ТЕХНОСФЕРА

ООП

Содержание

bless

```
{
  package Class::Name;
  #...
}

my $obj = bless {}, 'Package::Name';

my $obj2 = bless [], '...';
my $scalar = 42;
my $obj2 = bless \$scalar, '...';
```

Содержание

Методы

```
{
  package A;

  sub set_a {
    my ($self, $value) = @_;
    $self->{a} = $value;
    return;
  }

  sub get_a {
    my ($self) = @_;
    return $self->{a};
  }
}

my $obj = bless {}, 'A';

$obj->set_a(42);
print $obj->get_a(); # 42
```

Атрибуты

```
{
  package A;
  sub new {
    my ($class, %params) = @_;

    return bless \%params, $class;
  }
}

my $obj = A->new(a => 1, b => 2);

print $obj->{a}; # 1
print $obj->{b}; # 2
```

Методы класса

```
{
  package A;

  sub new {
    my ($class, %params) = @_;

    return bless \%params, $class;
  }

  sub get_a {
    my ($self) = @_;

    return $self->{a};
  }
}

my $obj = A->new(a => 42);
$obj->get_a(); # 42
```

Методы — еще варианты

```
$obj->A::get_a();

my $class = 'A';
$class->new();

my $method_name = $cond ? 'get_a' : 'get_b';
$obj->$method_name;

A::new(); # not the same!
```

Методы — indirect

```
new My::Class(1, 2, 3);

My::Class->new(1, 2, 3);
```

```
foo $obj(123); # $obj->foo(123);
```

```
use strict;
use warnings;

Syntax error!

exit 0;
```

Методы — WHY!?

```
use strict;
use warnings;

Syntax error!

exit 0;
```

```
use warnings;
use strict;
'error'->Syntax(!exit(0));
```

can

```
{
  package A;

  sub test {
    return 42;
  }
}

if (A->can('test')) {
  print A->test;
}

print A->can('test')->('A');
```

```
my $obj = bless {}, 'A';
$obj->can('test');
```

Filehandles

```
open(my $fh, '>', 'path/to/file');

$fh->autoflush();
$fh->print('content');

STDOUT->autoflush();
```

Пакеты

```
use Some::Package qw(a b c);
# Some::Package->import(qw(a b c));

no Some::Package;
# Some::Package->unimport;

use Some::Package 10.01
# Some::Package->VERSION(10.01);
```

Итого

```
my $obj = Class->new();

$obj->do_something();
print $obj->{attr};
```

```
$obj->can('method');
$fh->print(123);
$obj = new Class(); # :(
```

Содержание

DBI

```
$dbh = DBI->connect(
  $data_source,
  $username,
  $auth,
  \%attr
);

$rv = $dbh->do('DELETE FROM table');
```

XML::LibXML;

```
use XML::LibXML;

my $document = XML::LibXML->load_xml(
    string => '...'
);

my $list = $document->findnodes('...');
    # XML::LibXML::NodeList
```

```
        XML::LibXML::Node
       /                  \
 XML::LibXML::Document   XML::LibXML::Element
```

File::Spec

```
use File::Spec;

print File::Spec->catfile('a', 'b', 'c');
```

JSON

```
use JSON;

JSON->new->utf8->decode('...');

decode_json '...';
```

Содержание

Наследование

```
{
  package Lynx;

  BEGIN { push(@ISA, 'Dog', 'Cat') }
  use base qw(Dog Cat);
  use parent qw(Dog Cat);
}
```

UNIVERSAL

```
$obj->can('method');

$obj->isa('Animal');
Dog->isa('Animal');

$obj->VERSION(5.12);
```

SUPER

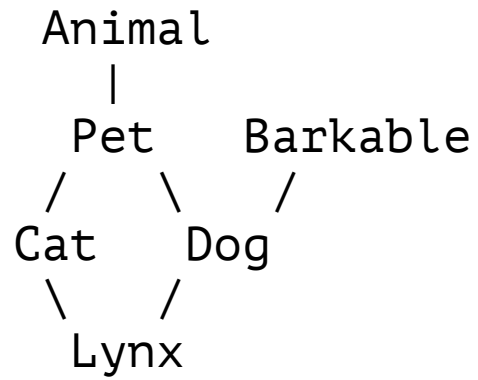
```
sub method {
  my ($self, %params) = @_;

  $self->SUPER::method(%params);

  return;
}
```

