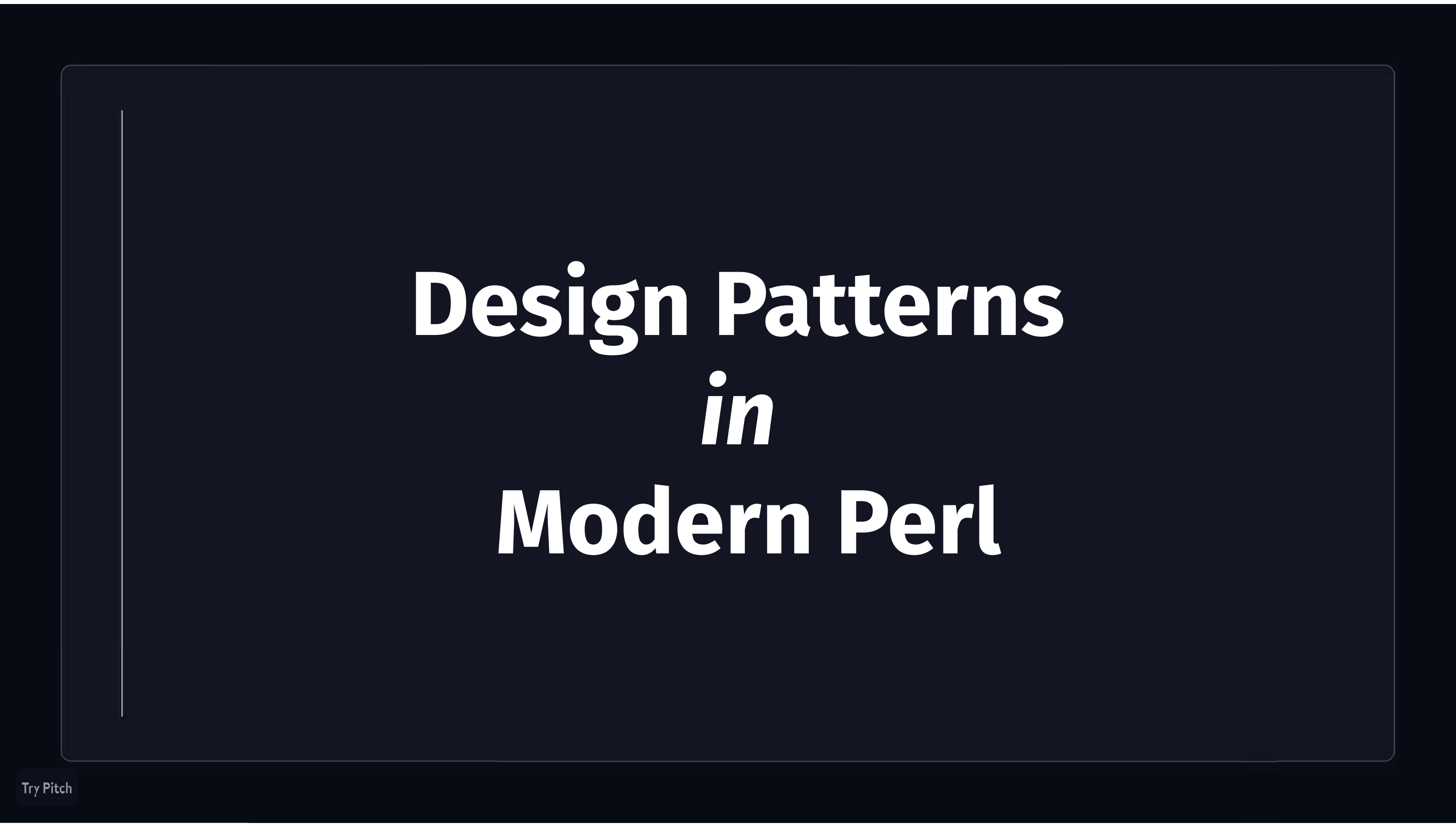
Mohammad Sajid Anwar

2022 WHITE CAMEL AWARDEE

https://manwar.org

https://github.com/manwar

https://theweeklychallenge.org



Design Patterns *in* Modern Perl

Modern Perl?

