Patterns for pragmatists
Primož Gabrijelčič
@thedefphigeek

PATTERNS FOR PRAGMATISTS
About me

Primož Gabrijelčič

http://primoz.gabrijelcic.org

• programmer, MVP, writer, blogger, consultant, speaker
• Blog http://thedelphigeek.com
• Twitter @thedelphigeek
• Skype gabr42
• LinkedIn gabr42
• GitHub gabr42
• SO gabr
CompareValue for booleans

CompareValue function is incredibly practical when you are writing comparers (functions that determine how some data structure is ordered). System.Math and System.StrUtils define a bunch of functions that can be used to compare integers, doubles, strings... There's, however, no CompareValue for booleans.

A CompareValue function compares two parameters, traditionally named left and right, and returns 0 if they are the same, –1 if left is smaller and 1 if right is smaller.

If we use the usual ordering of false < true, we can write the missing function as follows:

```pascal
function CompareValue(left, right: boolean): integer; overload;
begin
  if left < right then
    Result := -1
  else if left > right then
    Result := 1
  else
    Result := 0;
end;
```

Your task for today – if you choose to accept it – is: Write this function without any if statements.

Read more »
Books

http://tiny.cc/pg-ppotl
http://tiny.cc/pg-dhp
http://tiny.cc/pg-dpd
PATTERNS FOR PRAGMATISTS
“Pragmatic”

• Merriam-Webster:
  1. relating to matters of fact or practical affairs often to the exclusion of intellectual or artistic matters: **practical as opposed to idealistic**
     • pragmatic men of power have had no time or inclination to deal with ... social morality
       — K. B. Clark
  2. relating to or being in accordance with philosophical pragmatism
Design patterns

• Pattern = template for a solution
• Pattern = common vocabulary
• Pattern ≠ recipe

• architectural patterns > design patterns > idioms
• design patterns ≠ design principles (SOLID, DRY ...)
• “Classical” design patterns = “Design Patterns: Elements of Reusable Object-Oriented Software”
  • Very specific to object-oriented programming
  • Somewhat specific to C++
  • Better solutions exist for some of them

• Don’t use design patterns to architect the software
  • Use them to solve specific problems

• Design patterns are a tool, not a goal!
Delphi idioms

- Object creation and destruction
- Assign and AssignTo
- [Attributes]
- Iterating with for..in
- Helpers
- Actions
- And more ...
Architectural patterns

• Model-View-Controller, ...
• Domain driven design
• Multilayered architecture
• Data warehouse
• ...
Design principles

• SOLID  Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion
• DRY    Don’t repeat yourself
• KISS   Keep it simple stupid
• YAGNI  You ain’t gonna need it
• SoC    Separation of concerns
• NIH/PFE Not invented here / Proudly found elsewhere
• Creational patterns: *delegation*
  • Creating objects and groups of objects

• Structural patterns: *aggregation*
  • Define ways to compose objects

• Behavioral patterns: *communication*
  • Define responsibilities between objects

• Concurrency patterns: *cooperation*
  • Make multiple components work together
CREATIONAL PATTERNS
- Abstract factory
- Builder
- Dependency injection
- Factory method
- Lazy initialization
- Multiton

- Object pool
- Prototype pattern
- Resource acquisition is initialization (RAII)
- Singleton

Not covered in the book. Covered in more detail in this presentation.
A country should always have one and only one president/queen/king/head of state/... at any time. She or he is a singleton.

• Don’t use (true) singletons!
  • They cause problems with unit testing
  • They are not configurable

• Better approaches
  • Global factory
  • Global variable 😞
  • Injection 😊
Whenever I go somewhere with a car, I have to take into account a small possibility that the car will not start. If (and only if) that happens, I call my mechanic. That is lazy initialization.

