

To be covered today

- Packages and modules
- ■Object-oriented Perl programming
 - ▶ making objects
- using objects

CSE3395

CSE3395

Further reading on object-oriented Perl

Programming Perl (camel book)

references given throughout lecture notes

Advanced Perl Programming

chapters 6-8

manpages

perl mod
about packages and modules

perl toot
Tom's object-oriented tutorial; simpler introduction to OO

perl bo t

complete but terse explanation of OO

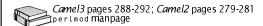
perl bot
Bag of object tricks; some more complex OO stuff

Packages

3

Keeping namespace unpolluted

- A package is a default namespace for global variables
- ► similar to namespace keyword in ANSI/ISO C++
- ▶ package parcel;
- all variables are now kept in the package called parcet
- ■Initial package is main
- Can refer to variables in other packages
 - \$parcel::scalar, @parcel::array, parcel::func()



5

Modules

4

Using packages for code re-use

- A module is a package that contains reusable code
 - ► If module's package is Weekday, module is stored in file Weekday.pm
- ■Access a module with the use keyword
- ▶ use Weekday ("weekday");
- ▶ includes all code of Weekday.pm
- ▶ implicitly calls Weekday::import("weekday")
 - usually makes main::weekday() equal to Weekday::weekday()



Object-orientation

- A class is a description of a type of thing
 - ▶ in Perl, implemented as a module
- An object is an instance of a class
- in Perl, implemented as a referenced value
- A method is a function that operates on a class or object
 - class methods (static methods) apply to the class as
- object methods apply to one object

CSE3395

CSE3395

CSE3395

▶ in Perl, both implemented as a subroutine in module



Making an object

- A constructor is a class method that returns a reference to a new object
- often called new, but not necessarily
- Typically, new object is an anonymous hash
- but can be any type of value, e.g. array
 object's attributes stored in hash like a struct
- Object must be blessed into class
- ▶ bless keyword associates object's value with class
- can check what class an object is in with ref keyword



Invoking methods

Calling a function on a class or object

- ■Perl translates method calls into calls to functions in the class' module
 - sometimes calls different class if using inheritance
- Class methods: class=>cmethod(params)
- class name is added as implicit first parameter
- translated to Class::cmethod("Class", params)
- ▶ \$person = Student->new(\$name);
- ■Object methods: \$oref->omethod(params)
- object reference is added as implicit first parameter
- translated to Class::omethod(\$oref, params) ▶ \$person->enrol("CSE3395");
- Camel3 pages 311-313; Camel2 pages 291, 295-297

perlobj, perltoot manpages

9

Example

A constructor (class method) for a Student class

```
package Student:
# This is called "new" only by convention.
   # Find out class name (useful if inherited).
   my ($class) = shift;
   # Make a new object, and bless it into the class.
   my ($self) = {};
  # Josetty - {y, bless $self, $class; # Initialize the object's attributes 
$self->{"ID"} = make_new_ID_number(); 
$self->{"NAME"} = $_[0]; 
$self->{"SUBJECTS"} = [];
   # Return a reference to the new object.
return $self;
```

10

Example

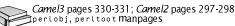
An object method for enrolling a student in a subject

```
package Student;
sub enrol
  # First (implicit) parameter is the object
  # reference.
my ($self) = shift;
  # Remaining parameters are the subjects to
  # add. Add them to the array reference in
# $self->{"SUBJECTS"}.
  push a{$self->{"SUBJECTS"}}, a_;
```

Destroying an object

Destructors

- ■Special method DESTROY() is called when an object is no longer referenced
- rarely needed because of Perl's reference-count memory management
- useful to keep track of class meta-data or to dump persistent objects to disk



perlobj, perltoot manpages

Example A random-number-generator class package Random; # The new constructor takes one parameter, # the range of numbers to produce. sub new # First (implicit) parameter is the class. my (\$class) = shift; # Get the range, and put it in an anon hash. my (\$range) = \$_[0]; my (\$self) = { range => \$range }; CSE3395 # Bless the reference into the Random package. bless \$self, \$class; return \$self: # This class is continued on the next slide.

