```
##################################################
#### example.pl
package example;

use limit;

@ISA = (limit);

$myjobset = example->new();

### ジョブ集合の指定
# 一般化jobsetを使う場合（直接ジョブ名のリストを代入）
@{$myjobset->{Jobset_names}} = (1..40);
# 従来のjobset相当（jobset_n を利用してジョブ数だけを指定）
$myjobset->set_jobset_n_card(40);

### 名前→コマンドライン引数ルールの指定
# 書き方(1)：format文字列
$myjobset->{Jobset_name2args} = "%d"; # デフォルトと同じ
# 書き方(2)：eval文字列
$myjobset->{Jobset_name2args} = '$name+1'; # (2..41) になる
$myjobset->{Jobset_name2args} = 'sprintf ("%d %d", $name / 10 , $name%10);';
    # (0 1) (0 2) ... (0 9) (1 0) (1 1) ... (1 9) ... になる.
    # ("(0..3)", "(0..9)")のような書き方も可能なようにも,
    # jobsetの拡張や, ("(0..3)", "(0..9)")→名前リスト の関数提供でできる
# 書き方(3)：関数
$myjobset->{Jobset_name2args} = sub { return $_[0]*$_[0]; };
$myjobset->{Jobset_name2args} = ¥&square;

# $myjobset->{Jobset_exe} = "taskset 0x3 ./a.out"; # 使用するコアを指定
$myjobset->{Jobset_exe} = "./fib";
# $myjobset->{Jobset_args} = "%d";

$myjobset->{Parallel_after_all} ='{print "All jobs finished in parallel.¥n"}';

$myjobset->{Limit_card} = 1;

### Go!
$myjobset->start();

=comment 旧バージョン
$myjobset->{Jobset_before} = '{print "Jobset started.¥n"}';
$myjobset->{Jobset_after} = '{print "Jobset finished.¥n"}';
$myjobset->{Parallel_before} = '{print "Parallel started.¥n"}';
$myjobset->{Parallel_after} = '{print "Parallel finished.¥n"}';
$myjobset->{Limit_before} = '{print "Limit started.¥n"}';
$myjobset->{Limit_after} = '{print "Limit finished.¥n"}';
$myjobset->{Example_before} = '{print "Example started.¥n"}';
$myjobset->{Example_after} = '{print "Example finished.¥n"}';
=cut

##################################################
#### limit.pm
package limit;

use parallel;
use Thread::Semaphore;

@ISA = (parallel);

sub new {
    my $class = shift;
    my $self = $class->SUPER::new();
    $self->{Limit_card}=1;
    return bless $self, $class;
}

sub before {
    my $self = shift;
    $semaphore->down;
```

```
    $self->SUPER::before();
}

sub after {
    my $self = shift;
    $self->SUPER::after();
    $semaphore->up;
}

sub start {
    my $self = shift;
    $semaphore = Thread::Semaphore->new($self->{Limit_card});
    $self->SUPER::start();
}

1;

################################################
#### parallel.pm
package parallel;

use job;
use jobset_n;

@ISA = (jobset_n);

$alive_threads = ();

sub new {
    my $class = shift;
    my $self = $class->SUPER::new();
    $self->{Parallel_after_all} = sub{};
    return bless $self, $class;
}

sub before {
  my ($self, $job) = @_;
  $self->SUPER::before();
}

sub after {
  my ($self, $job) = @_;
  $self->SUPER::after();
  $self->{Jobset_done}++;
#  if ($self->{Jobset_done} >= $self->{Jobset_card}) {
#      after_all();
#  }
}

sub start {
    my $self = shift;
    jobset::start($self);
    # jobを並列実行
    foreach $wt (@job::alive_threads) {
        $wt->join();
    }
}

sub after_all {
    my $self = shift;
    if ( ref $self->{Parallel_after_all} eq "CODE" ) {
        &{$self->{Parallel_after_all}}($self);
    } else {
        eval($self->{Parallel_after_all});
    }
}

1;

################################################
#### jobset.pm
```

```
#### 09/04/10：一般化バージョン
package jobset;

use job;

$alive_threads = ();

sub identity {
    return $_[0];
}

sub new {
    my $class = shift;
    my $self = {
        "Jobset_jobs" => (),
        "Jobset_done" => 5,
        # 以下はユーザ指定
        "Jobset_names" => (),   # ジョブ名のリスト
        "Jobset_name2args" => ¥&identity,    # string->string|string
        "Jobset_exe" => "",
        "Jobset_before" => '',
        "Jobset_after" => '',
    };
    return bless $self, $class;
}

sub before {
    my ($self, $job) = @_;
}

sub after {
    my ($self, $job) = @_;
}

sub start {
    my $self = shift;
    # jobオブジェクト（ジョブと呼ぶ）作成
    print $self->{Jobset_names} . "¥n";
    foreach my $name (@{$self->{Jobset_names}}) {
        my $exe = $self->{Jobset_exe};
        # 以下変更
        my $args;
        if ( ref $self->{Jobset_name2args} eq "CODE" ) {
            $args = &{$self->{Jobset_name2args}}($name);
        } else {
            $args = eval($self->{Jobset_name2args});
            $args =~ s/%d/$name/;
        }
        #$args =~ s/%d/$i/g;
        push (@{$self->{Jobset_jobs}}, job->new($self, $exe, $args));
    }
    # ジョブ投入を要求
    foreach $j (@{$self->{Jobset_jobs}}) {
        $j->before_do();
    }
}

1;

#####################################################
#### jobset_n.pm
#### 09/04/10：従来のjobset. 一般化バージョンを拡張（＝specialize）して定義
package jobset_n;
use jobset;
@ISA = (jobset);

sub new {
    my $class = shift;
    my $self = $class->SUPER::new();
    $self->{Jobset_n_card} = 0;
    @{$self->{Jobset_names}} = ();
```

```
    return bless $self, $class;
}

sub set_jobset_n_card {
    my ($self, $n) = @_;
    $self->{Jobset_n_card} = $n;
    @{$self->{Jobset_names}} = (1..$n);
}

1;

#####################################################
#### job.pm
package job;

use threads;

sub new {
    my ($class, $addr, $exe, $args) = @_;
    my $self = {
        "Job_addr" => $addr,
        "Job_exe" => $exe,
        "Job_args" => $args,
        "Job_stat" => "ready"
    };
    return bless $self, $class;
}

sub before_do {
    my $self = shift;
    my $addr = $self->{Job_addr};
    $addr->before($self);
    # ジョブ投入
    push (@alive_threads, threads->new(¥&after_do, $self));
}

sub after_do {
    my $self = shift;
    my $addr = $self->{Job_addr};
    my $cmd = "$self->{Job_exe} $self->{Job_args}";
    print "Start $cmd.¥n";
    system ("$cmd");
    $addr->after($self);
}

1;
```