Object Oriented Programming OOP IEC61131-3 Youtube Course by Runtimevic

Object Oriented Programming OOP IEC61131-3 PLC Youtube Course by Runtimevic.

runtimevic

Copyright © 2023 Víctor Durán.

Table of contents

1. Requeriments 4
2. Introduction 6
3. Types of paradigms 8
4. Concepts Previous 10
   4.1 Type of Data 10
   4.2 Variable types and special variables 11
   4.3 Access modifiers 12
   4.4 Access Modifiers Table 13
5. Classes and Objects 14
   5.1 Classes and Objects 14
   5.2 Function Block 15
   5.3 Object Method 19
   5.4 Object Property 23
   5.5 Inheritance 24
   5.6 THIS pointer 29
   5.7 SUPER pointer 30
   5.8 Interface 31
   5.9 pointer and reference 32
   5.10 Keyword Abstract 33
   5.11 Abstract FB vs Interface 34
   5.12 Fluent Interface 35
   5.13 Interface vs Inheritance 36
   5.14 Further operators 37
6. ExST 38
   6.1 Extended Structured Text 38
7. OOP Principles 39
   7.1 4 Pillars 39
   7.2 Abstraction 40
   7.3 Encapsulation 41
   7.4 Inheritance 42
   7.5 Polymorphism 43
8. SOLID 44
   8.1 SOLID 44
   8.2 SRP - Single Responsibility Principle 45
   8.3 OCP - Open/Closed Principle 46

Copyright © 2023 Víctor Durán.

8.4 LSP - Liskov Substitution Principle 47
8.5 ISP - Interface Segregation Principle 48
8.6 DIP - Dependency Inversion Principle 49
9. UML 50
9.1 UML 50
9.2 Class UML 51
9.3 Relations 52
9.4 StateChart UML 53
10. Types of Design for PLC programming 54
11. Design patterns 55
11.1 Design patterns 55
11.2 Strategy Pattern 56
11.3 The Abstract Factory Pattern 57
11.4 The Visitor Design Pattern 58
12. Libraries 59
13. Links OOP 60
14. TDD 61
14.1 TDD - Test Drive Development 61
14.2 Units Test 62

Copyright © 2023 Víctor Durán.

1. Requeriments

👨‍🎓 Requirements 👩‍🎓 :

The necessary requirements to follow this course would be installed the following softwares:

- 🔗 Beckhoff TwinCAT 3 XAE or the ide of 🔗 Codesys.
- Have a user account in 🔗 GitHub.
- Know the minimum of git or rely on visual tools such as:
- 🔗 GitHub Desktop.
- 🔗 sourcetree
- 🔗 tortoiseGit, etc...
- It would be good to have some previous knowledge of OOP theory, even if they are in other programming languages since they will be extrapolated to the approach of this OOP course IEC61131-3 for PLCs.

1.0.1 STEPS TO START:

- Clone the repository of 🔗GitHub:

  $ git clone https://github.com/runtimevic/OOP-IEC61131-3--Curso-Youtube.git

  or use for example github desktop to clone Github's repository...

- We will find the following folders:
- 🔗 TC3_OOP: Within this folder is the Twincat3 project, with everything that is explained in the YouTube videos...
- 🔗 Ficheros_PLCOpen_XML: Within this folder we will find the exported files in Plcopen XML format so that they can be imported on Twincat3 or in codesys of everything explained on YouTube, since being the standardized Plcopen format can be exported/imported in all brands ofPlcs that follow the standard Plcopen ..., but it is advisable to try to do what is explained from scratch to practice and assume the concepts explained ...
- The creation of this SSG website is also housed, (Static sites generator) which will be modified as we advance in this course of OOP IC-61131-3 PLC...

Copyright © 2023 Víctor Durán.

