

🇸🇪 På svenska (/sv/) | 📡 (<http://feeds.feedburner.com/JoelAbrahamsson>) 🐦 (<http://twitter.com/joelabrahamsson>)  (<http://www.linkedin.com/in/joelabrahamsson>) 🐱 (<http://www.github.com/joelabrahamsson/>)

You are here: [Home \(/\)](#) / [Programming \(/programming/\)](#) / [Architecture \(/programming/architecture/\)](#) / A simple example of the Open/Closed Principle

[Programming \(/programming/\)](#) / [Architecture \(/programming/architecture/\)](#)

July 13, 2010

A simple example of the Open/Closed Principle

TAGS [open closed principle \(/tag/open-closed-principle/\)](#), [solid principles \(/tag/solid-principles/\)](#)

A few weeks ago I did a presentation (</slides-from-tonights-episerver-meetup-an-introduction-to-solid/>) titled “How uncle Bob changed my life: An introduction to the SOLID principles” for a Swedish user group. I have previously written about a real world example of the Open/Closed Principle (</the-openclosed-principle-a-real-world-example/>) but during the presentation I used a much simpler example which I thought illustrated the principle quite well. I thought I’d post it here as well.

What is the Open/Closed Principle?

Let’s begin with a short summary of what the Open/Closed Principle is. It’s a principle for object oriented design first described by Bertrand Meyer that says that “*software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification*”.

At first thought that might sound quite academic and abstract. What it means though is that **we should strive to write code that doesn’t have to be changed every time the requirements change**. How we do that can differ a bit depending on the context, such as our programming language. When using Java, C# or some other statically typed language the solution often involves inheritance and polymorphism, which is what this example will illustrate.

An example – calculating area

Let’s say that we’ve got a Rectangle class. As most rectangles that I’ve encountered it has a width and a height.

```
public class Rectangle
{
    public double Width { get; set; }
    public double Height { get; set; }
}
```

Now our customer, Aldford (which apparently means “old river-ford”, did you know that?), wants us to build an application that can calculate the total area of a collection of rectangles.

That’s not a problem for us. We learned in school that the area of a rectangle is it’s width multiplied with it’s height and we mastered the for-each-loop a long time ago.

```
public class AreaCalculator
{
    public double Area(Rectangle[] shapes)
    {
        double area = 0;
        foreach (var shape in shapes)
        {
            area += shape.Width*shape.Height;
        }

        return area;
    }
}
```

We present our solution, the AreaCalculator class to Aldford and he signs us his praise. But he also wonders if we couldn’t extend it so that it could calculate the area of not only rectangles but of circles as well.

That complicates things a bit but after some pondering we come up with a solution where we change our Area method to accept a collection of objects instead of the more specific Rectangle type. Then we check what type each object is of and finally cast it to it’s type and calculate it’s area using the correct algorithm for the type.

```
public double Area(object[] shapes)
{
    double area = 0;
    foreach (var shape in shapes)
    {
        if (shape is Rectangle)
        {
            Rectangle rectangle = (Rectangle) shape;
            area += rectangle.Width*rectangle.Height;
        }
        else
        {
            Circle circle = (Circle)shape;
            area += circle.Radius * circle.Radius * Math.PI;
        }
    }

    return area;
}
```

The solution works and Aldford is happy.

Only, a week later he calls us and asks: “extending the AreaCalculator class to also calculate the area of triangles isn’t very hard, is it?”. Of course in this very basic scenario it isn’t but it does require us to modify the code. That is, AreaCalculator isn’t **closed for modification** as we need to change it in order to extend it. Or in other words: it isn’t **open for extension**.

