The easiest way to learn design patterns

With C# 12 code samples on .NET 8

International bestseller

Fiodar Sazanavets

The easiest way to learn design patterns

With C# 12 code samples on .NET 8

Fiodar Sazanavets

This book is for sale at
http://leanpub.com/the-easiest-way-to-learn-design-patterns

This version was published on 2024-01-14

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Fiodar Sazanavets by spreading the word about this book on Twitter!

The suggested hashtag for this book is #designpatterns.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#designpatterns

I dedicate this book to my dad Dzmitry Sazanavets, who is sadly not with us anymore. Rest in peace.

I also dedicate this book to my wife Olga. Without her support, this book would not have been possible.

Contents

Introduction

Design patterns are something that you will need to get familiar with as a programmer who works with object-oriented languages. This is primarily because they represent well-defined solutions to common software development problems. So, instead of thinking through all the details of your solution, you can simply check if any of the existing design patterns can be used. You won't have to reinvent the wheel.

As a software developer, you would be familiar with reusable software components, such as methods, functions, public classes, libraries, etc. Well, you can think of design patterns as reusable solutions to common software problems. If you are familiar with which design patterns can solve a specific type of problem, you will no longer have to come up with a bespoke solution, which could have taken you a significant amount of time to design and implement. All you'll have to do is just pick the most suitable design pattern and apply it in a specific way to solve this specific problem. The solution is already there. The time taken to implement it would not be much greater than the time it takes you to type the code in.

If you are new to design patterns, let's go through a quick overview of what they are.

What design patterns are

Design patterns are prescribed ways of implementing solutions to common problems. While every specific implementation of them will be bespoke and only relevant to a specific codebase, they still determine how your code should be structured. A design pattern would tell you how different object types in your code should

interact with one another. Specific objects would have specific roles, which are prescribed by a specific design pattern. However, a specific implementation of those objects would still be relevant only to a specific problem.

If you will, you can compare the concept of design patterns with Agile principles. For example, if you follow Scrum, you will go through all of the relevant ceremonies (daily standups, sprint reviews, etc.). You would have specific roles (Scrum master, product owner, and Scrum team members). You would work in sprints of a specific length. However, the problems that you will be solving during your sprints will be very specific to your organization. The actual implementation of the Scrum methodology will be bespoke.

Design patterns have been used in object-oriented programming for some time. But in 1994, they were officially classified and described in a book called *Design Patterns: Elements of Reusable Object-Oriented Software*, which was written by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, who are collectively known as *The Gang of Four*.

So, in a nutshell, each design pattern prescribes how the objects in your code should be structured and how they should behave. For solutions where multiple objects are involved, it prescribes the role for each object.

Let's have a look at a design pattern called *Strategy* as our example. This pattern is commonly used to make your code easily maintainable and testable in situations where you have complex conditional logic. In this design pattern, you have a class with the role of *Strategy*. This class stores an interface, which has the role of *Context*. There are several classes that implement this interface, each performing a specific behavior. Then, all you do inside your conditional logic is assign a specific *Context* implementation to the *Strategy*. And then, once all conditions have been examined, you just execute some method on the *Context* object which *Strategy* class encapsulates. This will execute the behavior that was specific

to a particular condition, as a specific implementation of *Context* was assigned to *Strategy* when a specific condition was hit.

So, the design patterns prescribe roles to objects and how these objects should interact with one another. We have one object with the role of *strategy* and several implementations of an object with the role of *Context*. But what the design pattern doesn't prescribe is what specific behavior should the implementation perform or what data type(s) should it return, if any. It's up to us to decide.

Some design patterns don't even involve multiple objects with different roles. *Singleton* and *Prototype* patterns, for example, prescribe how to apply internal logic to an object and what public members should this object have for other objects to access.

So, why would you use design patterns? Why just not provide a bespoke solution to every problem you face? Well, there are some good reasons why design patterns are better. But the reasons why you would want to learn them go beyond the fact that design patterns merely provide a better solution.

Why would you want to learn design patterns

Design patterns will indeed make you more effective at solving problems. When you have used them for a while, you will intuitively know where to apply them. So, you will not have to spend much time thinking of the solution. You will be able to apply one right away.

Not only will you be able to solve the problem quicker, but your code will be cleaner. It will be more readable, easier to test, and easier to maintain.

Let's go back to our example of *Strategy* pattern. You can, of course, just use complex conditional logic in a single method of your code.

But then imagine how hard it might be to test the logic. Also, if the logic is complex, it might not necessarily be the most readable code.