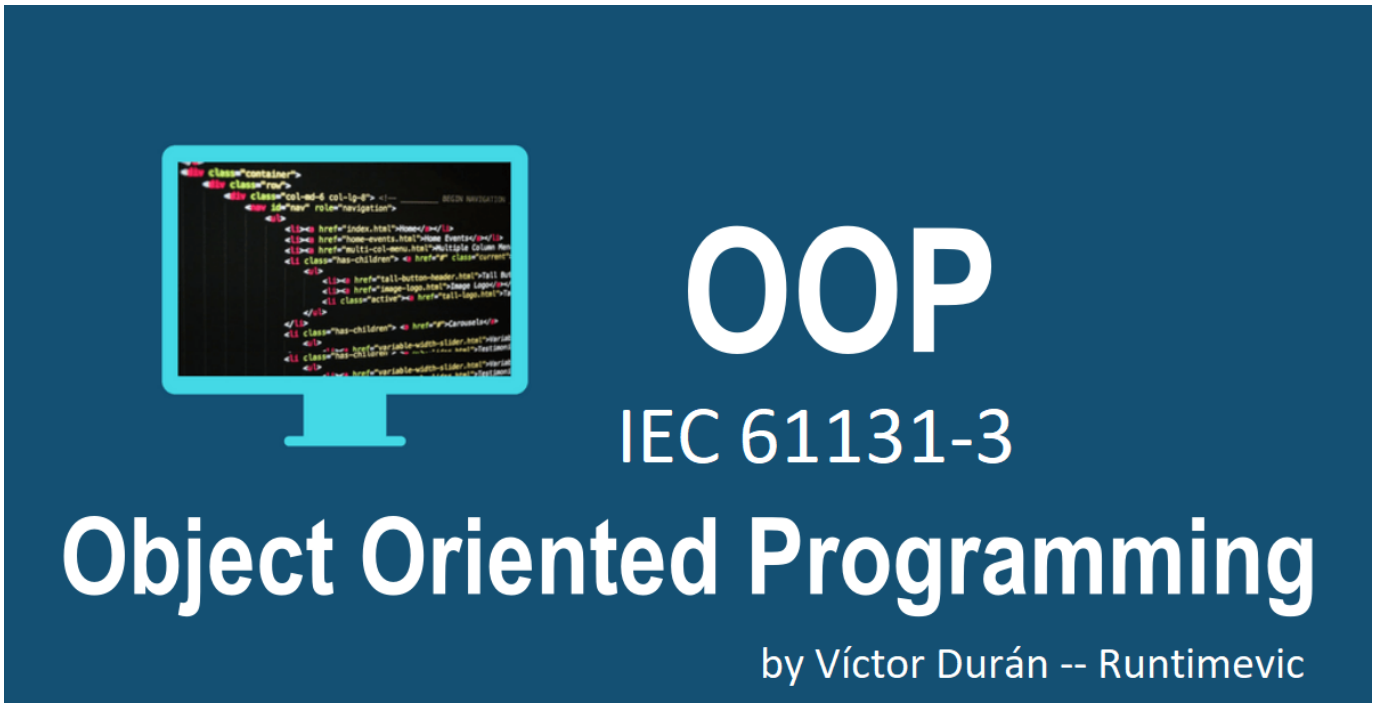
1.0.2 Link to the Youtube Video 001:

- 🔗 001 - OOP IEC 61131-3 PLC -- Introduction to the SSG documentation page, repository...

Copyright © 2023 Víctor Durán.

2. Introduction

📖 Object -oriented programming course YouTube - OOP:



by Runtimevic -- Víctor Durán Muñoz.

2.0.1 What is Oop?

- It is a paradigm that makes use of objects for software construction.

  . What is a paradigm?
- It has different interpretations, it can be a **model**, **example** o **pattern.**
- Is a **form** o o **style** to program.
- It seeks to capture reality towards the code.

2.0.2 How to think about objects?

- To focus on **Something of reality.**
- Details its **attributes, (properties)**

Copyright © 2023 Víctor Durán.

