DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

VCL WORLD AFTER CLEAN CODE

Bogdan Polak

wintech italia

embarcadero reseller partner

BOGDAN POLAK

BSC POLSKA

bogdan.polak@bsc.com.pl
@bogdanpolak

PIACENZA EDITION

DelphiDay italian conference

2019 - XVIII Edizione

wintech italia

AGENDA

1. Clean Code - quick review
2. OOP introduction + TComponent & TAction
3. Law of Demeter
4. Workshop: Extracting VCL OnClick

CLEAN CODE

Quick update

clean code

→ General rules

- Conventions
- KISS (simple stupid)
- Boy scout rule
- Root cause analysis

Boy scout rule

Leave the campground cleaner than you found it

Icons made by Freepik from www.flaticon.com is licensed by CC 3.0 BY

CLEAN CODE

- Design
  - switch/case ⇒ polymorphism
  - Separate multi-threading code
  - Prevent over-configurability.
  - Use dependency injection.
  - Law of Demeter

clean code

→ Understandability

- ◆ Be consistent
- ◆ Explanatory variables.
- ◆ Boundary conditions in one place.
- ◆ Primitive type ⇒ dedicated value objects
- ◆ Avoid logical dependency (in the same class)
- ◆ Avoid negative conditionals

CLEAN CODE

- Names
    - Descriptive
    - Meaningful
    - Pronounceable
    - Searchable
    - Magic numbers ⇒ constants
    - Avoid encodings (prefixes)

CLEAN CODE

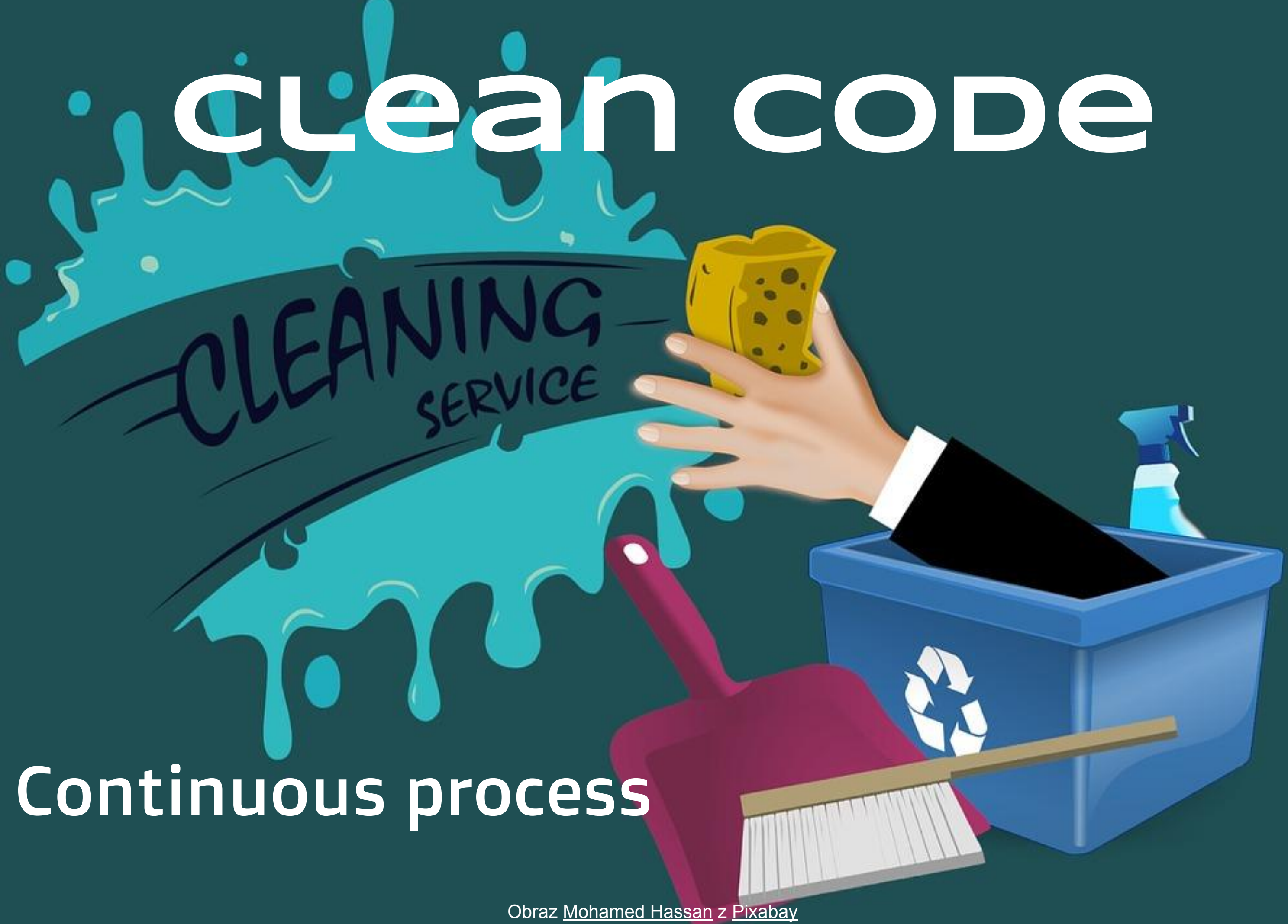
→ Functions / Methods

- ◆ Small
- ◆ One thing
- ◆ Fewer arguments
- ◆ No side effects
- ◆ No flag arguments (enumerable types)

clean code

- More
  - Comments rules
  - Source code structure
  - Objects and data structures
- See more:
  - https://gist.github.com/wojteklu/73c6914cc446146b8b533c0988cf8d29

