```
package Class::Observable;
use strict;

$Class::Observable::VERSION = 6.66;

use strict 'vars';

sub get_observers {
  my $self = shift;
  my @o = $self->_class_observers;
  push @o, @{$self->direct_observers} if ref $self;
  return _uniq(@o);
}

sub direct_observers_object {
  my $self = shift;
  return $self->{"Class::Observable::_observers"} ||= [];
}

sub direct_observers_class {
  my $class = shift;
  no strict 'refs';
  return \@{"$class\::Class_Observable_observers"};
}

sub direct_observers {
  my $self = shift;
  return ref($self) ? $self->direct_observers_object
                    : $self->direct_observers_class;
}

sub _class_observers {
  my $self = shift;
  my $class = ref($self) || $self;
  my @result = @{$class->direct_observers};
  {
    no strict 'refs';
    for (@{"$class\::ISA"}) {
      push @result, $_->_class_observers
        if $_->isa(__PACKAGE__);
    }
  }
  return @result;
}

sub copy_observers {
  my ($from, $to) = @_;
  $to->add_observer($from->get_observers);
}

# sub count_observers { scalar (my @a = $_[0]->get_observers) }
sub count_observers { $_[0]->get_observers }

sub add_observer {
  my ($self) = shift;

  my $o = $self->direct_observers;
  @$o = _uniq(@$o, @_);
}

sub _uniq {
  my %seen;
  return grep !$seen{$_}++, @_;
}

sub delete_observer {
```

```
  my ($self, @observers) = @_;
  my (%remove, @result);
  for (@observers) {
    $remove{$_} = 1;
  }
  @{$self->direct_observers} =
    grep !$remove{$_}, @{$self->direct_observers};
}

sub delete_all_observers {
  my $self = shift;
  my $obs = $self->direct_observers;
  my $N = @$obs;
  @{$obs} = ();
  return $N;
}

BEGIN { *delete_observers = \&delete_all_observers }

sub notify_observers {
  my ($self, @args) = @_;
  for my $o ($self->get_observers) {
    eval { ref($o) eq "CODE" ? $o->($self, @args) : $o->update($self, @args) };
    if ( $@ ) {
      $self->observer_error(
         "Failed to send observation from 'self' to 'o': $@" );
    }

  }
}

sub observer_error {
    shift; die @_, "\n";
}

1;
```