

gsrubylib

```
require 'debuglog' unless $gs_nodebuglog
require 'pry'       unless $gs_nopry
require 'contracts'
include Contracts
```

if object.in? collection	
if object.not_in? collection	
if object.not_nil?	
str = object.pp_s	
o.define_method(:add) do x,y x + y end	
squares = (1..10).build_hash { n [n, n*n] }	graph
squares.values.mapf(:to_s)	collectf
h = squares.apply_keys { k k.to_s }	
h = squares.apply_values { k k.to_s }	
"foo".indent(4)	
"bar".tabto(4)	
USAGE = %{\n usage: prog [-o dir] -h file...\n where\n -o dir outputs to DIR\n -h prints this message\n}.trim("\n")	
StringIO.string { o o.puts "Hi..." }	
class Person\n attr_predicate :young\n attr_predicate_rw :successful\nend	

Labels

```
Result = GS::Label.create(:win,\n                          :lose,\n                          :draw)\n\nresult = Result.lose\nresult.to_s / to_sym / symbol / inspect\nresult == Result[:lose]
```

Labels are safer than symbols because they guard against misspellings. They also “inspect” nicely.

Values

```
Person =\n  GS::Value.create(name: String, age: Nat, married: Bool) do\n    default married: false\n    ... other methods ...\n  end\n\np = Person[name: 'John', age: 25]      or Person.new(...)\n                                       or Person['John', 25]\n\np.name; p.age; p.married; p.married?\np[:name], p[:age]                     etc.\n\np.with(age: 26, married: true)\n\np.attributes\np.values\np.values(:name, :married)\n\ne = p.upgrade(Employee, title: 'Nurse', salary: 58400)\np = e.downgrade(Person)\n\nPerson.info      # "Person[name: String, ...]"
```

Values are read-only structs with Contracts built-in, default values, predicate methods, copy-constructors (*with*), transformers (*upgrade*, *downgrade*).

They combine type safety, state safety and convenience.

