

Subscribe to DeepL Pro to edit this document.  
Visit [www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)for more information.

Origins

Most software systems are unnecessarily overloaded with almost unnecessary features, which impairs their usability by end-users and also makes it difficult for developers to support and develop them. Following the KISS principle allows the development of solutions that are easy to use and easy to maintain.

What is the KISS principle?

KISS is a design principle adopted by the US Navy in 1960. The KISS principle states that most systems work best if they remain simple rather than complicated. Therefore, in design, simplicity should be a key goal and unnecessary complexity should be avoided. There are many variants of the acronym: 'keep it simple, stupid'. These are 'Keep it Simple, Silly', 'keep it short and simple', 'keep it simple and straightforward' and 'keep it small and simple'.

The term was reportedly coined by Clarence Johnson, lead military engineer at Lockheed Skunk Works (makers of the Lockheed U-2, SR-71 Blackbird and many other aircraft).

While it has been popular to use the decryption "Keep it simple, stupid. This principle is best illustrated by the story when Johnson handed a team of aircraft engineers a set of tools, setting them a condition: a mid-level mechanic must be able to repair the jet they were designing in the field with just those tools. So "stupid" refers to the relationship between the fact that things break and the complexity of the repairs required to make them.

In programming, following the KISS principle can be described as follows

don't use complicated and non-obvious solutions, keep the code as simple as possible

There is no point in increasing the level of abstraction limitlessly, you have to be able to stop in time;

It is not a good idea to include a huge library if you only need a couple of functions from it;

decomposing something complex into simple components is the architecturally correct approach;

Absolute mathematical accuracy or extreme detail is not always needed - most systems are not built for launching space shuttles, data can and should be processed with the accuracy that is sufficient for a good solution, and detail should be provided in the amount the user needs, not in the maximum possible amount.

Often the most correct solution is the simplest implementation of the task, with nothing superfluous.

The simpler the code, the easier it is for you and others involved in maintaining it to understand it. Simplicity means not using clever tricks or unnecessary complexity.

Examples:

Examples of violations of this principle include writing a separate function just to perform addition, or using a bitwise operator (right shift > > 1) to divide integers by 2.

(4 > > 1) === (4 / 2)

The latter may be more effective for some software compilers than the usual division / 2, but it reduces code comprehensibility very much.

Important note: For JS engines, the effectiveness of the shift operation is negated by the costly integer conversion.

With this approach, you are doing clever coding and over-optimisation. Both will make your code less and less comprehensible in the long run, both to other developers and to yourself, because you may have to deal with it again in a month, two years.

 KISS is a very general and abstract design principle which contains almost all other design principles. Design principles describe how to write "good" code. However, what does good code mean? Some people think it's code that runs as fast as possible, some think it's code that involves as many design patterns as possible... But the right answer lies on the surface. Code is information in its purest form. And the basic criteria for the value of information are 1)trustworthiness 2)accessibility 3)understandability. It's clear why reliability and accessibility are important. The code is useless if it has errors or if the server with the application is "dead". So why is code readability important? It is easier to look for errors in the clear code and easier to change, modify and maintain it. So, understandability is the main value a programmer should seek for. But there is one pitfall here. The point is that clarity is a purely subjective thing. We need a more objective criterion of clarity. This criterion is simplicity. Indeed, a simple application is clearer than a complex one. However, simplicity is difficult to achieve. This is what Peter Goodwin says in his book "The Craft of Programming":

If the project is simple, it is easy to understand... Developing a simple project is not so easy. It takes time. For any complex programme, the final decision comes from analysing a huge amount of information. When the code is well designed it seems as if it could not be otherwise, however it is possible that its simplicity is achieved as a result of hard mental work (and a large amount of refactoring). Doing the simple thing is difficult. If the structure of the code seems obvious, don't think it was given without difficulty.

**So, the KISS design principle proclaims that simplicity of code is paramount because simple code is the most comprehensible.**  
Almost all design principles aim to achieve code understandability. If you violate any design principle, you reduce the understandability of the code. Unintelligible code automatically makes people feel that the code is difficult to understand and modify. If any of these principles are violated, the KISS principle is also violated. Therefore, we can say that KISS includes almost all other design principles.  
Design patterns describe the most successful, simple and straightforward solutions to certain problems. If you use a design pattern where there is no problem that the pattern solves, you are violating KISS by making the code unnecessarily complicated. If you do NOT use a design pattern where there is a problem that the pattern solves, you are again violating KISS by making the code more complicated than it could be.

Due to the fact that different people may have different ideas about the concept of 'simplicity', the following **misconceptions about KISS** have become widespread:  
*Misconception 1.* If you think that simple code is code which is easiest to write, you might interpret this as saying that KISS calls you to write whatever you want without thinking about design at all.  
*Misconception 2:* If simple code is code that requires as little knowledge as possible to write, then it can be interpreted that the KISS principle calls for not using design patterns.  
  
Simplicity, on the other hand, should be understood as [not complex, devoid of artificiality, the most natural, not difficult, easily understandable](http://www.classes.ru/all-russian/russian-dictionary-Ushakov-term-59601.htm).

Going back to history, we already said that the KISS principle originated in aircraft construction and there were known cases where overzealous workers nailed extra armour plates on the aircraft to make it more survivable in combat, resulting in the weight of the aircraft becoming greater than designed and the aircraft simply could not take off. In addition, the qualifications of many of the workers were low. In such circumstances, aircraft designs that a drunken, unskilled worker could not assemble incorrectly, even if he wanted to, were of particular value. One echo of the design decisions of the time is the inability to mix and match the wrong plug into the socket inside the computer. However, if a aircraft engineer's work results in a drawing which will be used to create a product, in the case of a programmer the product is the drawing itself (figuratively speaking). In the case of a programmer, he has to write the code so that a drunken unskilled programmer can make changes to it in accordance with the changed business requirements (i.e. change the drawing, not build an aeroplane). Because of the differences in the specifics of aircraft construction and programming, the "Keep it simple stupid" transcription, appropriate in aircraft construction, no longer reflects so well the essence of the principle for the programmer. Many lazy programmers decipher "keep it simple stupid" as "don't bother designing" (compare, for example, the description of the KISS principle in this article with this [description](http://habrahabr.ru/post/153225/) here). Luckily, KISS has [some other transcripts](https://en.wikipedia.org/wiki/KISS_principle), one of which, in my opinion, best captures the essence of KISS in programming - "keep it simple and straightforward". Straightforward translates as simple, honest, straightforward, frank. "Keep it simple and straightforward" can thus be loosely translated as "Keep it simple and declarative", and to achieve declarability requires design.