Software Design Principles DRY and KISS

by Arvind Singh Baghel MVB · **Apr. 19, 18 · Java Zone · Tutorial**

In this article, I am going to explore software design principles and their benefits, why design principles are useful for us, and how to implement them in our daily programming. We will explore the DRY and KISS software design principles.

The DRY Principle: Don't Repeat Yourself

DRY stand for "Don't Repeat Yourself," a basic principle of software development aimed at reducing repetition of information. The DRY principle is stated as, "Every piece of knowledge or logic must have a single, unambiguous representation within a system."

Violations of DRY

"We enjoy typing" (or, "Wasting everyone's time."): "We enjoy typing," means writing the same code or logic again and again. It will be difficult to manage the code and if the logic changes, then we have to make changes in all the places where we have written the code, thereby wasting everyone's time.

How to Achieve DRY

To avoid violating the DRY principle, divide your system into pieces. Divide your code and logic into smaller reusable units and use that code by calling it where you want. Don't write lengthy methods, but divide logic and try to use the existing piece in your method.

DRY Benefits

Less code is good: It saves time and effort, is easy to maintain, and also reduces the chances of bugs.

One good example of the DRY principle is the helper class in enterprise libraries, in which every piece of code is unique in the libraries and helper classes.

KISS: Keep It Simple, Stupid

KISS: Keep It Simple, Stupid

The K[illegible] descriptive to keep the code simple and clear, making it easy to understand. After all, progr[illegible]ges are for humans to understand — computers can only understand 0 and 1 — so keep codin[illegible]aightforward. Keep your methods small. Each method should never be more than 40-50 lines.

Each method should only solve one small problem, not many use cases. If you have a lot of conditions in the method, break these out into smaller methods. It will not only be easier to read and maintain, but it can help find bugs a lot faster.

Violations of KISS

We have all likely experienced the situation where we get work to do in a project and found some messy code written. That leads us to ask why they have written these unnecessary lines. Just have a look at below two code snippets shown below. Both methods are doing the same thing. Now you have to decide which one to use:

```
public String weekday1(int day) {
    switch (day) {
        case 1:
            return "Monday";
        case 2:
            return "Tuesday";
        case 3:
            return "Wednesday";
        case 4:
            return "Thursday";
        case 5:
            return "Friday";
        case 6:
            return "Saturday";
        case 7:
            return "Sunday";
        default:
            throw new InvalidOperationException("day must be in range 1 to 7");
    }
}

public String weekday2(int day) {
    if ((day < 1) || (day > 7)) throw new InvalidOperationException("day must be in range 1 t

    string[] days = {
        "Monday",
        "Tuesday",
        "Wednesday",
        "Thursday"
```

```
29          "Thursday",
30          "Friday",
31          "Saturday",
32          "Sunday"
33      };
34
35      return days[day - 1];
36  }
```


How to Achieve KISS

To avoid violating the KISS principle, try to write simple code. Think of many solutions for your problem, then choose the best, simplest one and transform that into your code. Whenever you find lengthy code, divide that into multiple methods — right-click and refactor in the editor. Try to write small blocks of code that do a single task.

Benefit of KISS

If we have some functionality written by one developer and it was written with messy code, and if we ask for another developer to make modifications in that code, then first, they have to understand the code. Obviously, if the code is written simply, then there will not be any difficulty in understanding that code, and also will be easy to modify.

Summary

While writing any code or module, keep software design principles in mind and use them wisely, make them your habit so you don't need to keep remembering every time. It will save development time and make your software module robust, which could be easy to maintain and extend.

Like This Article? Read More From DZone

Software Design Principles

Software Design Principles DRY and KISS

SOLID Design Principles Explained: Interface Segregation

Free DZone Refcard
Getting Started With Vaadin 10

Topics: JAVA , DRY , KISS , DESIGN PRINCIPLES , TUTORIAL

ava Partner Resources

arn more about Kotlin
tBrains

arn how to refactor a monolithic application to work your way toward a scalable and resilient microsystem.
ghtbend

edictive Analytics + Big Data Quality: A Love Story
elissa Data

uilding Reactive Microservices in Java: Asynchronous and Event-Based Application Design
ed Hat Developer Program