**EOC Posting Template**

*Last updated: March 1, 2023*

*Please remove* [*smart quotes*](https://techtarget.box.com/s/vmxfhq1h6bcox4z9xzf9h3gtyv8whdlk) *from articles. Access our* [*posting template cheat sheet here*](https://techtarget.box.com/s/axkl0krtdeuq9osrbzd4rl9c3130nv7i)*.*

|  |  |
| --- | --- |
| Live URL |  |
| Row ID |  |

*Note: For Guest Contributor content only, the searcher intent, target keyword and keyword volume are optional.*

|  |
| --- |
| **Google results for keyword**  **(Searcher intent)** [**?**](#InboundLinks) |
| Copy/paste top 3 title, URL and summary results for primary keyword |

**Target keyword:**

**Keyword volume:**

**Title [60 characters]:**

How to apply the open

**Meta description [max. 156 characters]:**

Keep your code extensible, pluggable and adaptable. That's what SOLID's open-closed principle in Java means. Here's an example of how to properly apply the open-closed concept in Java.

**Summary [150-180 characters]:**

How does the open-closed SOLID principle work in a Java program? Here we show you what this important principle means, and how to implement the open-closed principle in Java.

Article Visuals

(include URLs, sources, captions, alt text)

Remember to check [Headshot Library](https://techtarget.box.com/s/1to0kabplk3s37bs77qw29gz28bm4a33), [Chart Library](https://techtarget.box.com/s/gkpzg1727s96j4p0l4m7iusoxv3wr35v) and [Press Room Library](https://techtarget.box.com/s/s4a2lqnoxhlfu8gr5rh2r8ba7ub6mo7w).

Embedded video YouTube or BrightTALK URL:

Podcast URL:

**Body** **[?](#Body" \o "Paste in body copy here directly from other Word document.):**

The SOLID, open-closed principle in Java, asserts that a well-designed software component will be open for ongoing extension, but closed to edits and modification.

In other words, a Java class that supports the open-closed principle will find future uses in originally un-envisioned scenarios, and be flexible enough to work with newly created classes, interfaces and Java records without any changes to the component's source code.

In other words, a Java class that supports the open-closed principle will be flexible enough to work with newly created classes, interfaces and Java records without the need to change the component's inner workings.

This allows the component to find future uses in originally un-envisioned scenarios, which gives the component future extensibility.

Developers often stumble over the open-closed principle in Java due to the fact that the original definition uses the word 'extension', which in Java implies *inheritence*. Confusingly, inheritance isn't the best way to implement the open-closed principle in Java.

### Open-closed principle example

Here is a simple example of a class that *fails* to implement the open-closed principle properly.

Imagine a program that has two instances of a square and needs a custom component to compare their area. The code would look like this:

class Square() {

int height;

int area() { return height \* height; }

}

public class OpenOpenExample {

public int compareArea(Square a, Square b) {

return a.area() - b.area();

}

}

The issue with the OpenOpenExample isn’t necessarily obvious.

For this specific use case, the OpenOpenExample works perfectly well. It returns zero if the two squares are the same size, a positive number if the first square is larger, and a negative number if the first square is smaller.

However, a problem quickly arises when a circle is brought into the mix.

### The extension problem

To compare the area of two circles, we would need to modify the OpenOpenExample. That would be a violation of the SOLID open-closed principle.

The following is the example updates the OpenOpenExample class to provide support for circles.

class Circle {

int r;

int area() { return Math.PI\*r\*r\*;}

}

class OpenOpenExample {

public int compareArea(Square a, Square b) {

return a.area() - b.area();

}

public int compareArea(Circle x, Circle y) {

return x.area() - y.area();

}

}

You can easily imagine the OpenOpenExample growing larger and larger as more shapes are introduced into the problem domain.

## The open-closed principle in Java applied

The addition of an interface to our example will help overcome the violation of the open-closed principle. An interface will allow for infinite future extensions without the need to ever edit the class again.

To fix this example, we first create an interface that both the circle and the square implement.

interface Shape {

int area();

}

class Circle **implements** Shape {

int r;

int area() { return Math.PI\*r\*r\*;}

}

class Square() **implements** {

int height;

int area() { return height \* height; }

}

We then create a new class named OpenClosedExample which has a single compareArea method that uses the Shape interface as arguments:

public class OpenClosedExample {

public int compareArea(Shape a, Shape b) {

return a.area() - b.area();

}

}

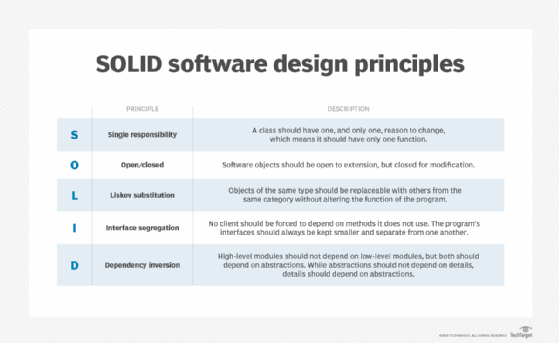
With this small change applied, any class that implements the Shape interface can be passed as an argument to the OpenClosedExample’s compareArea method.

This makes the class extensible to an infinite number of Shape classes that could potentially be created.

### SOLID principles applied

Furthermore, the OpenClosedExample will support all of these new classes without any edits needed to the source code.

In other words, this well designed Java component will be open for extension but closed to edits and modification, which just happens to be the official definition of the SOLID open-closed principle.



The open-closed concept is one of the five SOLID principles in object-oriented programming.

Alt: open-closed in SOLID

**Pull quote**

**Sidebar**

**Next Steps***(optional – this is another area for article links, but do not count as inbound links)*

**Tweets (***optional, your director will advise if needed***):**

**Contributor:**

**Contact Editor** [**?**](#_top)**:**

**Topic Channels**[**?**](#_top)**:** *Use the* [*taxonomy lookup sheet*](https://techtarget.box.com/v/EditorialTaxonomies) *to find appropriate topic channels for your content.*

**Inbound links** [**?**](#InboundLinks)**:** *For inbound links in Next Steps, please indicate "Next Steps" and the anchor text should be the exact title of your new content.*

|  |  |
| --- | --- |
| **Link** | **Anchor text** |
|  |  |
|  |  |
|  |  |

**Series**[**?**](#Series)**:** Choose an item.

***If this content completes a collection, please include the following:***

**Collection Title (65 characters max):**

**Abstract:**

**URLs in collection (highlight first article in the collection (collection main page)):**

1. *URL*
2. *URL*
3. *URL*
4. *URL*
5. *URL*
6. *URL*