```
import types

def private(*privates):
  def onDecorator(aClass):
    class onInstance:
      def __init__(self, *args, **kargs):
          self.wrapped = aClass(*args, **kargs)
          self.private_fields=privates
      def __getattr__(self, attr):
          if attr in privates:
             raise TypeError('private attribute fetch: ' + attr)
          else: return getattr(self.wrapped, attr)
      def __setattr__(self, attr, value):
          if attr == 'wrapped': self.__dict__[attr] = value
          elif attr in privates:
             raise TypeError('private attribute change: ' + attr)
          else: setattr(self.wrapped, attr, value)
    return onInstance
  return onDecorator

def make_getter(name):
  def getter(self):
    return self.__dict__[name]
  return getter

def make_setter(name):
  def setter(self, value):
    self.__dict__[name] = value
  return setter

gen_selectors = {'get': make_getter, 'set':make_setter}

def selectors(*fields_to_mask):
   def onDecorator(aClass):
      class SelectorsClass:
        def __init__(self, *args, **kargs):
            self.wrapped = aClass(*args, **kargs)

            for selector_type in fields_to_mask[0].keys():
              for field in fields_to_mask[0][selector_type]:
                 if (field in self.wrapped.private_fields):
                   self.wrapped.wrapped.__dict__[ \
                       selector_type+field[0].upper()+field[1:] \
                   ] = \
                       types.MethodType( \
                         gen_selectors[selector_type](field), \
                         self.wrapped.wrapped \
                       )
                 else:
                   raise TypeError( \
                    'attempt to add a selector for a non private attribute: '
                      + field)

        def __getattr__(self, attr):
            return getattr(self.wrapped, attr)
        def __setattr__(self, attr, value):
            if attr == 'wrapped': self.__dict__[attr] = value
            else: setattr(self.wrapped, attr, value)
      return SelectorsClass
   return onDecorator
```