- Details its **behaviors (METHODS)**

```
📱 Example: (Mobile-Smartphone Telephone)

. What attributes (Properties) We recognize?
    - color.
    - brand.
. What can be done? (METHODS)
    - Make calls.
    - Internet.
```

```
🚗 Example: (Coach)

. What attributes (Properties) We recognize?
    - color.
    - brand.
. What can be done? (Metodos)
    - drive.
    - curb.
    - speed up.
```

2.0.3 Links:

- 🔗 Codesys admit OOP
- 🔗 Beckhoff TwinCAT 3 admit OOP
- 🔗 Why Object Oriented PLC Programming is Essential for Industrial Automation

---

2.0.4 link to Video Youtube 001:

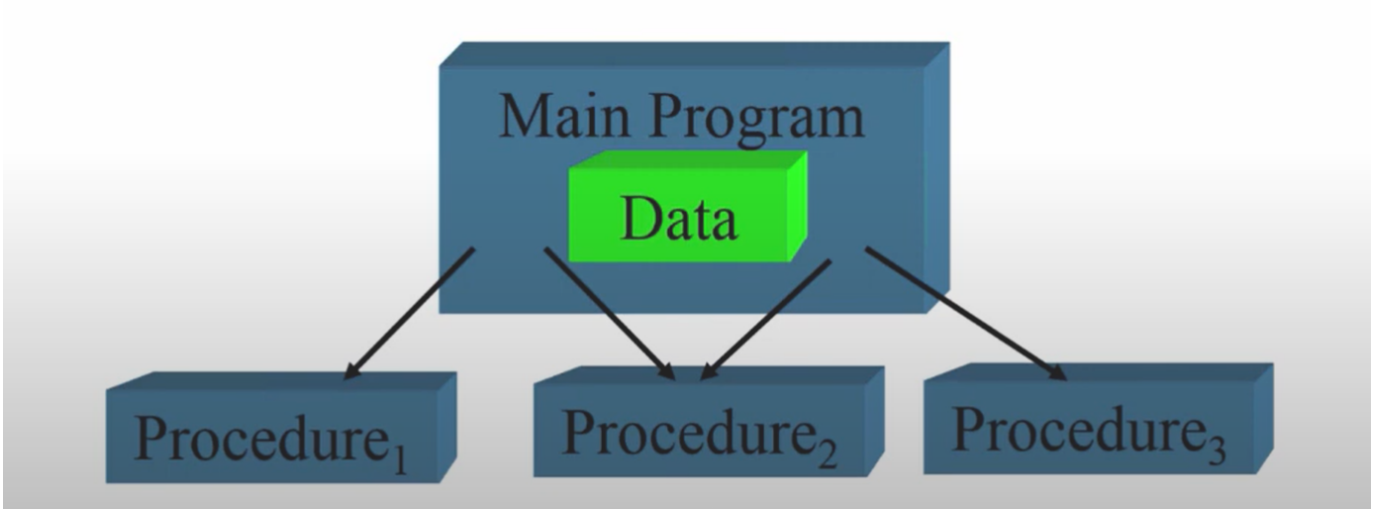
- 🔗 001 - OOP IEC 61131-3 PLC -- Introduction to the SSG documentation page, repository...

Copyright © 2023 Víctor Durán.