v5.38 - Jul 2023

v5.40 - Jun 2024

v5.42 - Jul 2025

Latest Release

v5.43.5 - Nov 2025

https://metacpan.org/release/CONTRA/perl-5.43.5/view/pod/perldelta.pod

Example experimental named parameters

```
#!/usr/bin/env perl

use v5.43;
use experimental 'signature_named_parameters';

sub hello(:$name = "Bob") {
    return "Hello $name!!";
}

say hello();                    # Hello Bob!!
say hello(name => "Joe");   # Hello Joe!!
```

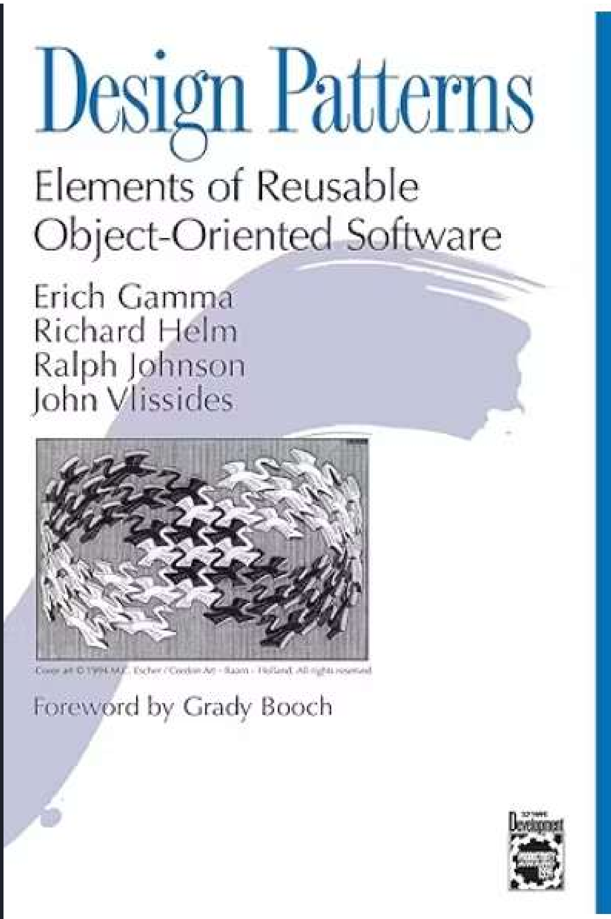
Try Pitch

2022

https://github.com/manwar/perl-cool-snippets

— 45 stars on GitHub —

Gang of Four Book


Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

Try Pitch

Creational Patterns	Structural Patterns	Behavioural Patterns
Abstract Factory	Adapter	Chain of Responsibility
Builder	Bridge	Command
Factory Method	Composite	Interpreter
Prototype	Filter	Iterator
Singleton	Decorator	Mediator
	Facade	Memento
	Proxy	Observer
		State
		Strategy
		Template
		Visitor

2021

https://github.com/manwar/Design-Patterns

— 57 stars on GitHub —

Main Features
Used Moo as base OOP framework
Implemented 17 out of 23 design patterns (pure code)

What is missing?
Raw bless-based class implementation
Also Object::Pad implementation

Sep 2025

https://theweeklychallenge.org/blog/design-pattern-factory

[Moo, Object::Pad, experimental class]

OOP Framework	Inheritance	Role/Interface
Raw bless-based class	✔	✗
Moo	✔	✔
Object::Pad	✔	✔
Experimental class feature	✔	✗

Design Patterns without Role/Interface
Prototype
Singleton
Composite
Memento

What's the blocker?

Missing support for role in raw bless-based class

Also in the experimental class feature (v5.42+)

What's the solution?

Add support for role in raw bless-based class

How about the same in experimental class feature (v5.42+) ?
(beyond my capacity)

Fallback to Object::Pad, easy choice.

Class::Mite

https://github.com/manwar/Class-Mite

Class	Role	Class::Clone	Class::More

Try Pitch

Inheritance using Class from Class::Mite

```
package Parent;

use Class;

sub location {
    my ($self) = @_;
    return $self->{name}, " lives in London!\n";
}

package Child;

use Class;
extends qw/Parent/;

package main;
print Child->new(name => 'Tom')->location;
```

Interface using Role from Class::Mite

```
package Animal;

use Role;
requires qw/speak/;

package Dog;

use Class;
with qw/Animal/;

sub speak {
   my ($self) = @_;
   return $self->{name}, " bark!\n";
}

package main;
print Dog->new(name => 'Tommy')->speak;
```

Comparative Analysis

https://theweeklychallenge.org/blog/bless-vs-class-mite

Singleton Design Pattern

Why?

No role needed.

Single class is enough for demo.

