**the switched class**

The **switched** class serves as base class for user classes with switch / option configuration method. It provides facilities for managing options through a simple interface.

For example:

set vehicle [new car -length 4.5 -width 2 -power 100 -fuel diesel]  
puts "my car was running on [switched::cget $vehicle -fuel]"  
switched::configure $vehicle -power 40 -fuel electricity  
puts "but is now running on clean [switched::cget $vehicle -fuel]"

Of course, as you might have guessed, the **car** class is derived from the **switched** class. Let us see how it works:

class car {  
    proc car {this args} switched {$args} {  
        # car specific initialization code here  
        switched::complete $this  
    }  
    ...  
}

The switched class constructor takes the optional configuration option / value pairs as parameters. The switched class layer then completely manages the switched options: it checks their validity, stores their values and provides a clean interface to the user layer configuration setting procedures.

The switched class members available to the programmer are:

**switched**

* complete{}
* *options{}*
* *set-option{}*
* ...
* complete
* -option
* ...

The **complete** procedure is used to tell the switched layer that the derived class object (a car in the examples) is completely built. At that time, the initial configuration of the switched object occurs, using default option values (see **options** procedure) eventually overridden by construction time values, passed at the time of the *new* operator invocation. The complete procedure must be called once only, usually around or at the end of the derived class constructor. *(****Note****: also check the* ***complete*** *data member later in this chapter)*

The **options** procedure must return the configuration description for **all** options that the switched object will accept. It is a pure virtual member procedure and therefore its implementation is **mandatory** in the derived class layer. The procedure must return a list of lists. Each list pertains to a single option and is composed of the switch name, the default value for the option and an optional initial value. For example:

class car {  
    ...  
    proc options {this} {  
        return [list\  
            [list -fuel petrol petrol]\  
            [list -length {} {}]\  
            [list -power {} {}]\  
            [list -width {} {}]\  
        ]  
    }  
    proc set-fuel {this value} {  
        ...  
    }  
    ...  
}

In this case, 4 options are specified: *fuel*, *length*, *power* and *width*. The default and initial values for the *fuel* option are identical and set to *petrol*. For the other options, values are all empty.

For each option, there must be a corresponding set-*option* procedure defined in the derived class layer. For example, since we defined a *fuel* option, there is a set-*fuel* procedure in the car class. The parameters always are the object identifier (since this is not a static procedure, but rather a dynamically defined virtual one), followed by the new value for the option. The set-*option* procedure is only invoked if the new value differs from the current one (a cache scheme for improving performance), or if there is no initial value set in the options procedure for that option.

In the options procedure, if the initial value differs from  the default value or is omitted, then initial configuration is forced and the corresponding set-*option* procedure is invoked by the switched *complete* procedure located at the end of the derived class constructor. For example:

class car {  
    ...  
    proc options {this} {  
        return [list\  
            [list -fuel petrol]\  
            [list -length {} {}]\  
            [list -power 100 50]\  
            [list -width {} {}]\  
        ]  
    }  
    ...  
}

In this case, configuration is forced on the *fuel* and *power* options, that is the corresponding set-*option* procedures will be invoked when the switched object is constructed (see set-*option* procedures documentation below).

For the *fuel* option, since there is no initial value, the set-*fuel* procedure is called with the default value (*petrol*) as argument. For the *power* option, since the initial value differs from the default value, the set-*power* procedure is called with the initial value as argument (*50*).

For the other options, since the initial values (last elements of the option lists) are identical to their default values, the corresponding set-*option* procedures will not be invoked. It is the programmer's responsibility to insure that the initial option values are correct.

The **set-*option*** procedures may be viewed as dynamic virtual functions. There must be one implementation per supported option, as returned by the *options* procedure. For example:

class car {  
    ...  
    proc options {this} {  
        return [list\  
            ...  
            [list -width {} {}]\  
        ]  
    }  
    ...  
    proc set-width {this value} {  
        ...  
    }  
    ...  
}

Since the *-width* option was listed in the options procedure, a *set-width* procedure implementation is provided, which of course would proceed to set the width of the car (and would modify the looks of a graphical representation, for example).

As you add a supported *option* in the list returned by the options procedure, the corresponding set-*option* procedure may be called as soon as the switched object is complete, which occurs when the switched level *complete* procedure is invoked. For example:

class car {  
    proc car {this args} switched {args} {  
        ...  
        switched::complete $this  
   }  
    ...  
    proc options {this} {  
        return [list\  
            [list -fuel petrol]\  
            [list -length 4.5]\  
            [list -power 350]\  
            [list -width 1.8]\  
        ]  
    }  
    proc set-fuel {this value} {  
        ...  
    }  
    proc set-length {this value} {  
        ...  
    }  
    proc set-power {this value} {  
        ...  
    }  
    proc set-width {this value} {  
        ...  
    }  
}

new car

In this case, a new car is created with no options, which causes the car constructor to be called, which in turns calls the switched level *complete* procedure after the car object layer is completely initialized. At this point, since there are no initial values in any option list in the options procedure, the set-fuel procedure is called with its default value of *petrol* as parameter, followed by the set-length call with *4.5* value, set-power with *350* value and finally with set-width with *1.8* as parameter. This is a good way to test the set-option procedures when debugging, and when done, just fill-in the initial option values.

The switched layer checks that an option is valid (that is, listed in the options procedure) but obviously does not check the validity of the value passed to the set-*option* procedure, which should throw an error (for example by using the Tcl error command) if the value is invalid.

The switched layer also keeps track of the options current values, so that a set-*option* procedure is called only when the corresponding option value passed as parameter is different from the current value (see *-option* data members description).

The ***-option*** data member is the option current value. There is one for each option listed in the options procedure. It is a read-only value which the switched layer checks against when an option is changed. It is rarely used at the layer derived from switched, except in the few cases, such as in the following example:

...

proc car::options {this} {  
    return {  
        ...  
        {-manufacturer {} {}}  
        ...  
    }  
}

proc car::set-manufacturer {this value} {}

proc car::printData {this} {  
    puts "manufacturer: $switched::($this,-manufacturer)"  
    ...  
}

In this case, the manufacturer's name is stored at the switched layer level (this is why the set-manufacturer procedure has nothing to do) and later retrieved in the printData procedure.

The **complete** data member (not to be confused with the complete procedure) is a boolean. Its initial value is *false* and it is set to *true* at the very end of the switched complete procedure. It becomes useful when some options should be set at construction time only and not dynamically, as the following example shows:

proc car::set-width {this value} {  
    if {$switched::($this,complete)} {  
        error {option -width cannot be set dynamically}  
    }  
    ...  
}