When *Strategy* pattern is implemented, it solves the problem. But beyond that, it makes your code clean. The original method that contains the conditional logic only selects which *Strategy* to implement. It doesn't do anything beyond that. Easy to read. Easy to test. Each *Strategy* implementation only contains an atomic piece of logic that is responsible for only one action. Easy to read. Easy to test.

However, the benefits of knowing design patterns go beyond them being good solutions to common problems. Many reputable organizations ask questions about them during job interviews. The major IT giants almost certainly do. So, not knowing design patterns might prevent you from getting a job in the company of your dreams.

The main problem with design patterns is that they are not necessarily easy to learn. Let's examine why that is.

Why design patterns are hard to learn

Many developers, especially the ones who don't have a lot of software-building experience, struggle with design patterns. But if you do struggle with them, it may prevent you from getting a programming job at a reputable organization. After all, recruiting managers often ask questions about design patterns. Otherwise, not knowing design patterns will make you less effective as a software developer, which will slow down your career progress.

The main reason why design patterns are so hard to learn is because of the way they are normally taught. Usually, if you pick pretty

much any book on design patterns or open pretty much any online article about them, it would provide a collection of design patterns that you would need to go through. You would then have to go through each of them, try your best to understand the principles behind it, and only then try to figure out how to apply it in a real-life situation.

It's a tedious process that doesn't always bring about the right results. It's not uncommon for software developers to memorize just a handful of design patterns that they have been using in their own projects. The remaining ones have been forgotten as soon as they've been learned. And it's hard to figure out which design pattern applies in which situation if you only remember a handful of them.

The goal of this book

This book provides a different approach. It uses a methodology that makes it easy to learn design patterns. So, you no longer have to brute-force your way through them. The process of effective learning is not about memorization. It's about associations. You learn new things easily when you can clearly see how new facts relate to your existing knowledge. And this is precisely the method that this book is built around.

You won't have to brute-force your way into design patterns. In fact, you won't even start with the design patterns. First, we will go through a list of common problems that software developers are required to solve. Those are the things that every software developer can associate with. Even if you haven't faced a particular type of problem yet, you will still be able to easily understand its description. For each of these problems, we will go through the design patterns that can solve it. For each one of them, you will go through its core principle and the description of how it can solve this type of problem. Only then you will be invited to examine this

particular design pattern in detail, so you can understand how to implement it in your own code.

This structure of the book also makes it valuable as a reference book. Even when you don't know or don't remember design patterns, looking them up becomes easy. What you need to find is a description of the type of problem you are trying to solve. And then you will be able to follow it to find the actual design patterns that you can apply to solve it.

Therefore this book is not only an effective learning tool. It's also a reference book that's incredibly easy to navigate. It's been structured in such a way that you'll be able to find the right answer in seconds.

One important thing to note is that this book doesn't aim to cover absolutely all design patterns that exist out there. It focuses on the classic design patterns that were outlined by *The Gang of Four*. So some of the design patterns that you have heard of might be missing.

However, these classic design patterns formed the foundation for all design patterns that came after. Therefore, if you get familiar with them, you will have no trouble learning any newer patterns.

The structure of this book

This book consists of three parts. It has been structured to facilitate easy learning of design patterns for those who aren't familiar with them.

Part 1: SOLID principles and why they are important

SOLID principles are the most fundamental rules of clean code in object-oriented programming. All design patterns depend on these

principles. This is why, in order to be able to understand design patterns, you need to be familiar with SOLID principles. This is why this is the very first thing that you will learn in this book.

Part 2: The problems that design patterns are intended to solve

This part lists various problem types that software developers commonly face. For each of these problems, it provides a list of design patterns that can solve it.

This part of the book doesn't provide an in-depth description of each of the patterns. It merely provides a brief overview of what each design pattern does and how it can solve a specific type of problem.

This part of the book has been structured in such a way to ensure that those developers who aren't yet familiar with design patterns can easily look them up and be able to quickly find the most suitable design pattern for a specific type of problem they are facing.

Part 3: Design patterns demonstrated in C#

The final part of the book breaks down each of the design patterns in depth. It uses C# code samples to demonstrate how you can implement each of the patterns in your own code.

About the author and his mission

My name is Fiodar Sazanavets. I am a senior full-stack software engineer with over a decade of experience. I know how hard it is to become a competent software developer. I've been through this process myself. And I'm passionate about helping other people, so the journey would be easier for them than it was for me.

