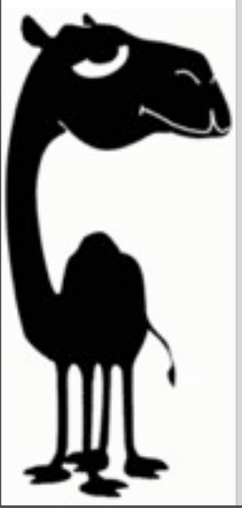
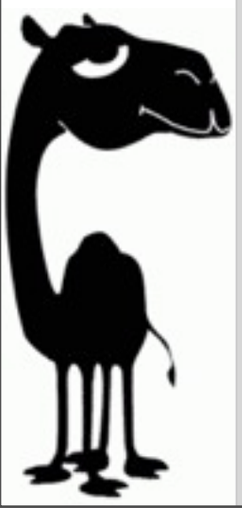
Amsterdam.pm

JoesMoes



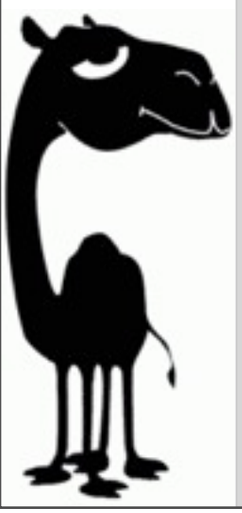
YousMousse



Amsterdam.pm

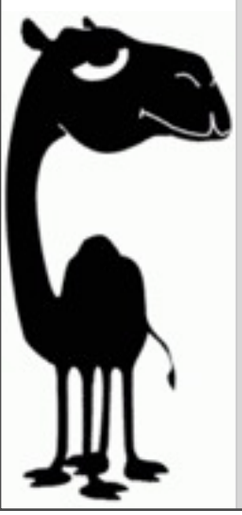
use Moose;

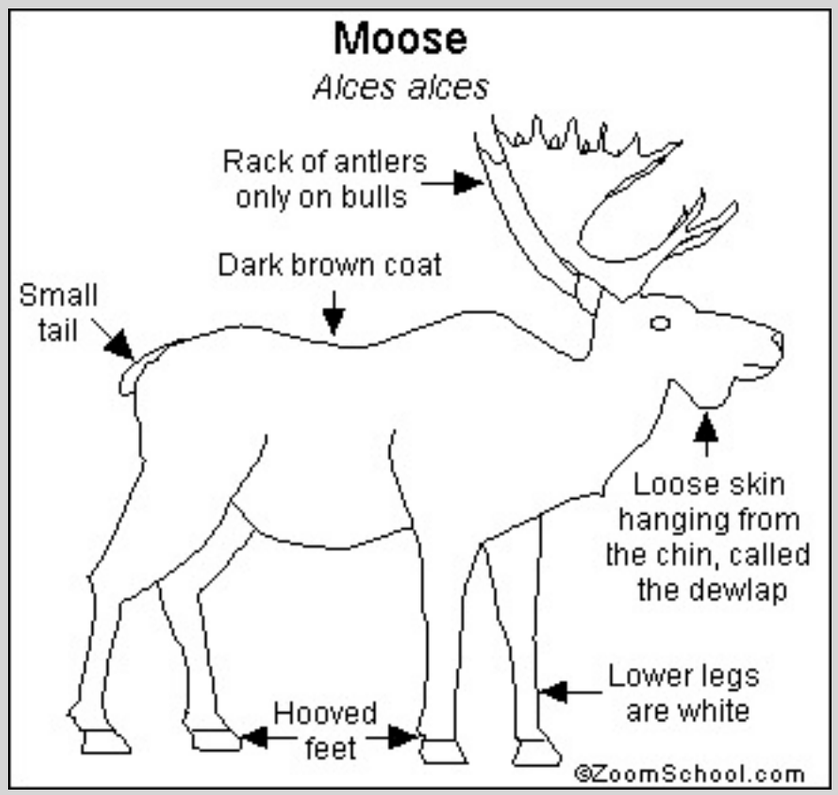
Abe Timmerman



What is Moose?

