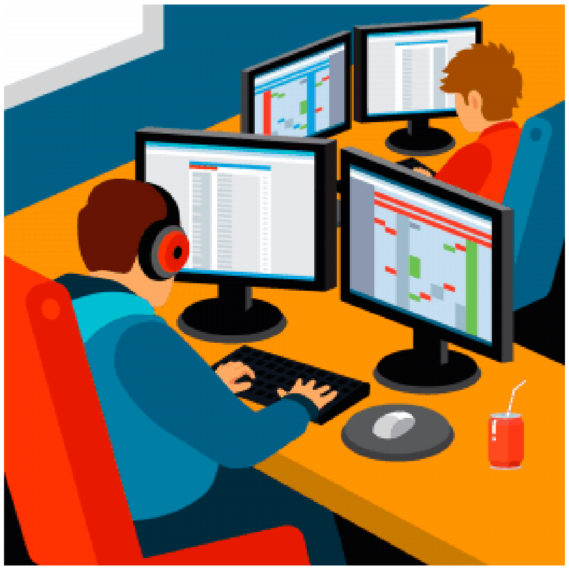
> *“Of course bad code can be cleaned up. But it’s very expensive.”*
> *—Robert C. Martin*

Maintenance does not refer to bug fixing. As any project grows, it will need new features, or changes to existing features.

Do you know that the major cost of any software project is in maintenance? The company will always release the first version, or minimum viable product (MVP), as early as possible. Additional or new features are always an afterthought as the software gets more use. Clean code makes maintenance relatively fast and easy.

These first three points explain how clean code can save a programmer's time. And, saving a little time every day will have a compound effect on the delivery time and cost of the software. That's good for your company.

You'll Feel Good



Does it help you feel confident to share your work with others, too? Or with your client?
If you're writing quality, clean code, you should feel super confident. You should not have a fear of breakdown; you can fix defects faster.

And that means you're also probably enjoying the programming.

Now, how do you write clean code?

How To Write Clean Code

> *"You should name a variable using the same care with which you name a first-born child."*
> *—Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship*

A programmer is an author, but they might make the mistake in identifying the audience. The audience of a programmer is other programmers, not computers. If computers were the audience, then you might be writing the code in machine language.

So, to make it easy to understand for your audience, you should use meaningful nomenclature for variables, functions, and classes. And make it more readable by using indentation, short method, and short statement, where appropriate:

- **Use easily pronounceable names for variables and methods.** Do not use abbreviations in the variable and method names. Use the variable name in full form so it can be easily pronounced and everyone can understand it.

Dirty code examples	Clean code examples

Dirty code examples	Clean code examples

public $notiSms;
public $addCmt;
	public $notifySms
	public $addComment;

foreach ($people as $x) {
echo $x->name;
}
	foreach ($people as $person)
	{
	echo $person->name;
	}

$user->createUser(); createUser method does not make sense as it is written in user class.	$user->create(); Remove redundancy

- **Use the name to show intention.** The purpose of the variable should be understandable to someone reading the name of the variable. Write the name as you would speak it.

Dirty code examples	Clean code examples

protected $d; // elapsed time in days	protected $elapsedTimeInDays;
	protected $daysSinceCreation;
	protected $daysSinceModification;
	protected $fileAgeInDays;

if ('paid' === $application->status) {
//process paid application
}
	if ($application->isPaid()) {
	//process paid application
	}

- **Don't be innovative; be simple.** Show the innovation in logic, not in naming variables or methods. Having a simple name makes it understandable for everyone.

Dirty code example	Clean code example
$order->letItGo();	$order->delete();

- **Be consistent.** Use one word for similar functions. Don't use "get" in one class and "fetch" in another.
- **Don't hesitate to use technical terms in names.** Go ahead, use the technical term. Your fellow programmer will understand it. For example, "jobQueue" is better than "jobs."

- **Use a verb as the first word in method and use a noun for class. Use camelCase for variable and function name. The class should start from the capital.**

Dirty code examples	Clean code examples
public function priceIncrement()	public function increasePrice()
Public $lengthValidateSubDomain	Public $validateLengthOfSubdomain;
class calculationIncentive	class Incentive

- **Use consistent naming conventions.** Always use uppercase, and separate words with underscores.

Dirty code example	Clean code example
define('APIKEY', '123456');	define('API_KEY', '123456');

- **Make functions apparent.** Keep a function as short as possible. My ideal length of a method is up to 15 lines. Sometimes it can go longer, but the code should be conceptually clean to understand.
- **Keep arguments to fewer than or equal to three.** (If arguments are greater than three, then you must think to refactor the function into a class.)

  You should also limit a function or method to a single task. (Avoid using "and" in a method name, like "validateAndSave." Instead, create two methods, one for validation and another for save).

  Indentation is also important. Your clean code must use four spaces for indents, not the tab key. If you're already in the habit of using the tab key, change your IDE setting to make the tab key denote four spaces as opposed to its usual five.

  If your method has more than three indentations, then it's time to refactor in new methods.