3. Types of paradigms

3.0.1 Types of Paradigms:

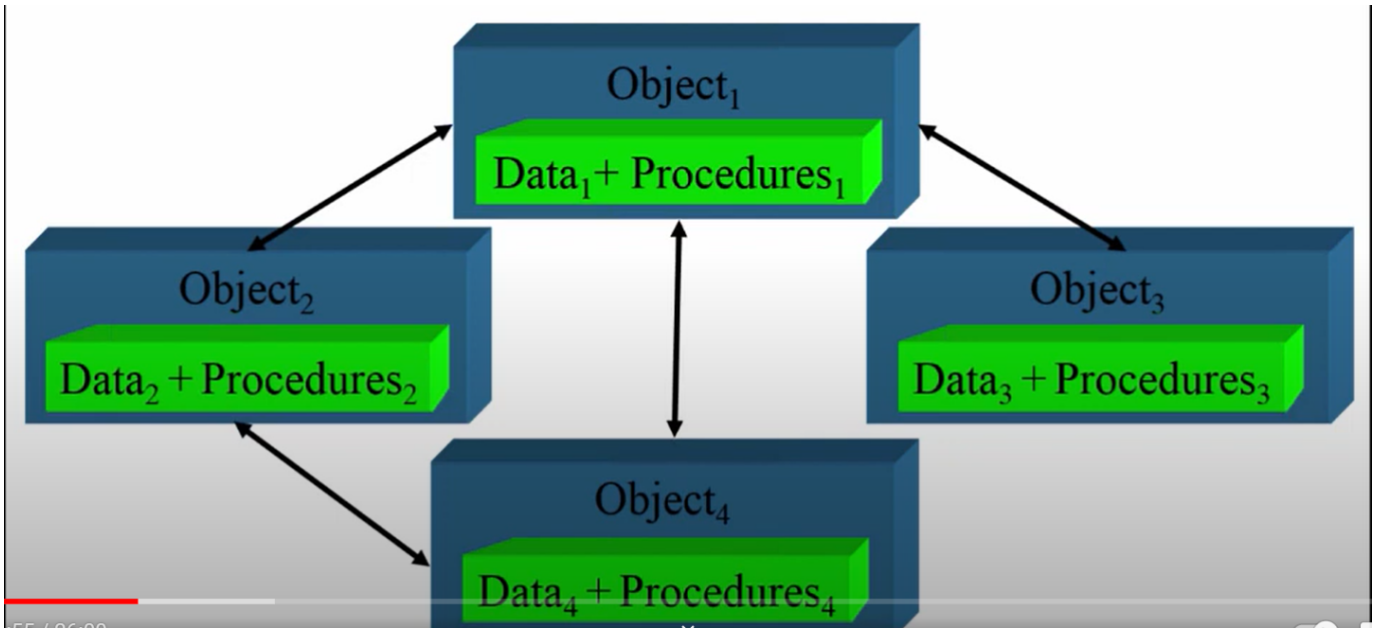
- Imperative -- (**Instructions to follow ** To solve a problem).
- Declarative -- (Se ** focuses on the problem ** to solve).
- Structured -- (The solution to a problem follows ** a sequence from start to finish **).
- Functional -- (Divide the problem into various solutions that will be executed by the ** declared functions **). Procedural programming or procedure programming is a programming paradigm.Many times it is applicable both in low -level programming languages and in high -level languages.In the event that this technique is applied in high -level languages, it will receive the name of functional programming.
- They are called separate routines from the main program
- Most global data -> No protection.
- The procedures can usually be independent -> bad reuse of the code.



- Object -oriented -- Build ** objects based on objects**.

```
wikipedia:
Object -oriented programming is a programming paradigm
Based on the concept of "objects", which can contain data and code.
The data is in the form of fields and the code is in the form of procedures.
```


Copyright © 2023 Víctor Durán.



3.0.2 ADVANTAGES OF PROGRAMMING with OOP:

- routines and data are combined in an object -> encapsulation.
- Methods/Properties -> Interface defined for calls and data access.

3.0.3 Link to the Youtube Video 002:

- 🔗 002 - OOP IEC 61131-3 PLC -- Clase y Objeto
- 🔗 003 - OOP IEC 61131-3 PLC -- Clase y Objeto

Copyright © 2023 Víctor Durán.

4. Concepts Previous

4.1 Type of Data

Copyright © 2023 Víctor Durán.

4.2 Variable types and special variables

Copyright © 2023 Víctor Durán.

4.3 Access modifiers

Copyright © 2023 Víctor Durán.

4.4 Access Modifiers Table

Copyright © 2023 Víctor Durán.