- Object system for perl 5
  - object construction/destruction
  - attribute declaration/accessors
  - method modifiers
  - specialize/inheritance
  - introspection
  - roles

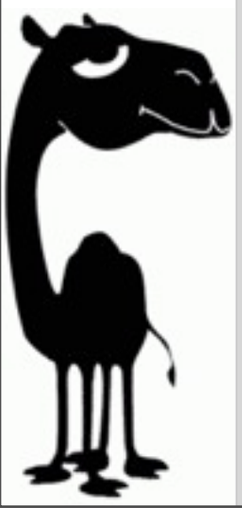




Amsterdam.pm

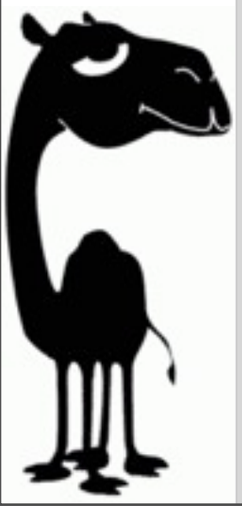
use Moose;

- => use warnings;
- => use strict;
- the package becomes a class



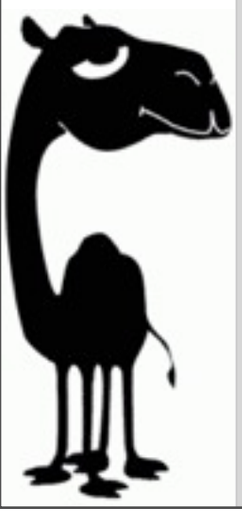
Attribute declaration

- 'has' keyword
- name
- properties
  - is => readonly/readwrite
  - isa => datatype
  - default => defaultvalue
  - (reader, writer) accessornames
  - ...



Attributes (code1)

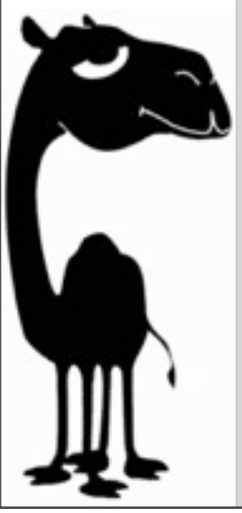
```
package Point::2D;
use Moose;

has x => (is => 'rw', isa => 'Int');
has y => (is => 'rw', isa => 'Int');

sub move {
    my $self = shift;
    my %args = @_;

    $self->x($self->x + $args{'delta_x'});
    $self->y($self->y + $args{'delta_y'});

    return $self;
}

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```



Attributes (code1)

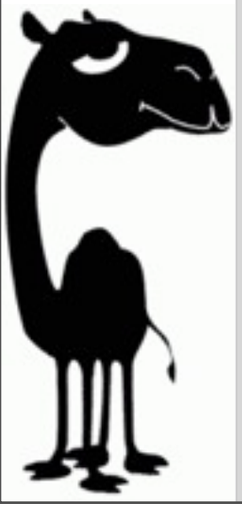
```
package Point::2D;
use Moose;

has x => (is => 'rw', isa => 'Int');
has y => (is => 'rw', isa => 'Int');

sub move {
    my $self = shift;
    my %args = @_;

    $self->x($self->x + $args{'delta_x'});
    $self->y($self->y + $args{'delta_y'});

    return $self;
}

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

attribute declaration



Attributes (code2)

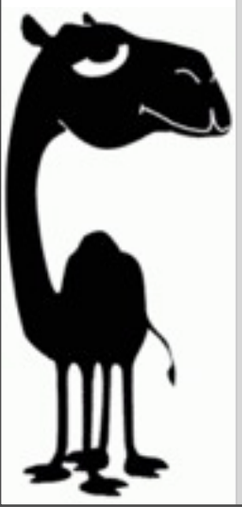
```
#! /usr/bin/perl
use warnings;
use strict;

use Test::More 'no_plan';

use Point::2D;

{
    my $o = Point::2D->new(x => 0, y => 0);
    isa_ok $o, 'Point::2D', "Oorsprong";
    is $o->x, 0, "  X-Coord " . $o->x;
    is $o->y, 0, "  Y-Coord " . $o->y;

    $o->move(delta_x => 42, delta_y => 42);
    is $o->x, 42, "  X-Coord " . $o->x;
    is $o->y, 42, "  Y-Coord " . $o->y
}
```



