

# Exercise

Exercise  
Comments

Object In Perl

Databases

```
enewetak.priv  
    state = free
```

```
eolie.priv  
    state = offline  
    jobs = 0/9889.omega.biotech.kth.se, 1/10835.omega.biotech.kth.se
```

```
elba.priv  
    state = offline
```

```
etorofu.priv  
    state = job-exclusive  
    jobs = 0/15420.omega.biotech.kth.se, 1/8963.omega.biotech.kth.se
```

Which are free?

# Exercise, Solution # 1

“Redefine the Line” solution.

```
sub printFree {
    open STATES, "ex9.txt"
        or die "Can't open ex9.txt: $!\n";
    local $\ = "\n\n";
    my $count = 0;
    while ($line = <STATES>) {
        my ($cname) = /(\b.+?).priv .*?\bstate = f/s;
        if (defined $cname) {
            print "$cname is free.\n";
            $count++;
        }
    }
    close STATES;
    print "Sorry, no machines are currently free\n"
        unless $count;
}
```

Exercise  
Comments

Object In Perl

Databases

## Exercise, Solution # 2

“Parsing” solution.

```
sub printFree2 {
    open STATES, "ex9.txt" or die;
    my ($info) = ('');
    while ($line = <STATES>) {
        if(length $line > 1) {
            $info .= $line;
            next;
        }
        my ($cname) =
            ($info =~/(\b.+?).priv .*?\bstate = f/s);
        print "$cname is free.\n" if defined $cname;
        $info = '';
    }
    close STATES;
}
```

Exercise  
Comments

Object In Perl

Databases

## Exercise, Solution # 3

“State-engine”-type solution.

```
sub printFree3 {
    open STATES, "ex9.txt"
        or die "Can't open ex9.txt: $!\n";
    my $count = 0;
    while (<STATES>) {
        if (/^\w+/) {
            $cname = $1;
        } elsif (/^state = f/) {
            print "$cname is free.\n";
            $count++;
        }
    }
    close STATES;
    print "Sorry, no machines are currently free\n"
        unless $count;
}
```

Exercise  
Comments

Object In Perl

Databases

## Exercise, Solution # 4

“Slurp'em all” Solution.

```
sub printFree4 {
    open STATES, "ex9.txt"
        or die "Can't open ex9.txt: $!\n";
    local $/ = undef;
    my $count = 0;
    my $content = <STATES>;
    close STATES;

    my @cname = ($content =~ /^(.+)\.priv.*?state = f/sg);
    if (defined $cname) {
        print "Free Hosts: ", join(' ', @cname), "\n";
    } else {
        print "Sorry, no machines are currently free\n";
    }
}
```

Exercise  
Comments

Object In Perl

Databases

# Exercise, Solution # 5

Exercise  
Comments

Object In Perl

Databases

“Single-line-hack” solution.

```
perl -e 'undef $/;
        print "Free Hosts:",
        join " ",
        ((<> =~ /^(.+)\.priv.*?state = f/sg)), "\n";' \
< ex9.txt
```

# Using printf

Exercise  
Comments

Object In Perl

Databases

Most common ways of formatting output:

Input	Format	Output
3	%3d	3
3	%03d	003
'aa'	%3s	aa
'aa'	%-3s	aa
3.333	%5.1f	3.3

# Objects in Perl

Exercise  
Comments

Object In Perl

Databases

Object-oriented programming implemented with help of already available tools (long live Perl flexibility!).

`object` references.

`class` package.

`method` subroutine.