Example A random-number-generator class, continued # Continuing in package Random from last slide. # The roll object method returns a random number # from 1 to the object's range, inclusive. # First (implicit) parameter is the object. my (\$self) = shift; # Fetch the range from the object. my (\$range) = \$self->{range}; # Return the random number. return int (rand \$range) + 1;

14

To indicate successful inclusion by the # use keyword, must end this file with truth.

sub roll

CSE3395

1;

Example

13

Using the random-number-generator class

```
package main;
# Dedicated to all the role players I know. :)
use Random:
# Want to generate numbers from 1 to 6 for one
# die and from 1 to 10 for the other.
$die1 = Random->new(6);
$die2 = Random->new(10);
# Roll both dice 100 times.
for ($i = 0; $i < 100; $i++)
{
  # Print out the sum of the two dice.
print $die1->roll() + $die2->roll(), "\n";
```

▶ package Student; @ISA = ("Person");

Any method not found in student will now fall back to the same-named method in Person

Inheritance Making a derived class

A class can be derived from one or more parent classes by setting the alsa array

Camel3 page 321-324; Camel2 page 292 perlobj, perltoot manpages

15

16

OO Perl versus C++

A language feature comparison

- Underlying representation
- ▶ Perl: anonymous hash/array/scalar, looked up by reference
- ► C++: opaque structure similar to a C struct
- Object methods and class methods
- Perl: equivalent syntax, distinguished by use
- ► C++: class methods denoted by static keyword
- ■Class data

CSE3395

- ► Perl: package-level global variables
- ► C++: variables defined with static keyword

OO Perl versus C++

A language feature comparison, continued

- ■Instance data
- Perl: stored as key/value in object hash
- C++: stored in opaque structure
- Perl: defined and invoked like other class methods
- ► C++: defined as class function, called using new keyword

Destructors

- ▶ Perl: defined as DESTROY function in module; invoked
- C++: defined as ~class function; invoked implicitly

OO Perl versus C++

A language feature comparison, continued

- Inheritance
- ▶ Perl: multiple, using alsa array
- ► C++: multiple, using :public Base in class declaration
- ■Polymorphism
- Perl: applies to all methods, no need to distinguish
 C++: off unless activated with virtual keyword

- ► Perl: none, relies on programmer's manners
- ► C++: rigorously enforced with public, private, protected keywords

Covered today

- Packages and modules
- ▶ package keyword
- Object-oriented Perl programming
- making objects
- bless function
- using objects
 - \$obj->method() notation

19 20

Going further

More things related to today's topic

- ■Symbol tables
- ▶ how Perl stores its data internally
- Camel3 pages 293-296; Camel2 pages 281-283
- Autoloading
 - automatically generating functions on the fly
 - ► Camel3 pages 296-298; Camel2 pages 284-285
- writing reusable code
- Camel3 pages 301-307

Advanced topics

More of what Perl can do

- Networking
- client/server programming, sockets, DNS, etc.
- Tying variables
- invisibly attaching a class/file/database to a variable
- Graphical user interfaces
- ▶ using the Tk/Perl windowing interface
- ■Code generation
- generating and evaluating Perl code on the fly
- ■Interfacing Perl and C
- taking advantage of both languages' strengths

21 22

Advanced topics

Finding out more about Perl

- Programming Perl
 - covers most aspects of Perl, including standard library, tying objects to variables, etc.
- Advanced Perl Programming
- Sriram Srinivasan, O'Reilly 1997
- covers advanced topics, including details on references, object orientation, networking, graphical user interfaces, writing libraries, embedding Perl in C and vice versa, etc.
- ► Tom Christiansen & Nathan Torkington, O'Reilly 1998
- ▶ lots of neat solutions to lots of common Perl problems
- manpages

CSE2395/CSE3395

CSE2395/CSE3395 lecture notes copyright © 2000-2001 Deborah Pickett. Reproduction of this presentation for nonprofit study use is permitted. All other reproduction, including for other educational courses, must be authorized in writing by the author.