- Simple in a single-threaded program
  ```pascal
  if not assigned(lazyObject) then
    lazyObject := TLazyObject.Create;
  Use(lazyObject);
  
  ```

- A bit trickier in a multi-threaded program
  ```pascal
  Spring.Lazy<T>
  ```
Imagine a kid making cookies out of dough. He can do nothing until he invokes a factory method and says “Give me a cutter”. You provide him with a cookie cutter and he can finally start making cookies. In what shape? That’s your call.

- Factory method = TFunc

Image source: https://sourcemaking.com/design_patterns/factory_method
STRUCTURAL PATTERNS
Structural patterns

- Adapter
- Bridge pattern
- Composite
- Decorator
- Extension object
- Facade

- Flyweight
- Front controller
- Marker
- Module
- Proxy
- Twin

Not covered in the book. Covered in more detail in this seminar.
Marker is a label attached to a product. It is a note on a car dashboard saying “Change oil at 150,000 km”, or a message on a sandwich in a communal kitchen stating “This belongs to me!”

IIImportantCustomer = interface [{'E32D6AE5-FB60-4414-B7BF-3E5BDFECDE64}']
end;

TImportantCustomer = class(TCustomer, IIImportantCustomer)
end;

if Supports(customer, IIImportantCustomer, ignored) then ...

• Attributes are in most cases a better solution
When you are accessing web from inside a business environment, the traffic usually flows through a http filtering and caching proxy. This software catches all http requests generated in browsers and other applications and then decides whether it will forward request to the target site, return the result from the cache, or deny the request if the site is on the blocked list.

- Protection proxy
- Remoting proxy
- Lazy initialization proxy
- Mocking proxy
- Logging proxy
- Locking/serialization proxy
BEHAVIORAL PATTERNS
Behavioral patterns

- Blackboard
- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Null object

- Observer (Publish/Subscribe)
- Servant
- Specification
- State
- Strategy
- Template method
- Visitor

Not covered in the book.

Covered in more detail in this seminar.
Most modern operating systems know the concept of a null device. For example, NUL on Windows and /dev/null on Unix and similar systems are devices which are empty if we read from them. They will also happily store any file you copy onto them, no matter what the size. You'll never be able to retrieve that file, though, as the null device stays empty no matter what you do with it.

- Replace ‘if assigned’ code with ‘do-nothing’ objects/interfaces
  - Null object ≠ nullable object
If you subscribe to a magazine, you don't go to the publisher every day to check if new edition is ready. Rather, you wait until the publisher sends you each issue.

- Also known as Publish-Subscribe
- Direct execution of the notification vs. messaging
- Optional *granularity*
  - Usually indicates that the object is too complex (SRP!)
- Live Bindings: TComponent.Observers
- Spring4D: Multicast events
  - Event<T>
CONCURRENCY PATTERNS
Concurrent patterns

- Active object
- Binding properties
- Blockchain pattern
- Compute kernel
- Double-checked locking
- Event-based asynchronous
- Future
- Guarded suspension
- Join
- Lock

Not covered in the book.

- Lock striping
- Messaging
- Monitor object
- Optimistic locking
- Pipeline (Staged processing)
- Reactor
- Read-write lock
- Scheduler
- Thread pool
- Thread-specific storage

Covered in more detail in this seminar.
Double-checked locking

When you are changing lanes in a car, you check the traffic around you, then turn on indicators, check the traffic again, and then change the lane.

• Faster access to code that is almost never used
• Shared object creation in a multi-threaded program
  • Lazy initialization
  • Spring4D / TLazy
In modern version control systems, such as SVN and git, you don’t lock a file that you want to change. Rather, you modify the file, commit a new version to the version control system, and hope that nobody has modified the same lines of the same file in the meantime.

• Even faster initialization of a shared object
  • Provided that we don’t care if we create the object twice
  • Spring4D / TLazyInitializer
If you play chess on the Internet, you are not sharing a chessboard with your partner. Instead of that, each of you has its own copy of the chessboard and figures and you synchronize the state between the two copies by sending messages (representing the piece moves) to each other.

- Windows messaging
- Queue and Synchronize
- Custom solutions
  - Example: threaded queue + polling
Q&A