Attributes (code2)

```
#! /usr/bin/perl
use warnings;
use strict;

use Test::More 'no_plan';

use Point::2D;

{
    my $o = Point::2D->new(x => 0, y => 0);
    isa_ok $o, 'Point::2D', "Oorsprong";
    is $o->x, 0, "  X-Coord " . $o->x;
    is $o->y, 0, "  Y-Coord " . $o->y;

    $o->move(delta_x => 42, delta_y => 42);
    is $o->x, 42, "  X-Coord " . $o->x;
    is $o->y, 42, "  Y-Coord " . $o->y
}
```

automatic constructor

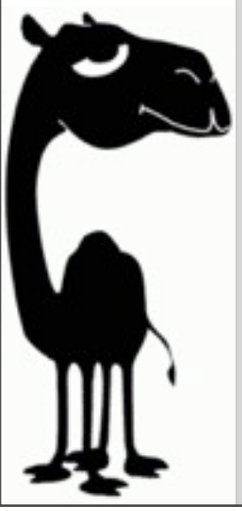


Attributes (code2)

```
#! /usr/bin/perl
use warnings;
use strict;

use Test::More 'no_plan';

use Point::2D;

{
    my $o = Point::2D->new(x => 0, y => 0);
    isa_ok $o, 'Point::2D', "Oorsprong";
    is $o->x, 0, "  X-Coord " . $o->x;
    is $o->y, 0, "  Y-Coord " . $o->y;

    $o->move(delta_x => 42, delta_y => 42);
    is $o->x, 42, "  X-Coord " . $o->x;
    is $o->y, 42, "  Y-Coord " . $o->y
}
```

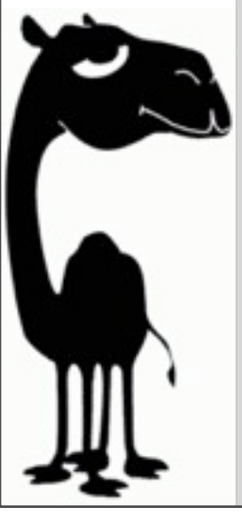
automatic constructor

automatic accessors



Specialize/inheritance

- 'extends' keyword
- also works for non Moose objects



Amsterdam.pm

Method modifiers

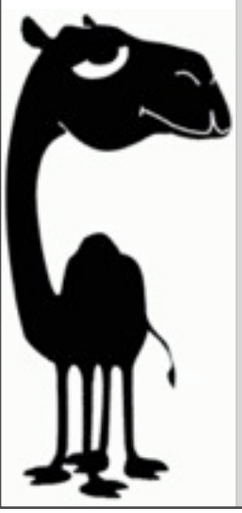
- before
- after
- around
- override / super
- augment / inner

Specialize (override)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

override move => sub {
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return super();
};

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

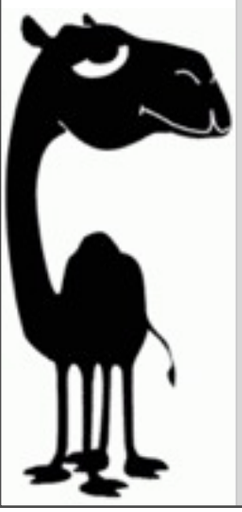
Specialize (override)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

override move => sub {
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return super();
};

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```



Specialize (override)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

override move => sub {
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return super();
};

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

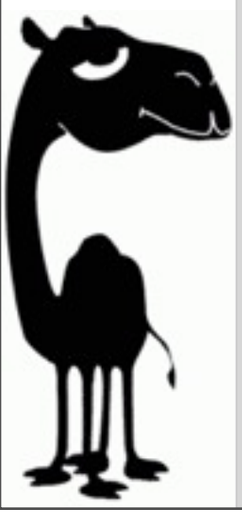


Specialize (override)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

override move => sub {
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return super();
};