Clean code



- → Continuous process

Obraz Mohamed Hassan z Pixabay

OOP Introduction

OOP Principles

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism
- Read more:
  - Article - How to explain object-oriented programming concepts to a 6-year-old
  - https://bit.ly/2XbiKyU freeCodeCamp(🔥)

COMPONENT

- Why TComponent
  - Memory management (Owner)
  - Notification system
    - Operations: Insert, Remove
    - Notify sibling and child components

component

- → TComponent in Action

☝ 🎓

ACTION

- Why TAction?
  - TComponent
  - Easy to bind
  - OnUpdate event
  - Caption, ShortCut, Icon index, etc

ACTION

→ TAction in OOP

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

ACTION

→ TAction in Action

☝️ 🎓

Law of Demeter

LOD

- → The Law of Demeter
- → Principle of Least Knowledge

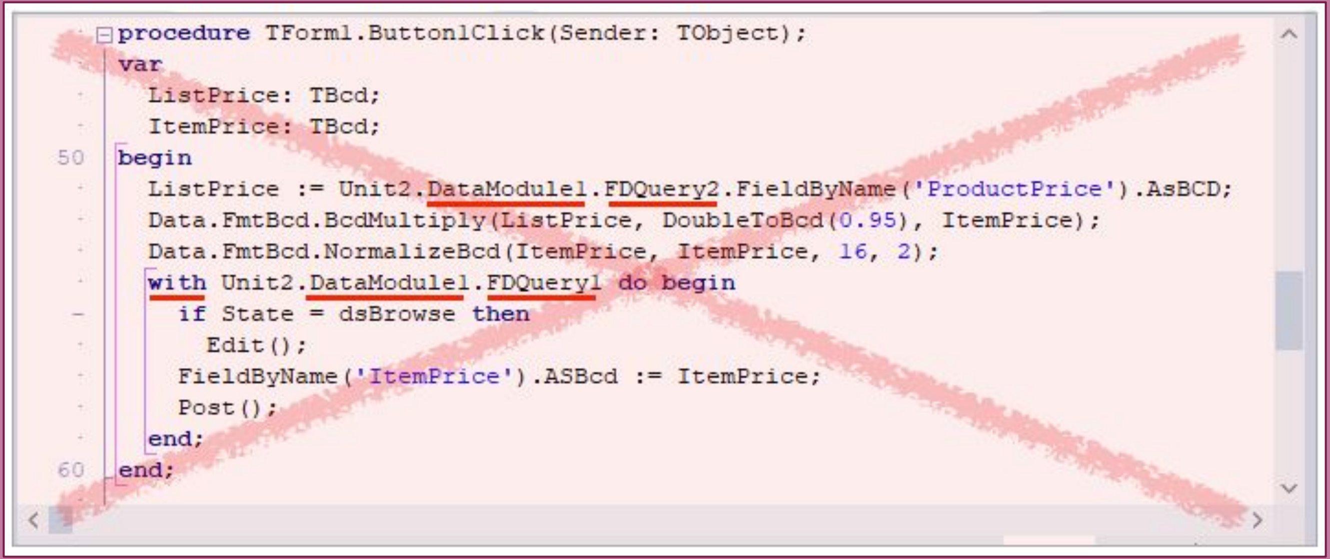
LOD

- → Manage complexity
- → Decrease coupling
- → Classes
- → Narrow view
- → Interact with a few other classes
- → Communicate with its immediate friends

Friends

- Objects
  - This (Self)
  - Passed in the parameters
  - Owned (created inside)
  - Direct components (stored in this class)

VCL Sample



```
procedure TForm1.Button1Click(Sender: TObject);
var
  ListPrice: TBcd;
  ItemPrice: TBcd;
begin
  ListPrice := Unit2.DataModule1.FDQuery2.FieldByName('ProductPrice').AsBCD;
  Data.FmtBcd.BcdMultiply(ListPrice, DoubleToBcd(0.95), ItemPrice);
  Data.FmtBcd.NormalizeBcd(ItemPrice, ItemPrice, 16, 2);
  with Unit2.DataModule1.FDQuery1 do begin
    if State = dsBrowse then
      Edit();
    FieldByName('ItemPrice').ASBcd := ItemPrice;
    Post();
  end;
end;
```