Design patterns were the subject that I found quite difficult to comprehend when I was a mid-level developer. At that point in time, I had heard about the design patterns. I kind of knew that they would make code cleaner and more maintainable. But there was one problem. I found them to be hard to learn.

No, there wasn't a lack of documentation. On the contrary, there are plenty of books and blog posts that describe what design patterns are and provide code samples of their implementations. But the problem with all of this material is that the way design patterns are usually taught makes them really hard to understand,

especially for those developers who have never used them before. If you are a novice and you examine any of the design patterns outside the context that it's used under, it will probably look like the code was over-complicated without any obvious reason. And that's precisely what made it so hard for me to learn the patterns. After all, I have been coding quite happily without using any of them. So I have abandoned my attempts to learn them.

But it all changed one day when I had a technical interview at one of the major IT companies in London. The company was great and the position that I was interviewing for was paying way above the market rate. But I ended up not getting this job. And my lack of knowledge of the design patterns was to blame.

During the technical interview, I was presented with a problem that one of the design patterns could have solved in minutes. In fact, this problem was specifically designed to assess whether or not a candidate knows design patterns. Because I didn't know them, it was taking me a long time to come up with a solution. And I didn't manage to fully implement the solution within the allocated time. So, essentially, I failed for two reasons:

1. I didn't solve the problem quickly enough
2. I didn't know design patterns

Later, I found out that this wasn't just an isolated case. There are plenty of good IT companies that won't hire you if you don't know design patterns. So I got back to trying to learn them.

They were still hard to learn. I had to brute-force my way through them. But I managed to learn them all. Then I started to use them in my job. And this was when I realized how underappreciated they are.

Design patterns aren't needed only to get your foot into the door of a good IT company. A lot of problems that you encounter as a software developer can be solved relatively quickly if you know

design patterns. You will also understand open-source code better, as plenty of popular repositories use design patterns.

This experience was what inspired me to write this book. My mission is to help as many people as possible with the process of learning design patterns. But I don't want them to brute-force their way through them like I once did. I want to make the process of learning them as smooth for the reader as possible.

I'm doing it by providing sufficient content first before showing the implementation of each of the design patterns in detail. After all, effective learning is not about trying to memorize things. Effective learning is about building associations between the stuff you already know and the new information you are trying to internalize. Therefore it would be hard to learn design patterns if you just start looking at them without sufficient knowledge of the context which they are used under.

Without the context, you won't be able to build mental associations. You won't be able to fully understand what each of the design patterns is used for. So you will have to fall back to trying to memorize code structures that would probably make little sense to you. And that is precisely what makes design patterns seem so hard to learn.

I sincerely hope that you will find this book useful. All the best!

As well as being an author of this book, I have written a number of other technical books and online courses. I regularly write about software development on my personal website, https://scientificprogrammer.net. I am also available to book as a personal programming mentor via https://mentorcruise.com/mentor/fiodarsazanavets.

Getting in touch

If you want to get in touch with me regarding the content of the book, you can contact me via either Twitter or LinkedIn. Perhaps, there is additional content that you want to have included in the next edition of the book. Or maybe you found some errors in the current edition. Any feedback is welcome. And this is how you can get in touch:

Twitter: https://twitter.com/FSazanavets

LinkedIn: https://www.linkedin.com/in/fiodar-sazanavets/

Helping to spread the word

If you found this book useful, please write a review about it on Amazon. This will help the book to be discoverable by more people and will help me, as an author, to produce more quality educational content in the future.

And, of course, if you liked this book, spread the word. Let your friends know about it. This way, you will help more people to become better software developers.

UML conventions used in the book

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Basic component

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Association

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Aggregation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Composition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Dependency

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Part 1: SOLID principles and why they are important

Many design patterns were developed with SOLID principles in mind. Therefore it will be hard to learn design patterns unless you know SOLID principles. This is why the first part of the book will explain what these principles are and will provide some examples of their implementation in C#.

But SOLID principles aren't merely a tool to help you learn design patterns. Every software developer who uses object-oriented programming languages needs to be familiar with SOLID principles and know how to apply them in their daily job. Being familiar with these principles will make you more effective in your job, which will help you to progress in your career. But another thing to remember is that questions about SOLID principles are often asked during technical interviews. So, if you don't know how to apply them, it may prevent you from getting the job that you want.

For those who aren't familiar with what these principles are, SOLID is an abbreviation that stands for the following:

- Single responsibility principle
- Open-closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

Do you need to always use SOLID principles? Maybe not. After all, principles are not laws. There might be some exceptions where your

code would actually be more readable or maintainable if you don't apply them. But to understand whether or not it makes sense not to apply them in any given situation, you first need to understand them. And this is what this part of this book will help you with.