- **Behavior of class and object.** One class should do one thing. If it is for the user, then all methods must be written entirely for the user experience.
- **Don't comment on bad code.** If you have to add comments to explain your code, it means you need to refactor your code and create new methods. Comment only if it is legally required or if you need to keep notes on the program's future or history.

Dirty code example	Clean code example

// Check to see if the employee is eligible for full benefits
if ($employee->flags && self::HOURLY_FLAG &&
$employee->age > 65)
	if ($employee-
	>isEligibleForFullBenefits())

- **Use Git for version history.** Sometimes, features change and methods need to be rewritten. Usually, we comment out the old code for fear that clients will make a U-turn and request the older version. But if you use the Git version control system, it will keep all versions stored, so there's no need to keep dead code. Remove it and make your code clean.
- **Avoid working with a large array.** Avoid making an array for a large data set; instead, use a class. That make it more readable, not to mention that it creates an additional safety for your application.
- **Do not repeat the code.** Every time you write a method, ask yourself if something similar has already been built. Check the code library or other documentation.
- **Don't hardcode.** Define constant or use variables instead of hardcoding the values. Using the variable will not only make it readable but will also make it easy to change if it is being used at multiple places.

Dirty code example	Clean code example

if (7 == $today) {
return 'It is holiday';
}
	const SATURDAY = 7;
	if (self::SATURDAY == $today) {
	return 'It is holiday';
	}

- **Make the statement readable.** To make the statement readable, keep the line short so you don't need to scroll horizontally to read the complete line.

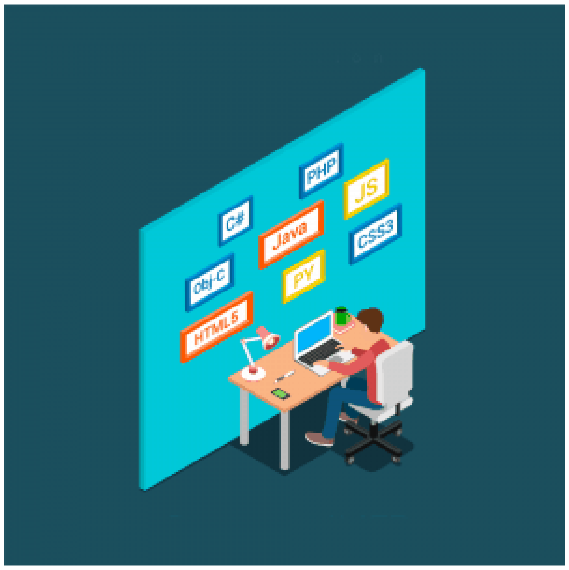
Some Other Tips for Cleaner Code

Review your code yourself. Review your code once in a while. I'm sure you'll find something new to improve on every time you revisit it.

Review your code with your colleagues. Review your colleagues' codes, and ask them to review yours. Don't hesitate to consider suggestions.

Every language has its own naming convention. If you are writing for PHP, use PSR-2's coding style guide.

To increase the quality of the code, you should use the TDD approach and write unit tests. Test-driven development makes code changes easy; you do not need to fear breakdown of the code. If you made any mistakes, the unit test will fail, and you will know what test case failed and what block of code was responsible for that.



Use the Git version control system to collaborate on development. Git becomes an essential tool when multiple programmers are working on a project. Code review becomes easy if you are using a version control system.

For future reading, check out *Clean Code*, by Robert C. Martin.

Moving Forward to Cleaner Code

Writing clean code has many benefits, and it's easy to see why.

With these tips, you can be well on your way to writing code that everyone can understand—and that will make life easier for you in the long run.

It will help your colleagues, your team, and your employer as well.

ABOUT THE AUTHOR

Rakesh Shekhawat

Rakesh Shekhawat has built websites and mobile applications for over 10 years and lives in Jaipur, India. His expertise is in helping startups build their tech products. Read more of Rakesh's writing and how to be a better programmer on Successfuler. You can also find him on Twitter @rsing2109.

Related Posts

The Value of Failure Is the Cost of Innovation

FEB 26, 2020 / BY DANILA PETROVA

A Successful Remote Career in Coding? More Likely Than One Might Think

FEB 10, 2020 / BY ANDRIANA MOSKOVSKA

Productivity for Programmers Through Time and Attention Management

FEB 07, 2020 / BY DANILA PETROVA

The Power of Lifestyle Changes for Programmers

FEB 05, 2020 / BY DANILA PETROVA

5 Signs You Should Become a Software Developer

FEB 03, 2020 / BY SLAVA VANIUKOV

From Software Engineer to Artificial Intelligence: Transform Your Programming Career

JAN 31, 2020 / BY MICHAEL IYAM

8 Disappointments of the Programmer: What You Should Be Prepared For

JAN 27, 2020 / BY CONNIE BENTON

How I Became THE MOST HATED MAN In Tech

NOV 20, 2019 / BY JOHN SONMEZ

0 Comments **Simple Programmer** **Disqus' Privacy Policy** 1 **Login**

Recommend 5 Tweet Share **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Do Not Sell My Data**

←Previous post Next post→