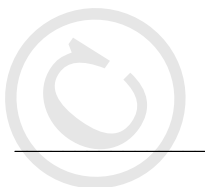


**Student-notities**  
**Object-georiënteerd Perl**  
**Voorbeeld-hoofdstuk**

06. Inheritance

 **at computing**  
**The Linux/UNIXperts**

Nijmegen

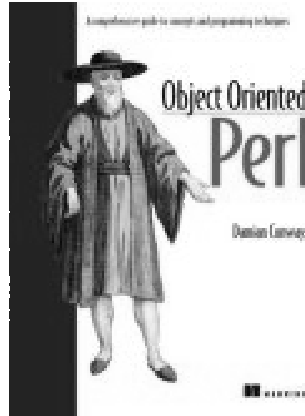


---

Copyright © AT Computing 2002  
Versie: 1b

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.

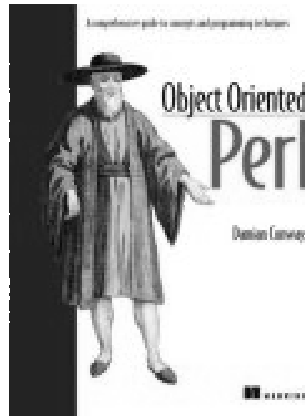




Figuur 1.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## @ISA

### ❖ "oud"

```
package main;
use Pers;
my $minou = Pers->new;

package Pers;
use Kat;
@Pers::ISA = 'Kat';

package Kat;
use Zoogdier;
@Kat::ISA = 'Zoogdier';

package Zoogdier;
use Dier;
@Zoogdier::ISA = 'Dier';

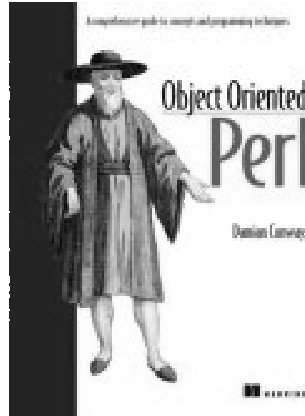
package Dier;
```

- ❖ Elk package is een losse file.
- ❖ De packages moeten expliciet om de code van de base-class vragen.
- ❖ @ISA moet een globale variabele zijn, vandaar de expliciete packagenaam.

**Figuur 2.**

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## use base

### ❖ vanaf Perl 5.004\_04

```
package Kat;  
use base 'Zoogdier';
```

staat voor

```
package Kat;  
BEGIN {  
    @Kat::ISA = 'Zoogdier';  
    require Zoogdier;  
}
```

### ❖ Dus

```
package main;  
use Pers;  
my $minou = Pers->new;
```

```
package Pers;  
use base 'Kat';
```

```
package Kat;  
use base 'Zoogdier';
```

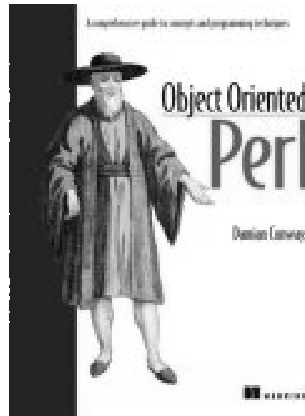
```
package Zoogdier;  
use base 'Dier';
```

```
package Dier;
```

Figuur 3.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Test is-a relatie

❖ Elk package inherits van **UNIVERSAL**

❖ Test instantie

```
my $minou = Kat->new;
$minou->isa('Kat')           ➔ true
$minou->isa('Zoogdier')     ➔ true
$minou->isa('Hond')         ➔ false

$minou->isa('HASH')         ➔ true
                           (yuk?)
```

❖ Test class

```
Kat->isa('Zoogdier')       ➔ true
Kat->isa('Dier')           ➔ true
Kat->isa('Hond')           ➔ false
Dier->isa('Kat')           ➔ false
```

❖ Ook voor andere referenties

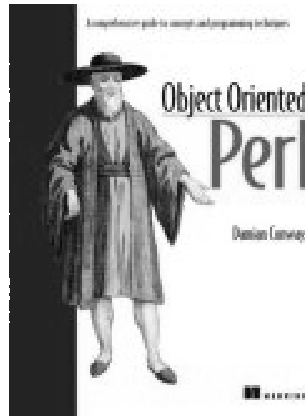
```
my $h = {};
print ref $h;              ➔ HASH
UNIVERSAL::isa($h, 'HASH') ➔ true
```

