



 **EGF Exercices – Pattern – UC1**
EGF 0.3.0

Benoît Langlois – Thales/EPM

Understanding how to develop patterns with EGF

Correction of the exercices

- ▶ Download the `org.eclipse.egf.usecase.pattern.uc1.zip` file on the [EGF wiki](#)

Prerequisite

- ▶ Installation of Eclipse 3.6 and egf 0.3.0
- ▶ Read the EGF Tutorial
- ▶ Understanding how to create factory components and patterns is explained in the « Eclipse Help/EGF section/Tutorials »

Problem Statement

- ▶ Writing a Jet ClassPattern which displays the names of EClasses contained in an ECore model
- ▶ Additional elements:
 - ▶ For the production plan invocation, using the « Model-Driven pattern strategy » task
 - ▶ Using a reporter
- ▶ Variant: Adding an AttributePattern for the Eattributes. What do you notice?

Learning

- ▶ Understanding the Pattern implementation

Difficulty

- ▶ 1/5

Correction

- ▶ `Pattern_UC1_1_ClassPattern.fcore`

Problem Statement

- ▶ Defining two patterns with the Jet nature, e.g. ClassPattern and ParentPattern
- ▶ ClassPattern generates EClass information
- ▶ Writing the ClassPattern which reuses the orchestration of the ParentOrchestration pattern

Learning

- ▶ Reusing super-pattern orchestration

Difficulty

- ▶ 1/5

Correction

- ▶ `Pattern_UC1_2_Inheritance.fcore`

Problem Statement

- ▶ Be three patterns with the Jet Nature, e.g., Hello, HelloWorld, HelloFriends. HelloWorld inherits from Hello and calls HelloFriends for delegation of its generation.
- ▶ The HelloWorld pattern displays this kind of message for each Eclass of an ecore model:
 - « Hello [class name], and all friends of [Class name] »
- ▶ « Hello » comes from the super-pattern Hello
- ▶ « , and all friends of [Class name] » comes from the Pattern HelloFriends

Learning

- ▶ Customized pattern inheritance
- ▶ Pattern call

Difficulty

- ▶ 2/5

Correction

- ▶ Pattern_UC1_3_InheritanceAndCall.fcore



Problem Statement

- ▶ Defining two patterns with the Jet nature, e.g. ClassPattern and AttributePattern
- ▶ Writing those patterns in order to produce this kind of result:

```
[Begin. "Class1"]  
  [Attribute "A1"]  
  [Attribute "A2"]  
  => 2 attributes  
[End. "Class1"]
```

Learning

- ▶ Pattern strategy
- ▶ Callback
- ▶ Variable management: 1) local variable to a pattern, 2) shared variable between patterns

Difficulty

- ▶ 3/5

Correction

- ▶ Pattern_UC1_4_Callback_StrategyBased.fcore

Problem Statement

- ▶ Defining a pattern with a callback. This callback calls a Java Class.

Learning

- ▶ Java Class Callback

Difficulty

- ▶ 1/5

Correction

- ▶ `Pattern_UC1_5_Callback_WithJavaClass.fcore`

Problem Statement

- ▶ Writing a JetClassPattern and JavaPattern. The JetClassPattern calls the JavaPattern.

Learning

- ▶ Multilingual pattern

Difficulty

- ▶ 1/5

Correction

- ▶ `Pattern_UC1_6_JetPatternCallsJavaPattern.fcore`

Problem Statement

- ▶ Defining two patterns with the Jet nature, e.g. ClassPattern and ForInjectionPattern
- ▶ ClassPattern generates EClass information
- ▶ ForInjectionPattern generates EStructuralFeature information
- ▶ Writing the ClassPattern which uses ForInjectionPattern by injection
- ▶ Clue:
 - ▶ An injection needs to initialize a variable
 - ▶ A query is necessary

Learning

- ▶ Pattern injection

Difficulty

- ▶ 4/5

Correction

- ▶ Pattern_UC1_7_Injection.fcore