5. Classes and Objects

5.1 Classes and Objects

Copyright © 2023 Víctor Durán.

5.2 Function Block

5.2.1 Function Block

Copyright © 2023 Víctor Durán.

5.2.2 Function Block Access Modifiers

Copyright © 2023 Víctor Durán.

5.2.3 Function Block Declaration variables

Copyright © 2023 Víctor Durán.

5.2.4 Constructor and Destructor

Copyright © 2023 Víctor Durán.

5.3 Object Method

5.3.1 Method

Copyright © 2023 Víctor Durán.

5.3.2 Method access modifiers

Copyright © 2023 Víctor Durán.

5.3.3 Method Declaration of variables

Copyright © 2023 Víctor Durán.

5.3.4 Method return variable types

Copyright © 2023 Víctor Durán.

5.4 Object Property

Copyright © 2023 Víctor Durán.

5.5 Inheritance

5.5.1 Inheritance Function Block

Inheritance function block:

The functions blocks are an excellent medium to maintain the sections of the program separated from each other.This improves the structure of the software and significantly simplifies reuse.Previously, expanding the functionality of an existing block of functions was always a delicate task.This meant modifying the code or programming a new block of functions around the existing block (that is, the existing functions block was effectively embedded within a new block of functions).In the last case, it was necessary to create all the input variables again and assign them to the input variables for the existing functions block.The same was required, in the opposite, for the output variables.

Twincat 3 and Codesys (IEC61131-3) introduce the concept of inheritance.Inheritance is one of the fundamental principles of object -oriented programming.The inheritance implies deriving a new block of functions from an existing block of functions.Next, you can expand the new block.To the extent allowed by the access specifiers of the main functions block, the new block of functions inherits all the properties and methods of the main functions block.Each block of functions can have any number of blocks of secondary functions, but only a block of main functions.The derivation of a block of functions is produced in the new statement of the Functions Block.The name of the new block of functions is followed by the keyword extended followed by the name of the main functions block.For example:

```
FUNCTION_BLOCK PUBLIC FB_NewEngine EXTENDS FB_Engine
```

The new block of derived functions (FB_NewEngine) He has all the properties and methods of his father (FB_Engine). However, the methods and properties are only inherited when the access specifier allows it.

The secondary functions block also inherits all variables **local**, **VAR_INPUT** , **VAR_OUTPUT** y **VAR_IN_OUT** of the main functions block.This behavior cannot be modified by access specifiers.

If the methods or properties of the main functions block have been declared as a protect, the secondary functions block (FB_Newengine) will be able to access them, but not from outside FB_NewEngine .

The inheritance applies only to the Pou of type function_block.

ACCESS SPECIFIERS:

Function_Block, Function or Program statements may include an access specifier.This restricts access and, where appropriate, the ability to inherit.

- PUBLIC:

Anyone can call or create an instance of the Pou.In addition, if the POU is a function_block, it can be used for inheritance.Restrictions are not applied.

- INTERN:

The Pou can only be used within its own name space.This allows the POU to be available only within a certain library, for example.

- FINAL:

The function_block cannot serve as a main block of functions.The methods and properties of this Pou cannot be inherited.Final is only allowed for pou of the type function_block.

The default configuration where no access specifier is defined is public.Private and Protected Access Specifiers are not allowed in Pou's statements.

If you plan to use the inheritance, the declaration of the Functions Block will have the following structure:

```
FUNCTION_BLOCK <Access specifier> <Name> EXTENDS <Name basic function block>
```


Copyright © 2023 Víctor Durán.

OVERWHELMING METHODS:

The new function_block FB_Newengine, which is derived from FB_ENGINE, may contain additional properties and methods.For example, we can add Gear property.This property can be used to consult and change the current march.It is necessary to configure getters and setters for this property.