LOD

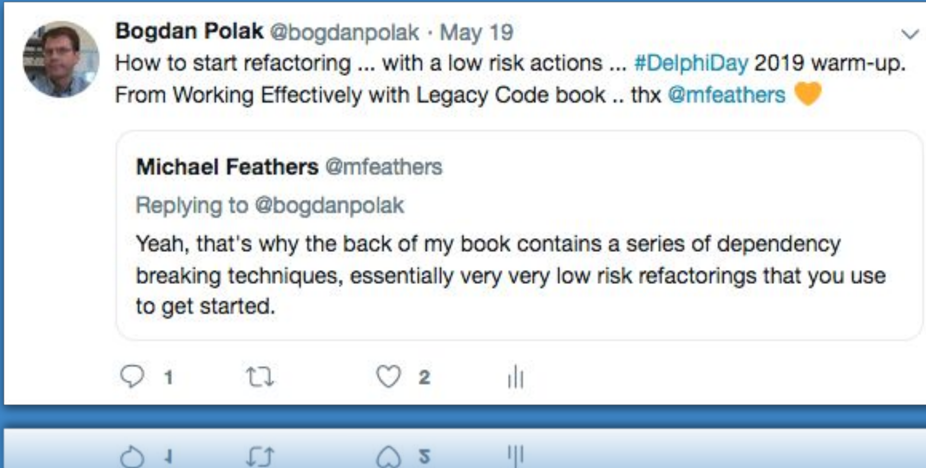
→ LoD in Action (with TDataModule)

☝️ 🎓

Extracting onclicks

warning!

- → Extracting is dangerous!
- → Covered with unit tests
- → One simple step at a time
- → How to start?



Bogdan Polak @bogdanpolak · May 19

How to start refactoring ... with a low risk actions ... #DelphiDay 2019 warm-up. From Working Effectively with Legacy Code book .. thx @mfeathers 🧡

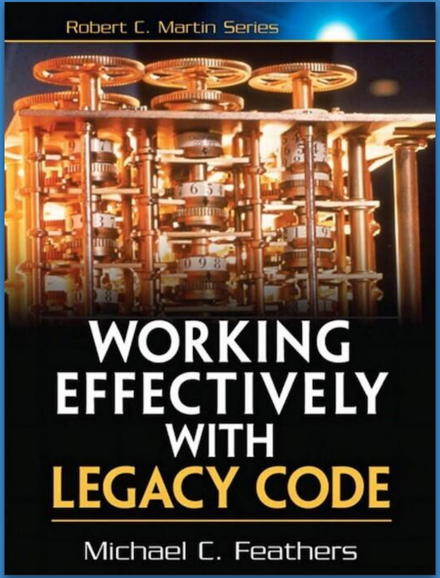
Michael Feathers @mfeathers

Replying to @bogdanpolak

Yeah, that's why the back of my book contains a series of dependency breaking techniques, essentially very very low risk refactorings that you use to get started.

First Book

- 2004
- Part III
  - To get started
  - Chapter 25
  - Dependency-Breaking Techniques



LOW risk

- Break Out Method
  - Move long method to a new class
- Adapt Parameter
  - Use Adapter Pattern for break dependencies
  - Method is dependent on very heavy object (interface)
- Encapsulate Global References
  - Change global data structures into class

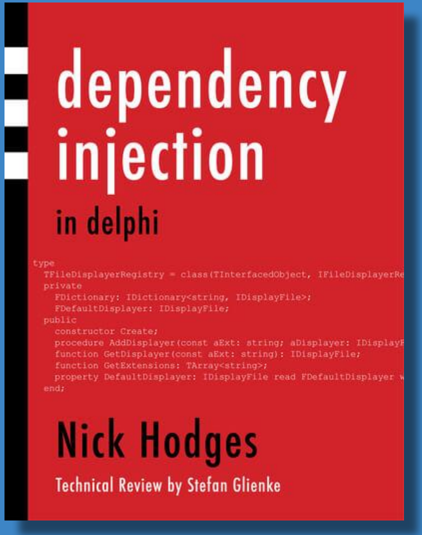
SECOND BOOK

- → 1st edition 1999
- → 2nd edition Dec 2018
  - ◆ Small steps covered by unit test safety net

The Addison-Wesley Signature Series
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
—M. Fowler (1999)
A Martin Fowler Signature Book
Refactoring
Improving the Design of Existing Code
Martin Fowler
with contributions by
Kent Beck
Second
Refactoring
Second Edition
Fowler

Injection

- → Dependency injection
- → Injection via
  - ◆ Constructor
  - ◆ Property
  - ◆ Method
- → Buy & read:
  - ◆ https://leanpub.com/dependencyinjectionindelphi



dependency injection

in delphi

```
type
  TFileDisplayerRegistry = class(TInterfacedObject, IFileDisplayerRe
  private
    FDictionary: IDictionary<string, IDisplayFile>;
    FDefaultDisplayer: IDisplayFile;
  public
    constructor Create;
    procedure AddDisplayer(const aExt: string; aDisplayer: IDisplayF
    function GetDisplayer(const aExt: string): IDisplayFile;
    function GetExtensions: TArray<string>;
    property DefaultDisplayer: IDisplayFile read FDefaultDisplayer w
  end;
```

Nick Hodges

Technical Review by Stefan Glienke

BOOKS Manager

→ Extraction in action

☝️🎓

That's all Folks!