But the important part is that you will struggle to fully understand design patterns unless you understand SOLID principles. SOLID principles are your foundation. Unlike design patterns, they don't apply to any specific type of software development problem. They are fundamental components of clean code. And this is why we will start with them.

1. Single responsibility principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

What is single responsibility principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

The importance of the single responsibility principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

The concept of class cohesion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

God Object – the opposite of single responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

2. Open-closed principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

What is open-closed principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Implementing the open-closed principle in C#

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

3. Liskov substitution principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

What is Liskov substitution principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Implementing Liskov substitution principle in C#

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

4. Interface segregation principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

What is interface segregation principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Importance of interface segregation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

When NotImplementedException is appropriate

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

5. Dependency inversion principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

What is dependency inversion principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why the dependency inversion principle is important

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Dependency inversion is not only useful in unit tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-

learn-design-patterns.

Part 2: The problems that design patterns are intended to solve

This part of the book lists common types of software problems that software developers face. For each of these problems, any suitable design patterns are listed.

In this part of the book, we don't go into each design pattern in detail. There is a brief description of how it's structured, what it does, and why it can solve a specific type of problem. Then, at the end of this part, we summarize all the problems that we have covered and, for each of them, we summarize the pros and cons of each design pattern that can solve it.

This part of the book serves two purposes. Firstly, it's much easier to learn by association. Software problems are something that every software developer can easily imagine. Some of these problems would be something that they are personally familiar with. Design patterns, on the other hand, are something that won't make a lot of sense right away if you are new to them. This is why we start with the descriptions of the problems first before we learn the actual design patterns.

Secondly, this structure makes it easy to look up suitable design patterns for those who don't know them yet. If you, as a programmer, are facing a challenging problem at work, the last thing that you would want to do is go through the descriptions of all design patterns and try to figure out which ones can help you solve the problem at hand. What you would want to do is find the description of the problem that you are facing and then use it to look up any suitable design patterns. If there is more than one, you would want

to know the pros and cons of each to assess which one of them fits your specific situation better. Only then you would want to know how to actually implement it. And this is what this part of the book facilitates.

So, use this section of the book both as a learning guide and as a reference source.

6. Not knowing what object implementations you'll need ahead of time

Imagine that you have a code that uses a particular object type. The code that uses this object only needs to know the signatures of accessible methods and properties of the object. It doesn't care about the implementation details. Therefore, using an interface that represents the object rather than a concrete implementation of the actual object is desirable.

If you know what concrete implementation your application will be using ahead of time, you can simply apply the dependency inversion principle along with some simple dependency injection. In this case, you will know that it's always a particular implementation of the interface that will be used throughout your working code and you will only replace it with mocked objects for unit testing. But what if the concrete implementation of the interface needs to be chosen dynamically based on runtime conditions? Or what if the shape of the object needs to be defined based on input parameters in the actual place where the object is about to be used? In either of these cases, the dependency injection approach wouldn't work.

Let's look at an example of this. Imagine that you are building an application that can play audio. The application should be able to work on both Windows and Linux. And inside it, you have an interface called `IPlayer` that has standard methods that an audio player would have, such as `Play`, `Pause`, and `Stop`. It is the implementation of this interface that actually interacts with the operating system and plays the audio.

The problem is that Windows and Linux have completely different

audio architectures; therefore you cannot just have a single concrete implementation of the `IPlayer` interface that would work on both operating systems. And you won't know ahead of time what operating system your application will run on.

Suitable design patterns

Factory Method

Factory Method is a method that returns a type of object that implements a particular interface. This method belongs to a Creator object that manages the lifecycle of the object that is to be returned.

Normally, you would have several variations of the Creator object, each returning a specific implementation of the object that you want. You would instantiate a specific variation of the Creator object based on a specific condition and would then call its Factory Method to obtain the implementation of the actual object.

In our example above, you would have one version of the Creator object that returns a Windows implementation of `IPlayer` and another version that returns its Linux implementation. The Creator object will also be responsible for initializing all dependencies that your `IPlayer` implementation needs. Some code blocks in your application will check which operating system it runs on and will initialize the corresponding version of the Creator object.

Why would you want to use Factory Method

Why bother using Factory Method, if you can just initialize any objects directly? Well, here is why:

- Good use of the single responsibility principle. The Creator object is solely responsible for creating only one specific implementation type of the end object and nothing else.

