Do you use design patterns? [closed]

Asked 16 years, 4 months ago Modified 6 months ago Viewed 4k times

29

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 7 months ago.

Improve this question

What's the penetration of design patterns in the real world? Do you use them in your day to day job - discussing how and where to apply them with your coworkers - or do they remain more of an academic concept?

Do they actually provide actual value to your job? Or are they just something that people talk about to sound smart?

Note: For the purpose of this question ignore 'simple' design patterns like *Singleton*. I'm talking about designing your code so you can take advantage of *Model View Controller*, etc.

language-agnostic design-patterns

Share

Improve this question

Follow

edited Aug 23, 2008 at 15:00

Chris

6,842 ● 6 ● 53 ● 67

asked Aug 14, 2008 at 19:39

Chris Smith

18.7k ● 14 ● 61 ● 81

16 Answers

Sorted by: Highest score (default)

56

Any large program that is well written will use design patterns, even if they aren't named or recognized as such. That's what design patterns are, designs that repeatedly and *naturally* occur. If you're interfacing with an ugly API, you'll likely find yourself implementing a `Facade` to clean it up. If you've got messaging between components that you need to decouple, you may find yourself using `Observer`. If you've got several interchangeable algorithms, you might end up using `Strategy`.

It's worth knowing the design patterns because you're more likely to recognize them and then converge on a clean solution more quickly. However, even if you don't know them at all, you'll end up creating them eventually (if you are a decent programmer).

And of course, if you are using a modern language, you'll probably be forced to use them for some things, because they're baked into the standard libraries.

Share Improve this answer Follow

edited Aug 14, 2008 at 20:00

answered Aug 14, 2008 at 19:54

Derek Park

46.8k ● 16 ● 59 ● 76

11

In my opinion, the question: "Do you **use** design pattern?", alone is a little flawed because the answer is universally YES.

Let me explain, we, programmers and designers, all use design patterns... we just don't always realise it. I know this sounds cliché, but you don't go to patterns, patterns come to you. You design stuff, it might look like an existing pattern, you name it that way so everyone understand what you are talking about and the rationale behind your design decision is stronger, knowing it has been discussed *ad nauseum* before.

I personally use patterns as a communication tool. That's it. They are not design solutions, they are not best practices, they are not tools in a toolbox.

Don't get me wrong, if you are a beginner, books on patterns will show you how a solution is best solved "using" their patterns rather than another flawed design.

You will probably learn from the exercise. However, you have to realise that this doesn't mean that every situation needs a corresponding pattern to solve it. Every situation has a quirk here and there that will require you to think about alternatives and take a difficult decision with no perfect answer. **That's** design.

Anti-pattern however are on a totally different class. You actually **want** to **actively** avoid anti-patterns. That's why the name anti-pattern is so controversial.

To get back to your original question:
"Do I use design patterns?", Yes!
"Do I actively lean toward design patterns?", No.

Share Improve this answer

Follow

answered Aug 14, 2008 at 20:16

Coincoin

28.5k ● 7 ● 56 ● 76

4

Yes. Design patterns can be wonderful when used appropriately. As you mentioned, I am now using Model-View-Controller (MVC) for all of my web projects. It is a very common pattern in the web space which makes server-side code much cleaner and well-organized.

Beyond that, here are some other patterns that may be useful:

- MVVM (Model-View-ViewModel): a similar pattern to MVC; used for WPF and Silverlight applications.

- Composition: Great for when you need to use a hierarchy of objects.
- Singleton: More elegant than using globals for storing items that truly need a single instance. As you mentioned, a simple pattern but it does have its uses.

It is worth noting a design pattern can also highlight a lack of language features and/or deficiencies in a language. For example, iterators are now built in as part of newer languages.

In general design patterns are quite useful but you should not use them everywhere; just where they are a good fit for your needs.

Share Improve this answer Follow

edited Jul 18, 2010 at 13:37

answered Feb 14, 2010 at 14:03

Justin Ethier

134k ● 52 ● 232 ● 287

3

I try to, yes. They do indeed help maintainability and readability of your code. However, there are people who do abuse them, usually (from what I've seen) by forcing a system into a pattern that doesn't exist.

Share Improve this answer Follow

answered Aug 14, 2008 at 19:41

Thomas Owens
116k ● 99 ● 317 ● 436

I try to use patterns if they are applicable. I think it's kind of sad seeing developers implement design patterns in code just for the sake of it. For the right task though, design patterns can be very useful and powerful.