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

specialize

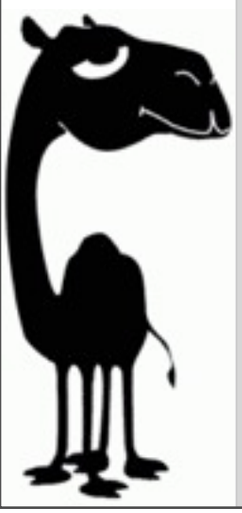
method modifier



Specialize (around)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

around move => sub {
    my $orig = shift;
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return $self->$orig(%args);
};


1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

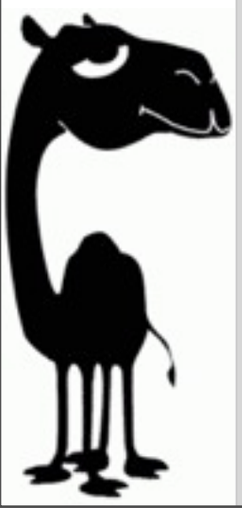
specialize



Specialize (around)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

around move => sub {
    my $orig = shift;
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return $self->$orig(%args);
};


1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

specialize

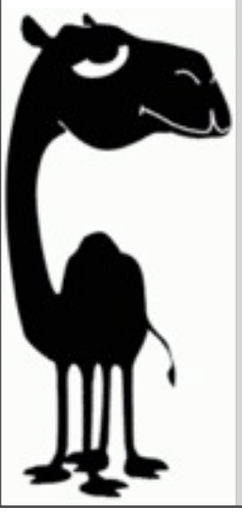


Specialize (around)

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

around move => sub {
    my $orig = shift;
    my $self = shift;
    my %args = @_;

    $self->z($self->z + $args{'delta_z'});
    return $self->$orig(%args);
};


1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

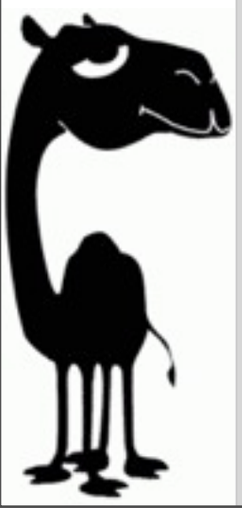
specialize

method modifier



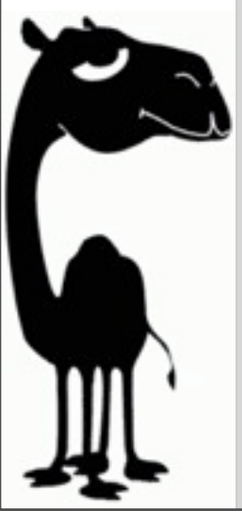
Introspection

- Via meta()
  - Relations
    - superclasses()
    - subclasses()
  - Methods
    - get_all_methods()
    - get_all_method_names()
  - Attributes
    - get_attribute()
    - get_attribute_list()
    - get_all_attributes()
  - "Fixate" class definition
    - meta->make_immutable()



Roles

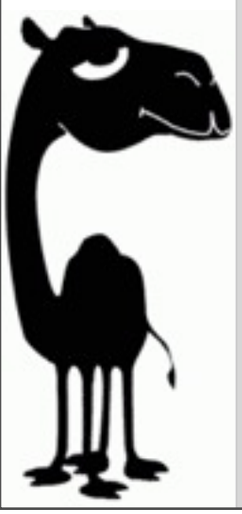
- aka "mixin"
- aka "interface"
- use Moose::Role;
  - requires ...
- with 'Role::Package';



Roles in action (1)

```
package Point::2D;
use Moose;

has x => (is => 'rw', isa => 'Int');
has y => (is => 'rw', isa => 'Int');

with 'Point::Movable';

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

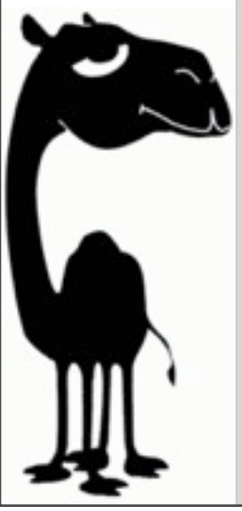


Roles in action (1)