Содержание

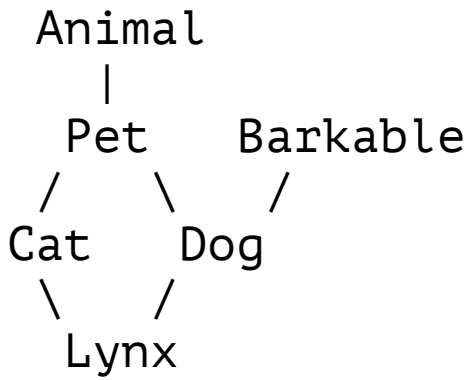
Method Resolution Order




```
Lynx->method();

qw(Lynx Cat Pet Animal Dog Barkable);
```

use mro;




```
use mro 'c3';

Lynx->method();

qw(Lynx Cat Dog Pet Animal Barkable);
```

mro — next::method

```
package A;
use mro;

sub foo {
  my ($self, $param) = @_;

  $param++;

  return $obj->next::method($param);
}
```

Содержание

blessed, ref

```
use JSON:
use Scalar::Util 'blessed';

ref JSON->new(); # 'JSON'
ref [];          # 'ARRAY'
ref {};          # 'HASH'
ref 0;           # ''

blessed JSON->new(); # 'JSON'
blessed [];          # undef
blessed {};          # undef
blessed 0;           # undef
```

DESTROY

```
package A;
sub new {
  my ($class, %params) = @_;
  return bless \%params, $class;
}

sub DESTROY {
  print 'D';
}
```

```
A->new(); # print 'D'
```

DESTROY — сложности

- `die`
- `local`
- `AUTOLOAD`
- `${^GLOBAL_PHASE} eq 'DESTRUCT'`

```
sub DESTROY {
  my ($self) = @_;
  $self->{handle}->close() if $self->{handle};
}
```

AUTOLOAD

```
package A;
our $AUTOLOAD;
sub new {
  my ($class, %params) = @_;
  return bless \%params, $class;
}
sub AUTOLOAD { print $AUTOLOAD }
```

```
A->new()->test(); # test
A->can('anything'); # :(
```

```
sub UNIVERSAL::AUTOLOAD {}

# Dog->m(); Animal->m(); UNIVERSAL->m();
# Dog->AUTOLOAD(); Animal->AUTOLOAD();
# UNIVERSAL->AUTOLOAD();
```

Исключения

```
eval {
  die Local::Exception->new();
  1;
} or do {
  my $error = $@;

  if (
    blessed($error) &&
    $error->isa('Local::Exception')
  ) {
     # ...
  }
  else {
    die $error;
  }
};
```

Исключения — модули

```
use Try::Tiny;
try {
  die 'foo';
}
catch {
  warn "caught error: $_"; # not $@
};
```

```
use Error qw(:try);
try {
  throw Error::Simple 'Oops!';
}
catch Error::Simple with { say 'Simple' }
catch Error::IO with     { say 'IO' }
except                   { say 'Except' }
otherwise                { say 'Otherwise' }
finally                  { say 'Finally' };
```

???

```
$hash{x} = 7;

print $hash{x};
```

42

tie

```
package Local::MyHash;

use Tie::Hash;
use base 'Tie::StdHash';

sub FETCH { 42 }
```

```
my %hash;
tie %hash, 'Local::MyHash';

$hash{x} = 7;

print $hash{x};
```

overload

```
my $x = Local::Int->new(42);
my $y = Local::Int->new(24);

print(($x + $y)->{value}); # 66
```

overload

```
package Local::Int;

use overload '+' => 'add';

sub new {
    my ($class, $value) = @_;
    return bless {value => $value}, $class;
}

sub add {
    my ($self, $other) = @_;

    return __PACKAGE__->new(
        $self->{value} + $other->{value}
    );
}
```