In a real world scenario where the code base is ten, a hundred or a thousand times larger and modifying the class means redeploying it’s assembly/package to five different servers that can be a pretty big problem. Oh, and in the real world Aldford would have changed the requirements five more times since you read the last sentence :-)

A solution that abides by the Open/Closed Principle

One way of solving this puzzle would be to create a base class for both rectangles and circles as well as any other shapes that Aldford can think of which defines an abstract method for calculating it’s area.

```
public abstract class Shape
{
    public abstract double Area();
}
```

Inheriting from Shape the Rectangle and Circle classes now looks like this:

```
public class Rectangle : Shape
{
    public double Width { get; set; }
    public double Height { get; set; }
    public override double Area()
    {
        return Width*Height;
    }
}

public class Circle : Shape
{
    public double Radius { get; set; }
    public override double Area()
    {
        return Radius*Radius*Math.PI;
    }
}
```

As we've moved the responsibility of actually calculating the area away from AreaCalculator's Area method it is now much simpler and robust as it can handle any type of Shape that we throw at it.

```
public double Area(Shape[] shapes)
{
    double area = 0;
    foreach (var shape in shapes)
    {
        area += shape.Area();
    }

    return area;
}
```

In other words we've closed it for modification by opening it up for extension.

When should we apply the Open/Closed Principle?

If we look back our previous example, where did we go wrong? Clearly even our first implementation of the Area wasn't open for extension. Should it have been? I'd say that it all depends on context. If we had had very strong suspicions that Aldford would ask us to support other shapes later on we could probably have prepared for that from the get-go. However, often it's not a good idea to try to anticipate changes in requirements ahead of time, as at least my psychic abilities haven't surfaced yet and preparing for future changes can easily lead to overly complex designs. Instead, I would suggest that we focus on writing code that is well written enough so that it's easy to change if the requirements change.

Once the requirements do change though it's quite likely that they will change in a similar way again later on. That is, if Aldford asks us to support another type of shape it's quite likely that he soon will ask for support for a third type of shape.

So, in other words, I definitely think we should have put some effort into abiding by the open/closed principle once the requirements started changing. Before that, in most cases, I would suggest limiting your efforts to ensuring that the code is well written enough so that it's easy to refactor if the requirements starts changing.

More examples of the Open/Closed Principle?

If you've got any other good and straight forward examples of the open/closed principle I'd love to hear about them as I really enjoy studying the SOLID principles in general and different ways to apply OCP in particular.

Any other feedback is of course also most welcome!

PS. For updates about new posts, sites I find useful and the occasional rant you can follow me on Twitter (<http://twitter.com/joelabrahamsson>). You are also most welcome to subscribe to the RSS-feed. (<http://feeds.feedburner.com/JoelAbrahamsson>)



Joel Abrahamsson

I'm a passionate web developer and systems architect living in Stockholm, Sweden. I work as CTO for a large media site and enjoy developing with all technologies, especially .NET, Node.js, and ElasticSearch. [Read more \(/about-joel/\)](/about-joel/)

Comments

58 Comments

JoelAbrahamsson.com

 Login ▾

 Recommend 18

 Tweet

 Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



Guest • 6 years ago

Clear and Excellent ...

9 ^ | ▾ • Reply • Share ›





 Guest • 6 years ago

we were been discussing the same with interface isn't it?

^ | ▾ • Reply • Share ›



Guest  • 6 years ago

 Yes da..
Now only L and I said that
1 ^ | v • Reply • Share ›

[Show more replies](#)



Ed Cottrell • 5 years ago

FYI, most of the times that you used "it's" above should actually be "its." The rule is that "it's" always means "it is" ("it's raining") or "it has" ("it's been raining all day"). When you mean a possessive, you use "its" and should say, for example, "The area of a rectangle is its height times its width."

12 ^ | v • Reply • Share ›



Mohammed Housseyn Taleb ➔ Ed Cottrell • 2 years ago

thank you soo much :)

1 ^ | v • Reply • Share ›



LeMec ➔ Ed Cottrell • 5 years ago

Wrong.