# Object Usage

Exercise  
Comments

Object In Perl

Databases

Assume you have a package to run a series of Gaussian calculations...

```
my $calculation = new Gaussian(functional => 'B3LYP');  
$calculation->set_molecule('coffeine.xyz');  
$calculation->run();  
$calculation->save_last_geometry_to('coffeine1.xyz');  
$calculation->set_molecule('coffeine1.xyz');  
$calculation->run();
```

# Object Implementation

```
package Gaussian;
sub new {
    my @class = shift;
    return bless { @_ };
}
sub set_molecule { my $self = shift;
    $self->{molecule} = $_[0];
}
sub run { my $self = shift;
    # Generate input file, run gaussian and
    # store the output location.
    $self->{output} = $self->{molecule} . '.out';
}
sub save_last_geometry_to { my $self = shift;
    my $fname = $_[0];
    # Extract the data from $self->{output}
    # and save to $fname.
}
1; # initialization successful?
```

Exercise  
Comments

Object In Perl

Databases

# Simple Databases

Simple databases allow efficient access to persistent hashes, without loading entire content to the memory (important when script is short-lived or the hash is large). `perldoc GDBM_File`.

```
use GDBM_File; # or SDBM_File, or DB_File
tie %words, 'GDBM_File', $ARGV[0], &GDBM_WRCREAT, 0640;
if(exists $words{$ARGV[1]}) {
    print 'Word ' . $ARGV[1] . ' translates to '
        . $words{$ARGV[1]} . "\n";
} else {
    print 'Database ' . $ARGV[0] .
        ' does not contain word ' . $ARGV[1] . "\n";
}
untie %words; # close database file.
```

- Difficult to go beyond simple  $k \rightarrow v$  relationships.

# Relational Databases

Exercise  
Comments

Object In Perl

Databases

Modules for accessing full-blown relational databases: MySQL, PostgreSQL, etc.. Packages usually distributed separately.

- Database Interface: `perldoc DBI`
- PostgreSQL database driver: `perldoc DBD::Pg`

# Usage Patterns of Relational Databases

Exercise  
Comments  
Object In Perl  
Databases

```
use DBI;
$dbh = DBI->connect("dbi:Pg:dbname=molecules")
    or die 'I think the server does not like me.';
$query = $dbh->prepare('SELECT name, weight FROM '.
                        'molecules ORDER BY name')
    or die $dbh->errstr;
$query->execute;
while( ($name, $weight) = $query->fetchrow_array) {
    print "$name has molecular weight $weight\n";
}
$dbh->disconnect;
```

Separation of prepare and execute relevant from performance and security standpoint.

# Database Updates

- Query needs to be prepared only once.
- Quoting of values is not a problem any more.
- Use `$dbh->do('QUERY')` for one-time, non-parametrized queries.

```
$query = $dbh->prepare('INSERT INTO ' .  
                        'pets(owner,kind,name) VALUES (?, ?, ?)');  
  
while(<CSV>) {  
    chomp;  
    my ($owner,$kind,$name) = split /,/;  
    $query->execute( $owner, $kind, $name );  
}
```

# Transactions

Exercise  
Comments

Object In Perl  
Databases

- Each database update is immediately flushed to disk – massive updates can have low performance.
- Sometimes, we want to have an ability to roll back transactions (when something goes wrong).
- Transactions are a way to collect a set of changes into a single batch.

## Transactions, cont.

```
$qp = $dbh->prepare('INSERT INTO pets(owner,kind,name)'  
                    . ' VALUES (?,?,?)');  
$qa = $dbh->prepare('INSERT INTO addresses(owner,address)'  
                    . ' VALUES (?,?)');  
$dbh->begin_work or die;  
while(<CSV>) {  
    chomp; my ($owner,$kind,$name,$town) = split /,/;  
    if($town eq 'Stockholm') {  
        $dbh->rollback; print "No big cities allowed.\n";  
        print "Skipping all the data up to now.\n";  
        $dbh->begin_work or die;  
    }  
    unless($qp->execute( $owner, $kind, $name ) and  
           $qa->execute( $owner, $town )) {  
        die 'Query execution failed: ' . $dbh->errstr;  
    }  
}  
$dbh->commit; # real write happens now!
```

- Comprehensive Perl Archive Network
- Packages for about everything.
- Visit `search.cpan.org`.
- Linux users may find it more convenient to install appropriate packages.
  - 1 Fedora: `yum install perl-Math-Spline`
  - 2 Debian/Ubuntu: `apt-get install perl-Math-Spline`

# Further Reading

Exercise  
Comments  
Object In Perl  
Databases

- Objects: Chapter 12.
- Web: <http://perl.com/>