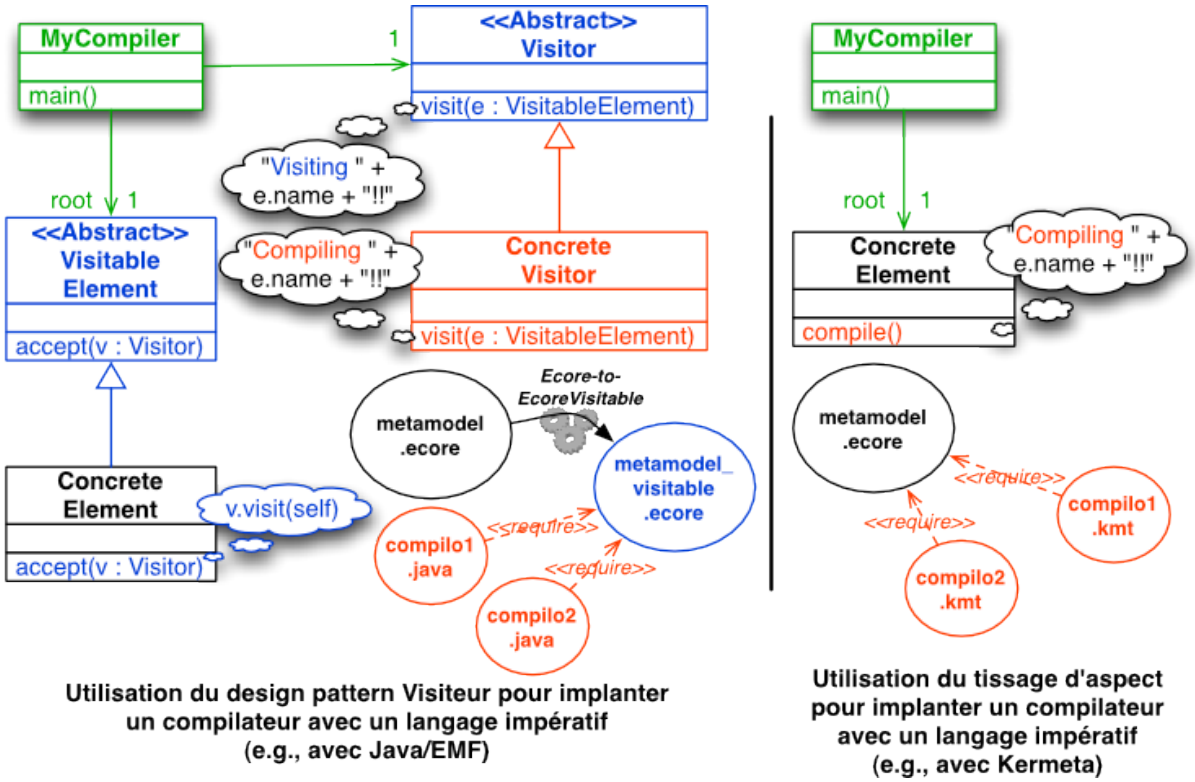
UNIVERSITÉ DE RENNES 1

CAO -- Ingénierie Dirigée par les Modèles

TP3/4 – Génération de code et de documentation

1. Refactoring, Visiteur, Aspects et Kermeta



Utilisation du design pattern Visiteur pour implanter un compilateur avec un langage impératif (e.g., avec Java/EMF)

Utilisation du tissage d'aspect pour implanter un compilateur avec un langage impératif (e.g., avec Kermeta)

Questions

- Quelle(s) technique(s) est envisageable pour manipuler (compiler, interpréter, transformer, etc.) le modèle ?
- Proposez un moyen d'automatiser les tâches récurentes pour rendre manipulable votre métamodèle ?

2. Génération d'une documentation

Nous souhaitons définir un générateur de documentation pour tout modèle conforme à Ecore et permettant de décrire pour chaque classe (`EClass`) ses différents attributs et références ainsi que les classes dont elle hérite (sur le principe de JavaDoc). On veut pouvoir afficher cette documentation dans un navigateur Web. Le générateur devra donc produire le code HTML correspondant.
Le générateur devra être testé sur le modèle du tableur (tableur.ecore) et sur le modèle de l'équipe de recherche (research_team.ecore).

