```
# create class meta-object using that most perfect of names
our %Some_Class = (         # our() is new to perl5.6
    CData1 => "",
    CData2 => "",
);

# this accessor is calling-package-relative
sub CData1 {
    my $obclass = shift;
    my $class   = ref($obclass) || $obclass;
    no strict "refs";       # to access eponymous meta-object
    $class->{CData1} = shift if @_;
    return $class->{CData1};
}

# but this accessor is not
sub CData2 {
    shift;                  # XXX: ignore calling class/object
    no strict "refs";       # to access eponymous meta-object
    __PACKAGE__ -> {CData2} = shift if @_;
    return __PACKAGE__ -> {CData2};
}
```

In the second accessor method, the __PACKAGE__ notation was used for two reasons. First, to avoid hardcoding the literal package name in the code in case we later want to change that name. Second, to clarify to the reader that what matters here is the package currently being compiled into, not the package of the invoking object or class. If the long sequence of non-alphabetic characters bothers you, you can always put the __PACKAGE__ in a variable first.

```
sub CData2 {
    shift;                  # XXX: ignore calling class/object
    no strict "refs";       # to access eponymous meta-object
    my $class = __PACKAGE__;
    $class->{CData2} = shift if @_;
    return $class->{CData2};
}
```

Even though we're using symbolic references for good not evil, some folks tend to become unnerved when they see so many places with strict ref checking disabled. Given a symbolic reference, you can always produce a real reference (the reverse is not true, though). So we'll create a subroutine that does this conversion for us. If invoked as a function of no arguments, it returns a reference to the compiling class's eponymous hash. Invoked as a class method, it returns a reference to the eponymous hash of its caller. And when invoked as an object method, this function returns a reference to the eponymous hash for whatever class the object belongs to.

```
package Some_Class;
use strict;

our %Some_Class = (         # our() is new to perl5.6
    CData1 => "",
    CData2 => "",
);

# tri-natured: function, class method, or object method
sub _classobj {
    my $obclass = shift || __PACKAGE__;
    my $class   = ref($obclass) || $obclass;
    no strict "refs";   # to convert sym ref to real one
    return \%$class;
}
```