<http://forum.wordreference....>

^ | v • Reply • Share ›



Ed Cottrell ➔ LeMec • 5 years ago

Actually, I'm correct here. An apostrophe followed by "s" will convert normal and proper nouns---"the dog," "the house," Susan, etc.---into possessive forms. It doesn't work that way with pronouns like "it," "him," "they," etc. The possessive form of "it" is "its," with no apostrophe. See, for example, this explanation:

<https://owl.english.purdue....> See also this dictionary definiton: <http://www.merriam-webster....>

7 ^ | v • Reply • Share ›

[Show more replies](#)

[Show more replies](#)



Safa • 5 years ago

Another example about OCP

in the book called : concise introduction to software engineering chapter 6: 6.1.3:Open close

Principle

example about Printer and client



4 ^ | v • Reply • Share ›



TrendingWendy → Safa • 4 years ago

So when you start a project, you have to create abstract classes for all your classes that could possibly be extended in the future? That's exactly what the author was warning against doing, "However, often it's not a good idea to try to anticipate changes in requirements ahead of time, as at least my psychic abilities haven't surfaced yet and preparing for future changes can easily lead to overly complex designs."

^ | v • Reply • Share ›



Dave Birkhead → TrendingWendy • 2 years ago

It is however something that wouldn't take a huge amount of time to adapt in an already written project, provided it has been written in a relatively Object Oriented manner.

^ | v • Reply • Share ›



Lig • 7 years ago

Hi, I wonder if you divide the classes in projects so that you don't have to recompile the whole thing and just the dll itself would be enough, my problem has always been changing the code then you have to recompile dlls which for some reason always end up messing up the other components as well.

2 ^ | v • Reply • Share ›



Piyush Kumar • 4 months ago

Very Good explanation. However, I have one doubt. Lets say, I had some requirement few days back - I had implemented it using interface(lets say Book with method findBook()) with its implementor(SoftwareBook). After some point of time, the same requirement has changed and now, it requires just to add a one line of code in findBook method of SoftwareBook class.Lets assume requirement is to just print the name of the book in the findBook method. So in this case what should

I do. Should I create :

1. A new implementor with a same method implementation with one line extra which prints the book name or
2. Or just add `System.out.println`(using Java) in the `findBook` method of the `SoftwareBook` class.

If I go with option 1, the code is getting duplicated so there will be a maintainability issue and if I go with option 2, that's violating OCP. So, what should I do in this case ?

1 ^ | v • Reply • Share ›



Henze Berkheij → Piyush Kumar • 2 months ago

That would be wrong. The method says: `find`. not `findAndPrint`. Even if the method name was altered to reflect the change, you'd be violating the Single Responsibility Principle too. because it would find you the book, and then print its name. if all your Book implementors require to print the name, you'd add that to your interface. By doing so you extend the functionality, but you do not change the existing functionality. e.g.: Every part of the system that was expecting to find a book and not printing it, still will just find a book and not printing it. Every part of the system that requires printing the name, will see the method `printName` is there and will execute it.

If not all your implementors require to print the name, you should create another interface that implements your Book interface with the addition of a `printName` method. then you'd be working conform the Interface Segregation principle.

^ | v • Reply • Share ›



Drew Verlee • 2 years ago

Now Alford asks if you could just add another method "Perimeter" to your system that applies to Circle and Radius. Now were forced to open up our objects and modify them. Would this not be a violation of the open closed principle?

It seems to me that the OpenClosed Principle is probably better expressed by a language that can adequately solve the Expression problem. https://en.wikipedia.org/wiki/Expression_problem

1 ^ | v • Reply • Share ›



Henze Berkheij → Drew Verlee • 2 months ago

no. you are not modifying functionality, you are adding functionality, you are extending it. A Circle would still be a Shape, but it has an other interface as well that adds the Perimeter method. if done right, Liskov would be so happy.

^ | v • Reply • Share ›



Drew Verlee → Henze Berkheij • 2 months ago

A class is a function and so extending a class is similar to modifying a function. The class contains branches of logic `area` `perimeter` that you call by passing it the function name. Indeed you can do this in ruby.

If you don't control the original Shape class (e.g its part of a library), then adding/extending the classes you want to have `perimeter` is going to be tricky in *some languages*.

We don't need worry about the perimeter of every shape in our system. If their is a point at which we do need every shape to have a perimeter then this requirement should be explicit in order to participate their and no where else.