Classe EcoreDoc et méthode main: Créez une nouvelle classe Kermeta appelée EcoreDoc. Dans la méthode `main()` :

- écrivez le nécessaire pour charger le modèle,
- effectuez le parcours du modèle et appelez les méthodes décrites ci-dessous,

- A l'aide de la classe kermeta::io::FileIO, générez les fichiers HTML nécessaires à la visualisation de la documentation[1].

Génération de la page web principale : Générer le code HTML qui permet de créer une page vierge contenant deux frames, l'une de 20% et l'autre de 80%.

index.html

```
<frameset cols="20%,80%">
    <frame src="toc.html" name="TOC" noresize>
    <frame src="contents.html" name="EcoreDoc" noresize>
</frameset>
```

Génération de la table des matières : Ecrivez une opération Kermeta qui génére, pour chaque classe du modèle, un lien HTML de la forme <href= « nom_de_classe.html »>.

toc.html

```
<html>
 <body>
   <h3> Classes of Tableur</h3>
   <CENTER>
   <a href="foo.html" target="EcoreDoc">
     Foo
   </a>
   ...
   </CENTER>
  </body>
</html>
```

Génération du contenu :

- Ecrivez une opération en Kermeta qui, pour une classe donnée, génère le contenu de la documentation. Vous modifierez au besoin la méthode short() que vous avez developpée au TP1.
- Modifiez votre méthode flat() pour générer la partie de la documentation qui correspond à la hiérarchie de classes de la classe en cours de traitement.

contents.html

```
Class Value: <br>
  <UL>
    <LI>val of type Integer</LI>
    <LI>cell of type
     <a href="cell.html">
          Cell
     </a>
    </LI>
  </UL>
```

[1] Rappel : la méthode ***writeTextToFile()*** ne prend en paramètre que des chemins absolus

3. Générateur de code Java

Nous proposons de définir un générateur de code Java pour Ecore que vous appliquerez aux modèles du tableur (tableur.ecore) et de l'équipe de recherche (research_team.ecore)[2]. Nous utiliserons pour cela Kermeta mais aussi KET, un langage de script, simplifiant la génération de texte (cf. documentation jointe).

Chargement du modèle : Créez une nouvelle classe Kermeta appelée Ecore2Java. Dans la méthode `main()` écrivez le nécessaire pour charger le modèle.

Génération des classes : Ecrivez un script KET qui génère l'entête des classes Java correspondant aux classes de votre modèle. Générez également la déclaration du package ainsi que les imports nécessaires à la compilation de la classe. Vous utiliserez FileIO pour générer un fichier pour chaque classe.

Génération des attributs : Ecrivez un second script KET permettant de générer les attributs et les références de chaque classe. Vous inclurez ce script KET dans le premier pour pouvoir générer les classes Java avec les attributs.

Génération des getters/setters : Ecrivez un autre script KET qui génère les getters et les setters pour tous les attributs d'une classe. L'inclure dans le premier pour obtenir votre générateur de code.

```
<%@ket
package="eClass2Class"
require="http://www.eclipse.org/emf/2002/Ecore EcoreUtil.kmt"
using="kermeta::language::structure ecore"
class="EClass2Class"
parameters="aPackage:EPackage, aClass:EClass"
%>

package <%=aPackage.name%>;
import java.util.*;

/**
* TODO javadoc
*/
<%
/*
* Class
*/
var counterSuperClass: Integer init 0
var interfaceList : String init ""
var interfaceName : String init ""
%>

<%if aClass.~abstract == true then%>public abstract class<%else%>public class <%end%>
<%=aClass.name%>{}
```

[2] Il s'agit ici de faire un générateur similaire (et simplifié !) à celui d'EMF pour Ecore ou à celui de MagicDraw pour UML.