Class::Accessor

```
package Foo;
use base qw(Class::Accessor);
Foo->follow_best_practice;
Foo->mk_accessors(qw(name role salary));
```

```
use base qw(Class::Accessor::Fast);
use base qw(Class::XSAccessor);
```

Итого

```
ref($obj);
blessed($obj);

{ A->new() }
```

- try
- tie
- overload
- AUTOLOAD

Содержание

Moose

```
package Person;

use Moose;

has first_name => (
  is  => 'rw',
  isa => 'Str',
);

has last_name => (
  is  => 'rw',
  isa => 'Str',
);
```

```
Person->new(
  first_name => 'Vadim',
  last_name  => 'Pushtaev',
);
```

Moose — наследование

```
package User;

use Moose;

extends 'Person';

has password => (
  is => 'ro',
  isa => 'Str',
);
```

Moose — инициализация

BUILD

```
has age      => (is => 'ro', isa => 'Int');
has is_adult => (is => 'ro', isa => 'Bool');

sub BUILD {
  my ($self) = @_;

  $self->is_adult($self->age >= 18);

  return;
}
```

Moose — инициализация

default

```
has age      => (is => 'ro', isa => 'Int');
has is_adult => (
  is => 'ro',
  isa => 'Bool',
  lazy => 1,
  default => sub {
    my ($self) = @_;
    return $self->age >= 18;
  }
);
```

Moose — инициализация

builder

```
has age       => (is => 'ro', isa => 'Int');
has is_adult => (
  is => 'ro', isa => 'Bool',
  lazy => 1,  builder => '_build_is_adult',
);

sub _build_is_adult {
  my ($self) = @_;
  return $self->age >= 18;
}
```

```
package SuperMan;
extends 'Person';
sub _build_is_adult { return 1; }
```

Moose — инициализация

Цепочки

```
has [qw(
  file_name
  fh
  file_content
  xml_document
  data
)] => (
  lazy_build => 1,
  # ...
);

sub _build_fh           { open(file_name) }
sub _build_file_content { read(fh) }
sub _build_xml_document { parse(file_content) }
sub _build_data         { find(xml_document) }
```

Moose — миксины

```
with 'Role::HasPassword';
```

```
package Role::HasPassword;
use Moose::Role;
use Some::Digest;

has password => (
  is => 'ro',
  isa => 'Str',
);

sub password_digest {
  my ($self) = @_;

  return Some::Digest->new($self->password);
}
```

Moose — делегирование

```
has doc => (
  is      => 'ro',
  isa     => 'Item',
  handles => [qw(read write size)],
);
```

```
has last_login => (
  is      => 'rw',
  isa     => 'DateTime',
  handles => { 'date_of_last_login' => 'date' },
);
```

```
{
  handles => qr/^get_(a|b|c)|set_(a|d|e)$/,
  handles => 'Role::Name',
}
```

Moose — и т. д.

```
before 'is_adult' => sub { shift->recalculate_age }
```

```
subtype 'ModernDateTime'
  => as 'DateTime'
  => where { $_->year() >= 1980 }
  => message { 'The date is not modern enough' };

has 'valid_dates' => (
  is  => 'ro',
  isa => 'ArrayRef[DateTime]',
);
```

```
package Config;
use MooseX::Singleton; # instead of Moose
has 'cache_dir' => ( ... );
```

Moose — аналоги

- Moose
- Mouse
- Moo
- Mo
- M

Содержание

ДЗ 6

https://github.com/Nikolo/Technosfera-perl/

`/homeworks/iter`

```perl
my $iterator = Local::Iterator::Array->new(
  array => [1, 2, 3]
);
$iterator->next(); # (1, 0);
$iterator->next(); # (2, 0);
$iterator->next(); # (3, 0);
$iterator->next(); # (undef, 1);
```

```perl
my $iterator = Local::Iterator::File->new(
    file => '/tmp/file.txt'
);
$iterator->next(); # ('A', 0);
$iterator->next(); # ('B', 0);
$iterator->all(); # [qw(C D E)]
$iterator->next(); # (undef, 1);
```