^ | v • Reply • Share ›



frases para orkut • 7 years ago

Would it make sense to create a virtual (replaceable) the "Run" to be asking all TAG_TYPE laden objects repository so they can do what has been omitted for the default constructor as the constructor with parameters that have done?

1 ^ | v • Reply • Share ›



deepa ayachit • 9 years ago

But I think interface would have been better

1 ^ | v • Reply • Share ›



Ed Blackburn • 9 years ago

I like the example nice and simple. Personally I'd favour an interface, it helps prevent you from falling into some kind of template pattern abuse (favour composition over inheritance).

1 ^ | v • Reply • Share ›



Ratn • 9 years ago

Well said ,It is also good example of encapsulation .if you do good encapsulation design is mostly flexible.

1 ^ | v • Reply • Share ›



Developer Kafasi • 4 months ago

Great article! I've just read this too: <https://www.developer.kafasi>



Great article! I've just read this too. <https://www.developer.karasi...>

Also great examples for open closed principle in c#. Check it out!

^ | v • Reply • Share ›



Swastik • a year ago

Very nice example...

^ | v • Reply • Share ›



Sudarshan • a year ago

effective and understandable post. Thanks

^ | v • Reply • Share ›



Satish Padidem • a year ago

Good one.

^ | v • Reply • Share ›



Nicola Gallazzi • a year ago

Nice job, thanks a lot

^ | v • Reply • Share ›



ERICK MWAZONGA • 2 years ago

great!

^ | v • Reply • Share ›



Yiannos Georgantas • 2 years ago

Perfect example, very well explained. Thank you.

^ | v • Reply • Share ›



Jeff Finlay • 2 years ago

Thanks! Excellent simple example that illustrates the progression on why we derive from abstracted classes. Also like the mention that it is okay to implement a concrete class first and abstraction comes later, because trying to predict the future can be wasteful

^ | v • Reply • Share ›



Stepan • 2 years ago

THE OPEN-CLOSED PRINCIPLE OF PROGRAMMING



I thank you very much! It is a really cool article! But I have a question.

How can the "open for extension, closed for modification" help to decrease redeploying assembly?

And does Open/Closed principle help to avoid recompiling libraries?

In a real world scenario where the code base is ten, a hundred or a thousand times larger and modifying the class means redeploying it's assembly/package to five different servers that can be a pretty big problem.

^ | v • Reply • Share ›



Dharmesh Sujeeun • 2 years ago

Great explanation - better than wikipedia's anyways. The example is well thought of also. Thank you for taking the time to share this!

^ | v • Reply • Share ›



LazerMax • 2 years ago

Thanks for the tutorial, it's one of the most clear and simple out there.

^ | v • Reply • Share ›



charliebrowncpt • 3 years ago

Thanks Joel. Concise, well explained and simple - exactly how software development ought to be done.

^ | v • Reply • Share ›



lingyong • 3 years ago

Very clear! Thanks!

^ | v • Reply • Share ›



Daniel • 3 years ago

Excellent! Thank you :) Extremely useful 6 years after written!

^ | v • Reply • Share ›



andreinaromero • 3 years ago

Is a good example

^ | v • Reply • Share ›



Erikzon Zarco • 3 years ago

Regards



Regarus

^ | v • Reply • Share ›



John • 3 years ago

The following video starting from the definition of this principle, identifies some common code smell and puts it into practice with a javascript example



^ | v • Reply • Share ›



paper writing services • 3 years ago

Seems like a pretty wonderful idea that you have shared this kind of information for those people to know on what is the difference between open/closed principle that they might going to encounter in the future when they will study about programming.

^ | v • Reply • Share ›



Peter Brightman • 5 years ago



Open for extension, closed for changes. A simple example is also an application that allows addons/plugins. For example Notepad++ or NUnit. Without changing the application, you can extend it by providing a DLL that extends the functionality without changing the basic application.