```
package Point::2D;
use Moose;

has x => (is => 'rw', isa => 'Int');
has y => (is => 'rw', isa => 'Int');

with 'Point::Movable';

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```

add the role

```
package Point::3D;
use Moose;

extends 'Point::2D';

has z => (is => 'rw', isa => 'Int', required => 1);

1;
no Moose;
__PACKAGE__->meta->make_immutable;
```



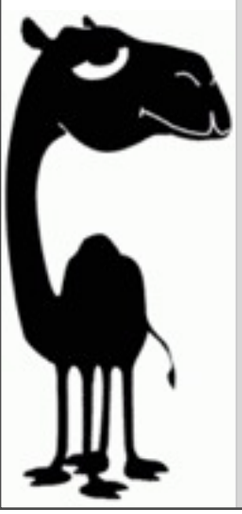
Roles in action (2)

```
package Point::Movable;
use Moose::Role;

sub move {
    my $self = shift;
    my %args = @_;

    my @attributes = grep
        /^(?:x|y|z)$/
      , map
        $_->name
      , $self->meta->get_all_attributes;

    for my $attribute (@attributes) {
        my $new_val = $self->$attribute + $args{'delta_'.$attribute};
        $self->$attribute($new_val);
    }
    return $self;
}

1;
```

Roles in action (2)

```
package Point::Movable;
use Moose::Role;

sub move {
    my $self = shift;
    my %args = @_;

    my @attributes = grep
        /^(?:x|y|z)$/
      , map
        $_->name
      , $self->meta->get_all_attributes;

    for my $attribute (@attributes) {
        my $new_val = $self->$attribute + $args{'delta_'.$attribute};
        $self->$attribute($new_val);
    }
    return $self;
}

1;
```

definine role

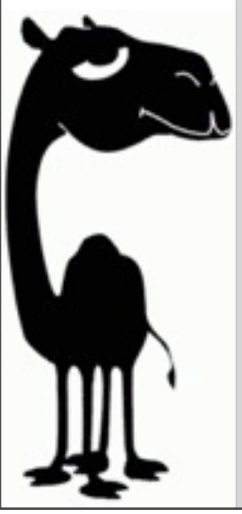


Roles in action (2)

```
package Point::Movable;
use Moose::Role;

sub move {
    my $self = shift;
    my %args = @_;

    my @attributes = grep
        /^(?:x|y|z)$/
      , map
        $_->name
      , $self->meta->get_all_attributes;

    for my $attribute (@attributes) {
        my $new_val = $self->$attribute + $args{'delta_'.$attribute};
        $self->$attribute($new_val);
    }
    return $self;
}

1;
```

definine role

definine behaviour



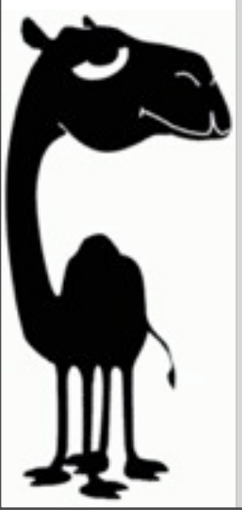
Roles in action (2)

```
package Point::Movable;
use Moose::Role;

sub move {
    my $self = shift;
    my %args = @_;

    my @attributes = grep
        /^(?:x|y|z)$/
      , map
        $_->name
      , $self->meta->get_all_attributes;

    for my $attribute (@attributes) {
        my $new_val = $self->$attribute + $args{'delta_'.$attribute};
        $self->$attribute($new_val);
    }
    return $self;
}

1;
```

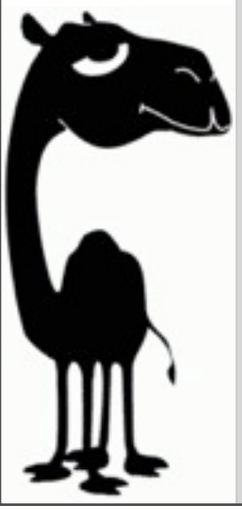
definine role

definine behaviour

introspection



Questions



Thank you!