

```

require SDBM_File;
require Tie::Hash;
@ISA = qw( Tie::Hash );

sub TIEHASH {
    my $type = shift;
    my $ref = SDBM_File->new(@_);
    bless {'dbm' => $ref}, $type;
}
sub FETCH {
    my $self = shift;
    my $ref = $self->{'dbm'};
    $ref->FETCH(@_);
}
sub STORE {
    my $self = shift;
    if (defined $_[0]){
        my $ref = $self->{'dbm'};
        $ref->STORE(@_);
    } else {
        die "Cannot STORE an undefined key in Mydbm\n";
    }
}

package main;
use Fcntl qw( O_RDWR O_CREAT );

tie %foo, "Mydbm", "Sdbm", O_RDWR|O_CREAT, 0640;
$foo{'bar'} = 123;
print "foo-bar = $foo{'bar'}\n";

tie %bar, "Mydbm", "Sdbm2", O_RDWR|O_CREAT, 0640;
$bar{'Cathy'} = 456;
print "bar-Cathy = $bar{'Cathy'}\n";

```

## 11.9 THINKING OF CODE REUSE

One strength of Object-Oriented languages is the ease with which old code can use new code. The following examples will demonstrate first how one can hinder code reuse and then how one can promote code reuse.

This first example illustrates a class which uses a fully-qualified method call to access the "private" method BAZ(). The second example will show that it is impossible to override the BAZ() method.

```

package F00;

sub new {
    my $type = shift;
    bless {}, $type;
}
sub bar {
    my $self = shift;
    $self->F00::private::BAZ;
}

package F00::private;

```