3

Share Improve this answer

Follow

answered Aug 14, 2008 at 19:46

Patrik Svensson
13.8k ● 8 ● 58 ● 77

But how can you become familiar with it, if you don't try to use it?! First time it will suck, but with time you will start to using them appropriately. In other works, how can you succed without failure? – Nikita Fedyashev Mar 29, 2009 at 18:10

Of course you should experiment, but not with production code. That was what I was talking about. – Patrik Svensson Mar 30, 2009 at 8:30

But isn't production code the true test that what you've done actually works - and works well? – Steven Evers May 15, 2009 at 22:31

There are many design patterns beyond the simple that are used in "real world". Good example Stackoverflow uses the Model View Controller Pattern. I have used Class Factories multiple times in projects for my employer, and I have seen many already written projects using them as well.

2

I am not saying every design pattern is being used but many are.

Share Improve this answer

Follow

answered Aug 14, 2008 at 19:42

Tanerax
5,856 • 5 • 30 • 28

2

Yes we do, it usually happens when we start designing something and then someone notices that it resembles an existing pattern. We then take a look at it and see how it would help us achieve our goal.

We also use patterns that are not documented but that emerge from designing a lot.

Mind you, we don't use them a lot.

Share Improve this answer

Follow

answered Aug 14, 2008 at 19:44

mbillard
38.8k • 18 • 75 • 98

2

Yes, Factory, Chain of Responsibility, Command, Proxy, Visitor, and Observer, among others, are in use in a codebase I work with daily. As far as MVC goes, this site seems to use it quite well, and the devs couldn't say enough good things in the latest podcast.

Share Improve this answer

Follow

edited Aug 14, 2008 at 20:02

answered Aug 14, 2008 at 19:44

Greg Hurlman

17.8k ● 6 ● 55 ● 87

1

Yes, I use a lot of well known design patterns, but I also end up building some software that I later find out uses a 'named' design pattern. Most elegant, reusable designs could be called a 'pattern'. It's a lot like dance moves. We all know the waltz, and the 2-step, but not everyone has a name for the 'bump and scoot' although most of us do it.

Share Improve this answer

Follow

answered Aug 14, 2008 at 19:47

Brad Barker

2,083 ● 3 ● 17 ● 28

1

MVC is very well known so yes we use design patterns quite a lot. Now if your asking about the Gang of Four patterns, there are several that I use because other maintainers will know the design and what we are working towards in the code. There are several though that remain fairly obscure for what we do, so if I use one I don't get the full benefits of using a pattern.

Are they important, yes because it gives you a method of talking about software design in a quick efficient and generally accepted way. Can you do better custom solutions, well yes (sorta)?

The original GoF patterns were pulled from production code, so they catalogued what was already being used in the wild. They aren't purely or even mostly an academic thing.

Share Improve this answer

Follow

answered Aug 14, 2008 at 19:47

Dan Blair

2,381 ● 3 ● 21 ● 32

1

I find the MVC pattern really useful to isolate your model logic, which can than be reused or worked on without too much trouble. It also helps de-coupling your classes and makes unit testing easier. I wrote about it recently (yes, shameless plug here...)

Also, I've recently used a factory pattern from a base class to generate and return the proper DataContext class that I needed on the fly, using LINQ.

Bridges are used when trying when trying to glue together two different technologies (like Cocoa and Ruby on the Mac, for example)

I find, however, that whenever I implement a pattern, it's because I knew about it before hand. Some extra thought generally goes into it as I find I must modify the original pattern slightly to accommodate my needs.

You just need to be careful not to become and architecture astronaut!

Share Improve this answer

Follow

answered Aug 14, 2008 at 19:50

Louis Salin

541 ● 1 ● 5 ● 8

1

Yes, design patterns are largely used in the real world - and daily by many of the people I work with.

In my opinion the biggest value provided by design patterns is that they provide a universal, high level language for you to convey software design to other programmers.

For instance instead of describing your new class as a "utility that creates one of several other classes based on some combination of input criteria", you can simply say it's an *"abstract factory"* and everyone instantly understands what you're talking about.

Share Improve this answer

Follow

answered Aug 27, 2008 at 17:50

Vinnie

12.7k ● 15 ● 61 ● 80

1

Yes, design patterns or abstractly patterns are part of my life, where I look, I begin to see them. Therefore, I am surrounded by them. But, as you know, little knowledge is a dangerous thing. Therefore, I strongly recommend you to read GoF book.