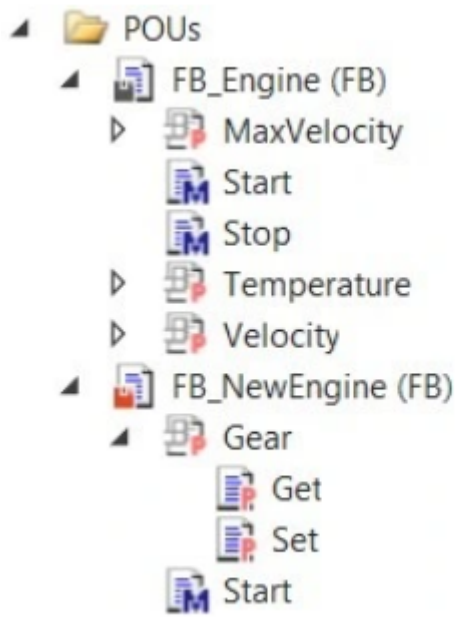
However, we must also ensure that the parameter NGear of the Start () method is passed to this property.Because the FB_Engine main functions block has no access to this new property, a new method must be created exactly the same parameters in FB_Newengine.We copy the existing code to the new method and add new code so that the NGEAR parameter passes to Gear Property.

```
1  METHOD PUBLIC Start
2  VAR_INPUT
3    nGear : INT := 2;
4    fVelocity : LREAL := 8.0;
5  END_VAR
6
7  IF (fVelocity < MaxVelocity) THEN
8    velocityInternal := fVelocity;
9  ELSE
10   velocityInternal := MaxVelocity;
11 END_IF
12 Gear := nGear; // new
```

Line 12 copy the NGEAR parameter to the Gear Property.

When a method or property that is already present in the main functions block is redefined within the secondary functions block, this is called overwhelming.The FB_Newengine Function Block overwrites the Start () method.

Therefore, FB_Newengine has the new gear property and overwrites the Start () method.




```
1  fbNewEngine.Start(1, 7.5);
```

Call the Start () method in FB_Newengine, since this method has been redefined (overwhelming) in FB_Newengine.

While:

```
1  fbNewEngine.Stop();
```

Call the stop () method from FB_Engine.The Stop () method has been inherited by FB_Newengine of FB_ENGINE.

Copyright © 2023 Víctor Durán.

INHERITANCE LINKS FUNCTION BLOCK:

- 🔗 stefanhenneken.net,iec-61131-3-methods-properties-and-inheritance
- 🔗 Simple Codesys OOP - Inheritance
- 🔗 TC11.Beckhoff TwinCAT3 Function Block Extend.JP

LINK TO THE YOUTUBE VIDEO 007:

- 🔗 007 - OOP IEC 61131-3 PLC -- Inheritance FB

Copyright © 2023 Víctor Durán.

5.5.2 Inheritance Structure

Copyright © 2023 Víctor Durán.

5.5.3 Inheritance Interface

Copyright © 2023 Víctor Durán.

5.6 THIS pointer

Copyright © 2023 Víctor Durán.

5.7 SUPER pointer

Copyright © 2023 Víctor Durán.

5.8 Interface

Copyright © 2023 Víctor Durán.

5.9 pointer and reference

Copyright © 2023 Víctor Durán.

5.10 Keyword Abstract

Copyright © 2023 Víctor Durán.

5.11 Abstract FB vs Interface

Copyright © 2023 Víctor Durán.

5.12 Fluent Interface

Copyright © 2023 Víctor Durán.

5.13 Interface vs Inheritance

Copyright © 2023 Víctor Durán.

5.14 Further operators

Copyright © 2023 Víctor Durán.

6. ExST

6.1 Extended Structured Text

EXST - Extended Structured Text:

- 🔗 Structured Text and Extended Structured Text (ExST), infosys.beckhoff.com

Copyright © 2023 Víctor Durán.

7. OOP Principles

7.1 4 Pillars

Copyright © 2023 Víctor Durán.

7.2 Abstraction

Copyright © 2023 Víctor Durán.

7.3 Encapsulation

Copyright © 2023 Víctor Durán.

