Einschub: Perl-Module

0.1 Klassen, Instanzen, Vererbung

- Klassenkonzept: Eine Klasse beschreibt einen Objekttyp mit Namen, Attribute und Methoden (in diesen kann unterschieden werden in virtuelle, abstrakte, private, geschützte (protected) oder öffentliche (public) Methoden)

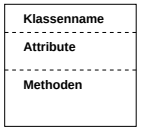

Abbildung 1: allgemeine Klasse

- Klassen können instantiiert werden, d.h. ein konkretes Objekt kann angelegt werden; dazu wird von der Klasse in Konstruktor als Methode zur Verfügung gestellt (meist: **new**, in Win32 auch **Create**())

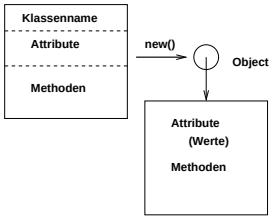

Abbildung 2: Objekt einer Klasse

- Klassenhierarchien durch Spezialisierung (Typerweiterung): neue Attribute und Methoden können hinzukommen, bestehende Methoden werden durch neue Methoden überlagert

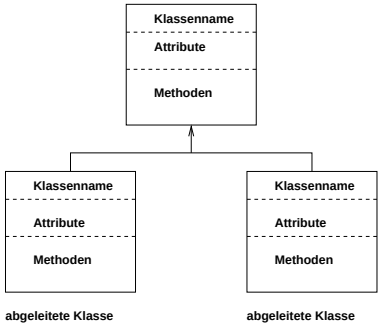

Abbildung 3: Klassenhierarchie

0.2 Modularisierung in Perl

- Namespace (Namensraum, Bereich mit eigener Symboltabelle)

  Default-Namespace: **main**

- Package: Ein Namensraum wird in einem Package gehalten, beginnend mit einer *Package* Deklaration (Parameter Package-Name)
  - Objekte in einem Package sind lokal (ausgenommen die mit **my** deklarierten: deren Gültigkeit ist auf den umgebenden Block / Subroutine beschränkt)
  - Bezug auf Objekte in einem anderen Package: `$OtherPackage::Object`
  - fehlt `OtherPackage`, so wird `main` angenommen:
    `$var` ist identisch mit `$main::var` und mit `$::var`
  - Packages können geschachtelt werden:
    Package `Land` in Package `Staat`: eine Variable `$var` in `Land` wird bezeichnet als `$Staat::Land::var` (relative Benennung ist **nicht** zulässig!)
- Module: Ein Modul ist ein Package, das in einer Datei gehalten wird, deren Name der Name des Package ist (Endung: **.pm**)
  - Perl sucht Module in der Liste `@INC` mit Bibliotheks-Directories (entspricht in etwa der PATH-Variable)

```
#!/usr/local/bin/perl -w

foreach $i (@INC) { print "$i\n" }
```

liefert auf thales

```
/usr/local/lib/perl5/5.00503/sun4-solaris
/usr/local/lib/perl5/5.00503
/usr/local/lib/perl5/site_perl/5.005/sun4-solaris
/usr/local/lib/perl5/site_perl/5.005
.
```

auf Linux

```
/usr/lib/perl5/5.8.0/i586-linux-thread-multi
/usr/lib/perl5/5.8.0
/usr/lib/perl5/site_perl/5.8.0/i586-linux-thread-multi
/usr/lib/perl5/site_perl/5.8.0
/usr/lib/perl5/site_perl
.
```

Welche Dateien sind enthalten? Unter anderem: `IO.pm` und Directory `IO`

- Module (Package), enthalten in einem Package, (z.B. `ParentPackage::ModulName`) sucht Perl in einem Unterkatalog von `@INC` (also dort nach `ModulName.pm`.
- Beispiel:

```
swg@spatz:> cat /usr/lib/perl5/5.8.0/i586-linux-thread-multi/IO.pm | \
            head -3
#
package IO;

swg@spatz:> ls /usr/lib/perl5/5.8.0/i586-linux-thread-multi/IO
Dir.pm     File.pm    Handle.pm    Pipe.pm    Poll.pm    Seekable.pm
Select.pm  Socket.pm
swg@spatz:> cat /usr/lib/perl5/5.8.0/i586-linux-thread-multi/IO/Handle.pm
            | head -3

package IO::Handle;
swg@spatz:>
```

- Benötigt man (z.B. via `use` oder `require`) das Modul `IO::File`, so wird dieses als `IO/File.pm` gesucht!
- `use` vs. `require`
  - `use` zieht das Modul zur Übersetzungszeit herein (flutet den Namensraum)
  - `require`: alle verwendeten Objekte müssen qualifiziert werden!

0.3 OO in Perl

- Ein Modul, das eine Klasse definiert muss einen Konstruktor für die Erzeugung einer Instanz (Objekt) enthalten: **new()**
- *new()* liefert eine Referenz auf das erzeugte Objekt!
- Methoden sind Funktionen (Subroutinen), die als erstes Argument eine solche Objektreferenz erwarten
- Aufruf von Methoden:

  ```
  PackageName->constructor(args)->method_name(args);
  ```

  oder

  ```
  $object = PackageName->constructor(args);
  $object->method_name(args);
  ```

- Objekte werden gelöscht (*garbage collector*), wenn die letzte Referenz darauf verschwunden ist

Objekt-Syntax am Beispiel IO

Programm 0.1 Instanz erzeugen und bearbeiten

```
#!/usr/local/bin/perl -w
use strict;
use IO::File;

my $addressbook = "addressbook"; # Dateiname zuweisen

################ one way ############################
my $book = new IO::File ; # Instanz erzeugen

$book->open("< $addressbook") or
   die "Cannot open $addressbook\n";

############ another way ############################
my $outbook = new IO::File ">XXX" or
   die "Cannot open XXX\n";

#####################################################

while (<$book>) {
   chomp;
   my ($name, $address) = split /:/;
   $outbook->write("$name:$address\n");
}
$book->close;
```

Programm 0.2 Instanz erzeugen und bearbeiten

```
#!/usr/local/bin/perl -w
use strict;
use IO::File;

my $addressbook = "addressbook"; # Dateiname zuweisen

################ one way ############################
my $book = new IO::File "$addressbook", "r" or
   die "Cannot open $addressbook\n";

############ another way ############################
my $outbook = new IO::File "XXX", O_WRONLY|O_APPEND or
   die "Cannot open XXX\n";

#####################################################

while (<$book>) {
   chomp;
   my ($name, $address) = split /:/;
   $outbook->write("$name:$address\n");
}
$book->close;
```