One of the main problems about design patterns, most developers just do not get the idea, or do not believe in

them. And most of the time they argue about the variables, loops, or switches. But, I strongly believe that if you do not speak the pattern language, your software will not go far and you will find yourselves in a maintenance nightmare.

As you know, anti-pattern is also dangerous thing and it happens when you have little expertise on design patterns. And refactoring anti-patterns is much more harder. As a recommended book about this problem please read "AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis".

answered Mar 30, 2010 at 11:56

baris.aydinoz

1,950 ● 2 ● 18 ● 30

1

Yes.

We are even using them in my current job: Mainframe coding with COBOL and PL/I.

So far I have seen Adaptor, Visitor, Facade, Module, Observer and something very close to Composite and Iterator. Due to the nature of the languages it's mostly strutural patterns that are used. Also, I'm not always sure that the people who use them do so conciously :D

Share Improve this answer

Follow

answered Oct 19, 2012 at 8:45

Anders Johansen

10.4k ● 7 ● 36 ● 52

0

I absolutely use design patterns. At this point I take MVC for granted as a design pattern. My primary reason for using them is that I am humble enough to know that I am likely not the first person to encounter a particular problem. I rarely start a piece of code knowing which pattern I am going to use; I constantly watch the code to see if it naturally develops into an existing pattern.

I am also very fond of Martin Fowler's Patterns of Enterprise Application Architecture. When a problem or task presents itself, I flip to related section (it's mostly a reference book) and read a few overviews of the patterns. Once I have a better idea of the general problem and the existing solutions, I begin to see the long term path my

code will likely take via the experience of others. I end up making much better decisions.

Design patterns definitely play a big role in all of my "for the future" ideas.

Share Improve this answer

Follow

answered Aug 14, 2008 at 21:50

Barrett Conrad

1,878 ● 14 ● 14

-1

```
interface MediaPlayer {
    void play(String audioType, String fileName);
}

interface AdvancedMediaPlayer {
    void playVlc(String fileName);
    void playMp4(String fileName);
}

class VlcPlayer implements AdvancedMediaPlayer {
    @Override
    public void playVlc(String fileName) {
        System.out.println("Playing vlc file. Name: "
    }

    @Override
    public void playMp4(String fileName) {
        // Do nothing
    }
}

class Mp4Player implements AdvancedMediaPlayer {
    @Override
    public void playVlc(String fileName) {

    }

    @Override
    public void playMp4(String fileName) {
```

```
        System.out.println("Playing mp4 file. Name: "
    }
}

class MediaAdapter implements MediaPlayer {
    AdvancedMediaPlayer advancedMusicPlayer;

    public MediaAdapter(String audioType) {
        if (audioType.equalsIgnoreCase("vlc")) {
            advancedMusicPlayer = new VlcPlayer();
        } else if (audioType.equalsIgnoreCase("mp4"))
            advancedMusicPlayer = new Mp4Player();
        }
    }

    @Override
    public void play(String audioType, String fileName
        if (audioType.equalsIgnoreCase("vlc")) {
            advancedMusicPlayer.playVlc(fileName);
        } else if (audioType.equalsIgnoreCase("mp4"))
            advancedMusicPlayer.playMp4(fileName);
        }
    }
}

class AudioPlayer implements MediaPlayer {
    MediaAdapter mediaAdapter;

    @Override
    public void play(String audioType, String fileName
        if (audioType.equalsIgnoreCase("mp3")) {
            System.out.println("Playing mp3 file. Name
        } else if (audioType.equalsIgnoreCase("vlc") |
audioType.equalsIgnoreCase("mp4")) {
            mediaAdapter = new MediaAdapter(audioType)
            mediaAdapter.play(audioType, fileName);
        } else {
            System.out.println("Invalid media. " + aud
supported");
        }
    }
}

public class AdapterPatternDemo {
```

```
    public static void main(String[] args) {
        AudioPlayer audioPlayer = new AudioPlayer();

        audioPlayer.play("mp3", "beyond the horizon.mp
        audioPlayer.play("mp4", "alone.mp4");
        audioPlayer.play("vlc", "far far away.vlc");
        audioPlayer.play("avi", "mind me.avi");
    }
}
```

Share Improve this answer

Follow

edited Jun 1 at 20:47

gurkan

1,482 ● 7 ● 19 ● 28

answered May 22 at 9:44

Dileep Gowda

1