Singleton Design Pattern using raw bless

```
package Singleton;

our $INSTANCE;
sub instance {  $INSTANCE //= bless { count => 0 }, __PACKAGE__; }
sub counter  {   ++shift->{count};                                  }


package main;
print Singleton->instance->counter; # 1
print Singleton->instance->counter; # 2
print Singleton->instance->counter; # 3
```

Try Pitch

Singleton Design Pattern using Class from Class::Mite

```
package Singleton;

use Class;
my $instance;
sub BUILD     { shift->{count} //= 0                 }
sub instance { $instance //= __PACKAGE__->new  }
sub counter   { ++shift->{count}                     }

package main;

print Singleton->instance->counter; # 1
print Singleton->instance->counter; # 2
print Singleton->instance->counter; # 3
```

Try Pitch

Singleton Design Pattern using Moo and MooX::Singleton

```
package Singleton;

use Moo;
with qw/MooX::Singleton/;

has 'count' => (is => 'rw', default => sub { 0 });

sub counter($self) {
    $self->count($self->count + 1)
}

package main;

print Singleton->instance->counter; # 1
print Singleton->instance->counter; # 2
print Singleton->instance->counter; # 3
```

Singleton Design Pattern using experimental class feature

```
use v5.42;
use experimental qw/class/;

class Singleton {
    field $count = 0;
    state $instance;

    sub instance       { $instance //= __PACKAGE__->new }
    method counter { ++$count                                   }
}

package main;

print Singleton->instance->counter; # 1
print Singleton->instance->counter; # 2
print Singleton->instance->counter; # 3
```

Try Pitch

Singleton Design Pattern using Object::Pad

```
use Object::Pad;

class Singleton {
    my $instance;
    field $count :reader :writer = 0;

    method instance :common {
        $instance //= __PACKAGE__->new;
    }

    method counter {
        $self->set_count($self->count + 1);
        return $self->count;
    }
}

package main;
print Singleton->instance->counter; # 1
print Singleton->instance->counter; # 2
print Singleton->instance->counter; # 3
```

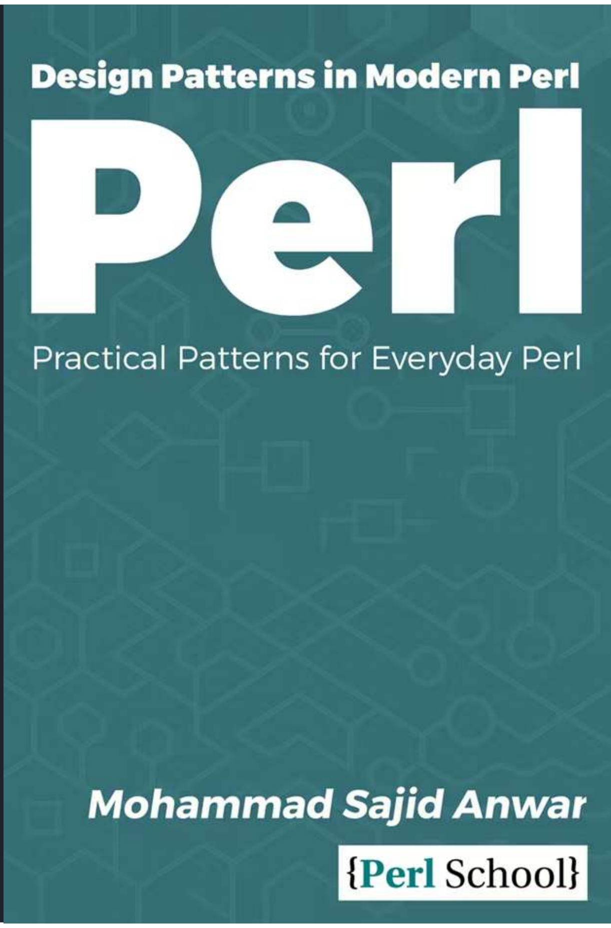
Try Pitch

Personal Blogs

https://theweeklychallenge.org/blogs

29th Nov 2025
Buy on Amazon / LeanPub

https://perlschool.com/books/design-patterns


Design Patterns in Modern Perl
Perl
Practical Patterns for Everyday Perl
Mohammad Sajid Anwar
{Perl School}

Thank You

Organiser: Andrew Mehta and JJ Atria

Gold Sponsor: https://perlfoundation.org

Bronze Sponsor: https://www.simplelists.com