^ | v • Reply • Share ›



singhswat • 5 years ago

very nice !!! I would appreciate if hypothetical examples are associated/ referenced with framework examples.. never the the less, very good and quality stuff.. thanks

^ | v • Reply • Share ›



Safa • 5 years ago

Helpful post ,Thank you Joel:)

I found another example about OCP in Software Engineering book,7th edition by Roger Pressman,chapter 10,Figure 10.4 but the author use interface instead of abstract class

<http://www.amazon.com/Softw...>

^ | v • Reply • Share ›



inf3rno • 5 years ago

Clean Code, chapter 6: Objects and data structures - strange that RC Martin did not use this term...

^ | v • Reply • Share ›



applicant tracking software • 8 years ago

Joel, huge help. I've been working on a new project to create a base document class which holds all the shared information, while using subclasses for each specific document type. This open/close example was a huge help!

^ | v • Reply • Share ›



Nirav • 8 years ago

Joel,

Please can you put an example in interface for open/closed principle?

^ | v • Reply • Share ›



Efim • 8 years ago

Good example.

Question: in first example is a high coupling and low cohesion too?

thanks

^ | v • Reply • Share ›



Jasmin • 8 years ago

A lot of thanks to u to illustrate the idea so nicely.It just clear up my all confusions about OCP.

^ | v • Reply • Share ›



Deepa Ayachit • 9 years ago

Good and simple example :)

^ | v • Reply • Share ›



Nicklas • 9 years ago

For pure code examples of SOLID principles you can look here: <http://solidexamples.codepl...>

Sharing is caring ♥

Tweet

أعجبني ٥٣

More on SOLID

- › Introduction to Inversion of Control (/inversion-of-control-an-introduction-with-examples-in-net/)
- › The Open/Closed Principle – A real world example (/the-openclosed-principle-a-real-world-example/)

More about Software Architecture

Programming (/programming/) ► Architecture (/programming/architecture/) ►

August 09, 2011

The DRY obsession

› The Don't Repeat Yourself principle is important. But it's not the only important principle or consideration for software design. Like all good things it should be used with caution and afterthought.

(/the-dry-obsession/)

Programming (/programming/) ► Architecture (/programming/architecture/) ►

November 09, 2010

Inversion of Control – It's broader than just injecting components

➤ Today I attended an excellent workshop about Scala development held by Ted Neward. During the session Ted demonstrated a different, functional, aspect of the Inversion of Control concept which he illustrated in Scala. Since it's fully possible to do the same thing in C# I thought would borrow and modify his example to show a different type of Inversion of Control.

(/inversion-of-control-its-broader-than-just-injecting-components/)

[Programming \(/programming/\)](/programming/) ► [Architecture \(/programming/architecture/\)](/programming/architecture/) ►

June 21, 2010

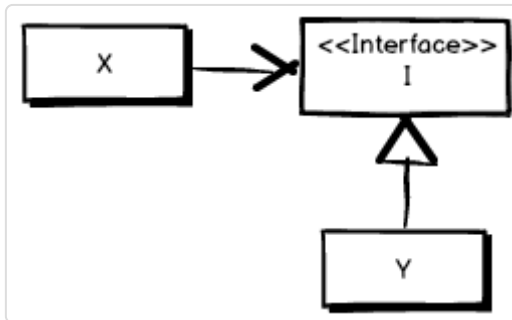
Slides from tonight's EPiServer meetup – An introduction to SOLID

(/slides-from-tonights-episerver-meetup-an-introduction-to-solid/)

[Programming \(/programming/\)](/programming/) ► [Architecture \(/programming/architecture/\)](/programming/architecture/) ►

March 29, 2010

Inversion of Control – An Introduction with Examples in .NET



➤ An introduction to Inversion of Control, using the Dependency Injection and Service Locator patterns, along with simple examples in C#.

(/inversion-of-control-an-introduction-with-examples-in-net/)

[Programming \(/programming/\)](/programming/) ► [Architecture \(/programming/architecture/\)](/programming/architecture/) ►

November 15, 2009

The Open/Closed Principle – A real world example

➤ Out of the five SOLID principles the Open/Closed Principle is probably the one that I've had the hardest time understanding. However, a while ago though I found some code that I had written years ago that made me think to myself "Hey, this is clearly violating the Open Closed Principle!".

(/the-openclosed-principle-a-real-world-example/)