❖ Zie `perldoc UNIVERSAL`

Figuur 4.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Inherit functionaliteit -I

- ❖ Methods worden automatisch opgezocht in base-class:

```

package main;
use Kat;
my $minou = Kat->new;
$minou->eet('brokjes');

package Kat;
use base 'Zoogdier';
sub new() {bless {}, shift}

package Zoogdier;
sub eet($) {print "jummie $_[1]\n"}

```

- ❖ Dynamisch zoeken

```
$minou->eet('brokjes');
```

wordt *niet*

```
Kat::eet($minou, 'brokjes');
```

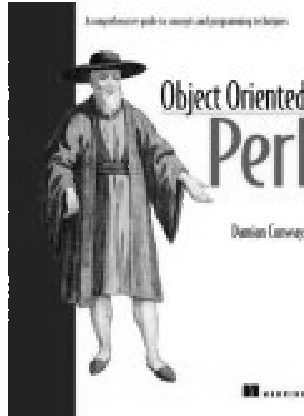
maar

```
Zoogdier::eet($minou, 'brokjes');
```

Figuur 5.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Inherit functionaliteit -II

- ❖ Nu heeft de kat een (voor zoogdieren) speciaal levenspatroon

```
package main;
use Kat;
my $minou = Kat->new;
$minou->eet('brokjes');

package Kat;
use base 'Zoogdier';
sub new() {...};

sub eet($) {
    my ($self, $wat) = @_;

    $self->SUPER::eet($wat);

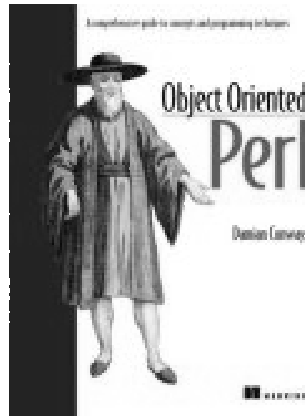
    $self->gebruikKattenbak;
    $self->slaap('half uur');
}

package Zoogdier;
sub eet($) {print "Lekker $_[1]!\n"}
```

Figuur 6.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Test can

### ❖ UNIVERSAL::can

#### ❖ Test instance

```
my $minou = Kat->new;  
$minou->can('new')           ➔ true  
$minou->can('eet')           ➔ true  
$minou->can('vliegen')       ➔ false
```

#### ❖ Test class

```
Kat->can('new')               ➔ true  
Kat->can('eet')               ➔ true  
Kat->can('vliegen')          ➔ false
```

#### ❖ can retourneert code referentie

```
$minou->eet('brokjes');
```

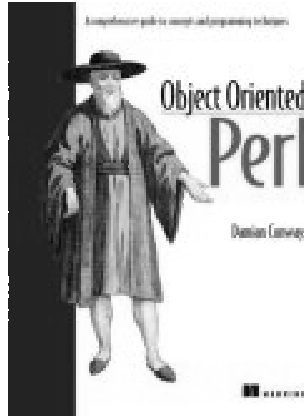
is gelijk aan (N.B.: `ref $minou eq 'Kat'`)

```
Kat->can('eet')->($minou, 'brokjes');
```

Figuur 7.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Inherit data

- ❖ Elk object is één (referentie naar een) hash. Bij `$minou` zit `Dier`, `Zoogdier` en `Kat` kennis in dezelfde hash.
- ❖ Gevaar: *name-clashes*, daarom wordt meestal de afkorting van de packagenaam aan de keys toegevoegd.

```
use Data::Dumper;
my $minou = Kat->new;
print Dumper($minou);
```

➔

```
$VAR1 = bless( {
    'K_naam' => 'Minou',
    'K_vlooiën' => 1,
    'Z_tepels' => 8,
    'D_lengte' => 45,
    'D_leeftijd' => 13
  }, 'Kat' );
```

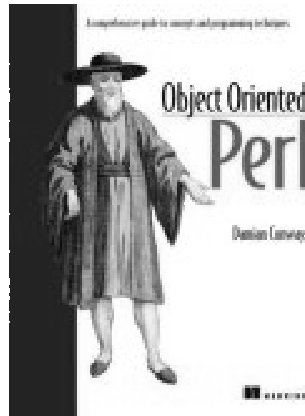
- ❖ In de praktijk
 

<code>Dier</code>	➔ <code>D_lengte</code>
<code>Dier::Zoogdier</code>	➔ <code>DZ_tepels</code>
<code>Dier::Zoogdier::Kat</code>	➔ <code>DZK_vlooiën</code>
- ❖ Zie `Class::Delegation`