- The pattern has been prescribed in such a way that it makes it easy to extend the functionality of the output object and not violate the open-closed principle.
- Easy to write unit tests, as Creational logic will be separated from the conditional logic.

Abstract Factory

Abstract Factory is a design pattern that uses multiple Factory Methods inside of the Creator object, so you can create a whole family of related objects based on a particular condition instead of just one object.

Let's change the above example slightly. Imagine that our app needs to be able to either play audio or video, so you will need to implement two separate interfaces – `IAudioPlayer` and `IVideoPlayer`. Once again, it must work on either Linux or Windows.

In this case, your Abstract Factory will have a separate method to return an implementation of `IAudioPlayer` and a separate method to return an implementation of `IVideoPlayer`. You will have a version of the Factory that is specific to Windows and another version that is specific to Linux.

It's known as Abstract Factory because it either implements an interface or extends an abstract class. It is then up to the concrete implementation of the Factory to create concrete implementations of the output objects that are both relevant to a specific condition.

Builder

Builder design pattern is similar to the Factory Method, but instead of just returning a concrete implementation of an object all at once, it builds the object step-by-step.

Let's go back to our OS-independent app that plays audio. In the Factory Method example, we had a concrete implementation of the `IPlayer` interface for each operating system. However, if we would choose to use Builder instead of a Factory Method, we would have a single concrete implementation of the interface that would act as a shell object. Let's call it `Player`. It will be this type that gets produced in all scenarios, but the concrete parameters and dependencies will be injected into it based on what kind of operating system it's running on.

For example, both Linux and Windows allow you to play audio and manipulate its volume via the command line. On Linux, it will be Bash Terminal. On Windows, it will be either cmd or PowerShell.

The principles are similar. In both cases, you would be typing commands. But the exact commands will be completely different. Plus there are likely to be separate commands for playing audio files and for manipulating audio volumes.

So, in this case, our `Player` class will simply delegate the playback of audio to operating system components that are accessible via a command line interface. It's only the actual commands that will be different. And this is where a Builder design pattern comes into play.

Builder consists of two main components – Builder and Director. Builder is a class that returns a specific object type. But it also has a number of methods that modify this object type before it gets returned. The director is a class that has a number of methods, each of which accepts a Builder class, calls methods on it with a specific set of parameters, and gets the Builder to return the finished object.

So, in our case, imagine that our Builder class for the `Player` object (which we will call `PlayerBuilder`) has the following methods: `SetPlaybackInterface` (that accepts a playback interface instance), `SetAudioVolumeInterface`(that accepts audio volume interface instance) and `BuildPlayer` (that doesn't accept any parameters). When we instantiate our `PlayerBuilder` class, we instantiate a

private instance of the `Player` class inside of it. Then, by executing `SetPlaybackInterface` and `SetAudioVolumeInterface` methods, we dynamically add the required dependencies to the instance of our `Player` class. And finally, by executing the `BuildPlayer` method, we are returning a complete instance of a `Player` object.

In this case, our Director class will have two methods, both of which accept `PlayerBuilder` as the parameter: `BuildWindowsPlayer` and `BuildLinuxPlayer`. Both of these methods will call all the methods on the `PlayerBuilder` class in the same order and both will return an instance of a `Player` class. But in the first case, the methods will be called with Windows-specific abstractions of the command line interface, it's the Linux-specific abstractions that would be injected into the `Player` instance.

However, unlike either Abstract Factory or Factory Method, Builder is not only used to build an object the implementation details of which can only be made known at runtime. It is also used to gradually build complex objects.

For example, .NET has an inbuilt `StringBuilder` class in its core `System` library. This class is used for building a string from several sub-strings, so you don't have to just keep replacing an immutable string in the same variable.

Why would you want to use Builder

- Good use of the single responsibility principle. Each Builder method has its own very specific role and there is a single method on the Director class per each condition.
- Easy to write unit tests. This is facilitated by the single responsibility principle.
- No need to have different versions of a class if only some of its implementation details may change in different circumstances.

Because an object can be built step-by-step, it is the best design

pattern to decide what the object will be if you have to adjust its properties one by one by multiple steps of conditional logic.

You can reuse the same code for different representations of the final object.