7.4 Inheritance

Copyright © 2023 Víctor Durán.

7.5 Polymorphism

Copyright © 2023 Víctor Durán.

8. SOLID

8.1 SOLID

Copyright © 2023 Víctor Durán.

8.2 SRP - Single Responsibility Principle

Copyright © 2023 Víctor Durán.

8.3 OCP - Open/Closed Principle

Copyright © 2023 Víctor Durán.

8.4 LSP - Liskov Substitution Principle

Copyright © 2023 Víctor Durán.

8.5 ISP - Interface Segregation Principle

Copyright © 2023 Víctor Durán.

8.6 DIP - Dependency Inversion Principle

Copyright © 2023 Víctor Durán.

9. UML

9.1 UML

Copyright © 2023 Víctor Durán.

9.2 Class UML

Copyright © 2023 Víctor Durán.

9.3 Relations

Copyright © 2023 Víctor Durán.

9.4 StateChart UML

Copyright © 2023 Víctor Durán.

10. Types of Design for PLC programming

10.0.1 Types of design for PLC programming:

Development engineering for OOP programming - Component design, unit, device, object... - Objects are the basic object - oriented programming units. - A component provides services, while an object provides operations and methods.A component can be understood by all, while an object can only be understood by developers. - The units are the smallest code groups that can be maintained and executed independently - Design for unit tests. - UML design.

Units: (Example of units): - I_InputDigital(p_On, p_Off) - I_OutputDigital(M_ON, M_OFF) - I_InputAnalog - I_OutputAnalog - I_Run:(M_Start, M_Stop)

-FBTimer -FCAnalogSensor -FBGenericUnit

!!! points that can be included in the course!!!: - Objects composition (Composition of objects)

- Basic of Structured Text programming Language
- UDT (structures)
- Modular Design
- Polymorphism
- Advanced State Pattern
- Wrappers and Features
- Layered Design
- Final Project covering a real-world problem to be solved using OOP
- Structured text (ST), Extended structured text (ExST)

Copyright © 2023 Víctor Durán.

11. Design patterns

11.1 Design patterns

Copyright © 2023 Víctor Durán.

11.2 Strategy Pattern

Copyright © 2023 Víctor Durán.

11.3 The Abstract Factory Pattern

Copyright © 2023 Víctor Durán.

11.4 The Visitor Design Pattern

Copyright © 2023 Víctor Durán.

12. Libraries

12.0.1 Libraries:

When you develop a project, what do you do when you want to reuse the same program for another project?Probably the most common is to copy and paste.This is fine for small projects, but as the application grows, libraries allow us to administer the functions and functions blocks that we have created.

Through the use of libraries, we can administer the software that we have created in multiple projects.First, it is a fact that different devices will have different functions, but still, there will always be common parts.In the world of software development, that concept of library management is quite common.

12.0.2 What are the advantages of using the library?

- The software is modular, for example, if I have cylinder software, I can use the cylinder library, and if I have registration software, I can use the registration library.
- Each library is tested independently.

12.0.3 Links Libraries:

- 🔗 soup01.com,beckhofftwincat3-library-management
- 🔗 PLC programming using TwinCAT 3 - Libraries (Part 11/18)
- 🔗 help.codesys.com,_cds_obj_library_manager/
- 🔗 help.codesys.com,_cds_library_development_information/
- 🔗 help.codesys.com,_tm_test_action_libraries_addlibrary
- 🔗 CODESYS Webinar Library Management Basics
- 🔗 CoDeSys - How to add libraries and more with Machine Control Studio.

Copyright © 2023 Víctor Durán.

13. Links OOP

13.0.1 Links of OOP:

13.0.2 Mention to the sources links used to carry out this documentation:

Copyright © 2023 Víctor Durán.

14. TDD

14.1 TDD - Test Drive Development

Copyright © 2023 Víctor Durán.

14.2 Units Test

Copyright © 2023 Víctor Durán.