Figuur 8.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Instantiatie -I

❖ Init op elke laag waar iets moet gebeuren.

```

package Dier::Zoogdier::Kat;
use base 'Dier::Zoogdier';
sub init($)
  my ($self, $args) = @_;
  $args->{tepels} || = 8;
  $self->SUPER::init($args);
  $self->{DZK_vlooiën}
    = defined $args->{vlooiën}
      ? $args->{vlooiën} : 3;
  $self;
}

```

nul is geldige waarde

```

package Dier::Zoogdier;
use base 'Dier';
sub init($) {
  my ($self, $args) = @_;
  $self->SUPER::init($args);
  $self->{DZ_tepels} = $args->{tepels} || 2;
  $self;
}

```

nul is niet mogelijk

```

package Dier;
sub new() {
  my ($class, %args) = @_;
  (bless {}, $class)->init(\%args);
}
sub init($) {
  my ($self, $args) = @_;
  $self->{D_leeftijd} = $args->{leeftijd} || 0;
  $self->{D_lengte} = $args->{lengte};
  $self;
}

```

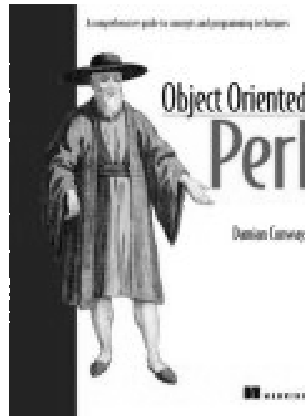
nul is ook de default

undef is de default

Figuur 9.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Instantiatie -II

❖ We produceren onze nieuwe kat

```
package main;
use Dier::Zoogdier::Kat;
use Data::Dumper;

my $minou = Dier::Zoogdier::Kat->new
( vlooiën => 0
  , leeftijd => 13
  );

print Dumper($minou);
```

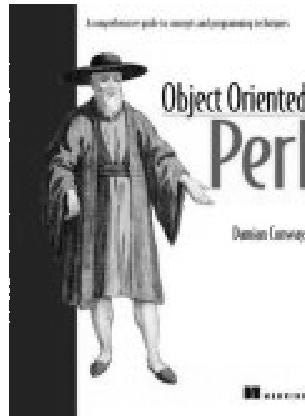
➔

```
...
'D_lengte'      => undef,
'D_leeftijd'    => 13,
'DZ_tepels'     => 8,
'DZK_vlooiën'  => 0
...
```

Figuur 10.

### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Multiple Inheritance -I

- ❖ Opgelet? @ISA is een array!

```
package Kat;  
use Zoogdier;  
use Kostenpost;  
@Kat::ISA = qw(Zoogdier Kostenpost);
```

of

```
package Kat;  
use base qw(Zoogdier Kostenpost);
```

- ❖ Volgorde inheritance is soms belangrijk:

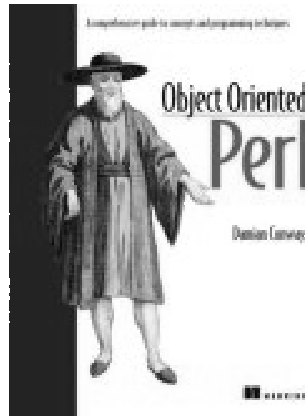
```
package Zoogdier;  
sub prijs() { 42 } # aanschafprijs  
  
package Kostenpost;  
sub prijs() { 112 } # voedingskosten  
  
package main;  
use Kat;  
my $minou = Kat->new;  
print $minou->prijs; ➔ 42
```

Figuur 11.



### Student-notities

Bij deze cursus leveren wij — naast dit werkboek — het boek "*Object Oriented Perl*" (van de auteur D. Conway) waarin deze materie verder is uitgewerkt.



## Multiple Inheritance -II

❖ Prioriteit belangrijk bij **SUPER** en **can**.

❖ Zelf vorken bouwen:

```
package Kat;
sub init($) {
    my ($self, $args) = @_;
    $self->Zoogdier::init($args);
    $self->Kostenpost::init($args);
    $self->{K_naam} = 'Minou';
    $self;
}

sub DESTROY() {
    my $self = shift;

    $self->Zoogdier::DESTROY
        if Zoogdier->can('DESTROY');

    $self->Kostenpost::DESTROY
        if Kostenpost->can('DESTROY');

    print "R.I.P.\n";
}
```

❖ Zie module **NEXT**

☞ multiple inheritance support dus (kunst)matig.

Figuur 12.