7. Making several exact copies of a complex object

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prototype

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you want to use Prototype

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

8. Using many instances of an object while keeping code running smoothly

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Object Pool

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you want to use Object Pool

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Flyweight

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you want to use Flyweight

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prototype

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Prototype alongside Object Pool

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

9. Using the same single instance of an object throughout the application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Singleton

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you want to use Singleton

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Object Pool

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Object Pool as Singleton

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

10. Third-party components aren't directly compatible with your code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Adapter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Adapter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

11. Adding new functionality to existing objects that cannot be modified

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Decorator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Decorator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Accessing complex back-end logic from the presentation layer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

13. User interface and business logic are developed separately

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Facade instead of Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Proxy instead of Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

14. Building a complex object hierarchy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Composite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Composite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

15. Implementing complex conditional logic

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Factory Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-

learn-design-patterns.

The differences between Factory Method and Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Abstract factory

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

16. Multiple object instances of different types need to be able to communicate with each other

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Mediator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Mediator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-

learn-design-patterns.

Observer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Observer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

17. Multiple stages of processing are needed

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Chain of Responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use a Chain of Responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Builder

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-

learn-design-patterns.

Why would you use Builder instead of Chain of Responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

18. The system is controlled by complex combinations of inputs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

19. Ability to undo an action that has been applied

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Command instead of Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

20. Ability to traverse a collection without knowing its underlying structure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Iterator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would you use Iterator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

21. Creating a family of related algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Suitable design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Template Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would use Template Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Visitor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-

learn-design-patterns.

Why would you want to use Visitor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

State

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Why would use State

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

22. Summary of the problems design patterns are intended to solve

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Not knowing what object implementations you'll need ahead of time

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Making several exact copies of a complex object

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Using many instances of an object while keeping code running smoothly

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Using the same single instance of an object throughout the application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Third-party components aren't directly compatible with your code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Adding new functionality to existing objects that cannot be modified

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Accessing complex back-end logic from the presentation layer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

User interface and business logic are developed separately

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Building a complex object hierarchy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Implementing complex conditional logic

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Multiple object instances of different types need to be able to communicate with each other

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-

learn-design-patterns.

Multiple stages of processing are needed

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

The system is controlled by complex combinations of inputs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Ability to undo an action that has been applied

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Ability to traverse a collection without knowing its underlying structure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Creating a family of related algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Part 3: Design patterns demonstrated in C#

The third and final part of the book will teach you each of the design patterns in detail. You will learn the basic structure of each of the patterns. Then you will be provided with an example of its implementation.

All code samples are provided in C#. We use the .NET 6 template style.

None of the code is copyrighted, so please feel free to use it and modify it as you please.

23. Design pattern categories

The classic design patterns can be split into three distinct categories: **Creational**, **Structural**, and **Behavioral**. And the design patterns in each category do exactly what the name of the category suggests.

Creational design patterns define the ways of creating objects. The following patterns belong in this category:

- Factory Method
- Abstract Factory
- Builder
- Prototype
- Singleton

Structural design patterns prescribe how to structure your objects. These are the patterns that belong in this category:

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

Behavioral design patterns tell us how objects are supposed to behave. This category consists of the following patterns:

- Chain of Responsibility

- Command
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

This section of the book is split into sub-sections. Each of these covers a specific category of design patterns.

Creational design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

24. Factory Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Factory Method implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Factory Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Factory Method

25. Abstract Factory

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Abstract Factory implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Abstract Factory

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Abstract Factory

26. Builder

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Builder implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Builder

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Builder

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

27. Prototype

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prototype implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Prototype

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Prototype

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

28. Singleton

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Singleton implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Singleton

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Singleton

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Structural design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

29. Adapter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Adapter implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Adapter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Adapter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

30. Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Bridge implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Bridge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

31. Composite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Composite implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Composite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Composite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

32. Decorator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Decorator implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Decorator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Composite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

33. Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Facade implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

34. Flyweight

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Flyweight implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Flyweight

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Flyweight

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

35. Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Proxy implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Behavioral design patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

36. Chain of Responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Chain of Responsibility implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Chain of Responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Chain of Responsibility

37. Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Command implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

38. Iterator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Iterator implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Iterator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Iterator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

39. Mediator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Mediator implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Mediator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Mediator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

40. Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Memento implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

41. Observer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Observer implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Observer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Observer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

42. State

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

State implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using State

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using State

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

43. Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Strategy implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

44. Template Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Template Method implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Template Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Template Method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

45. Visitor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Prerequisites

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Visitor implementation example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Benefits of using Visitor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Caveats of using Visitor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.

Epilogue

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/the-easiest-way-to-learn-design-patterns.