An introduction to MDR : The Model Driven approach

Frédéric Fondement

Software Engineering Lab

Swiss Federal Institute of Technology Lausanne

Switzerland

December 2, 2003

Contents

- About the MOF
- About JMI
- The MDR tool
- Demo
- Outlook

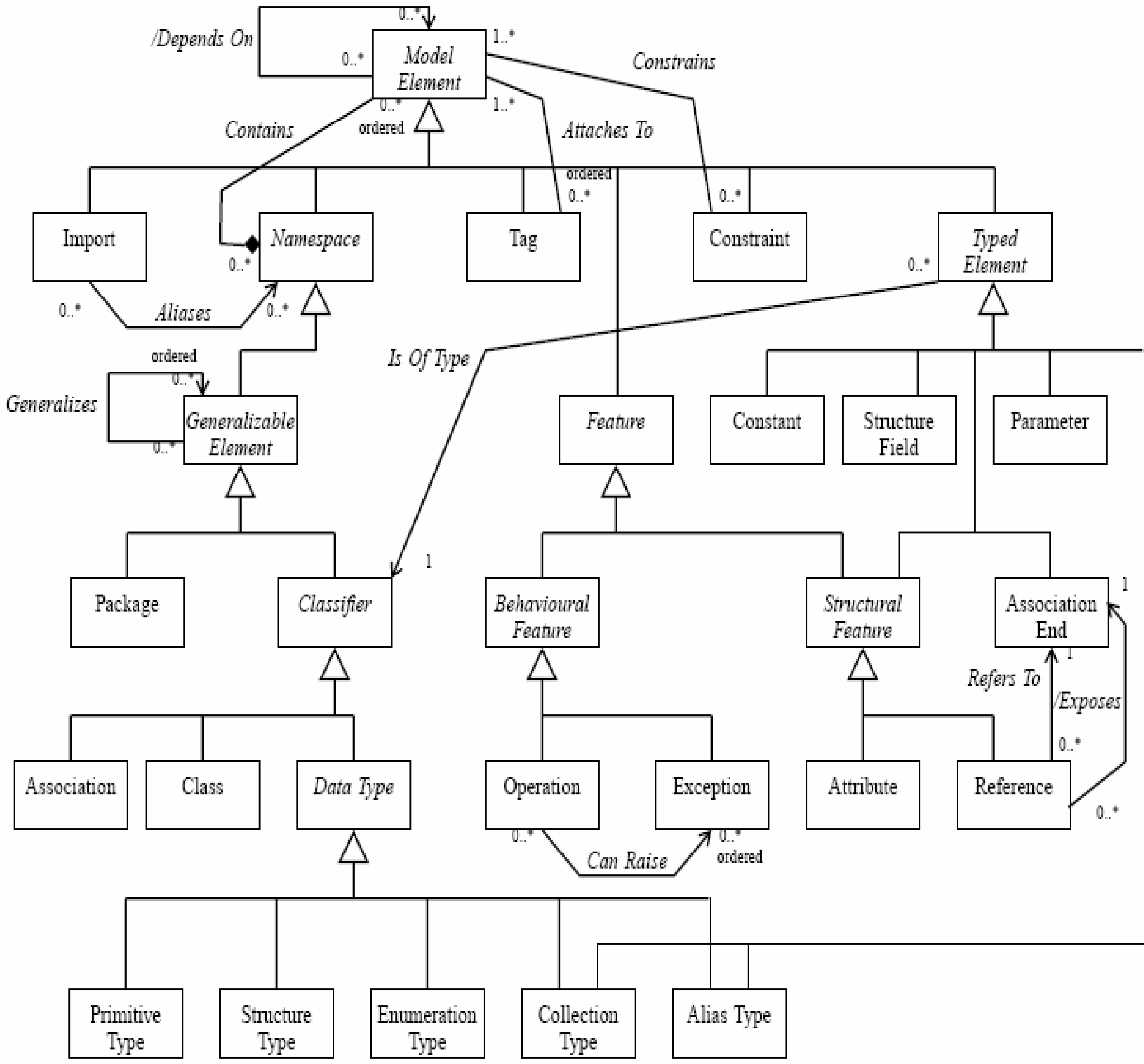
Contents

MOF : Generalities

- An <u>object</u> is « instance of » (the UML definition of) a <u>class</u>
- A <u>database</u> record is « instance of » a <u>table schema</u>
- <u>class</u> or <u>table schema</u> are concepts described in a ***metamodel***
  - A class is composed of attributes and operations and have a name
  - An attribute belongs to a class, have a name, a type and a multiplicity
  - …
- A metamodel is represented as a (meta-)class model
  - Metaclasses class, attribute, operation…
- But where is described a metaclass ?
  - In a meta metamodel !
  - Should we continue (meta meta metamodel…) ?
  - Are there different kinds of meta metamodels ?

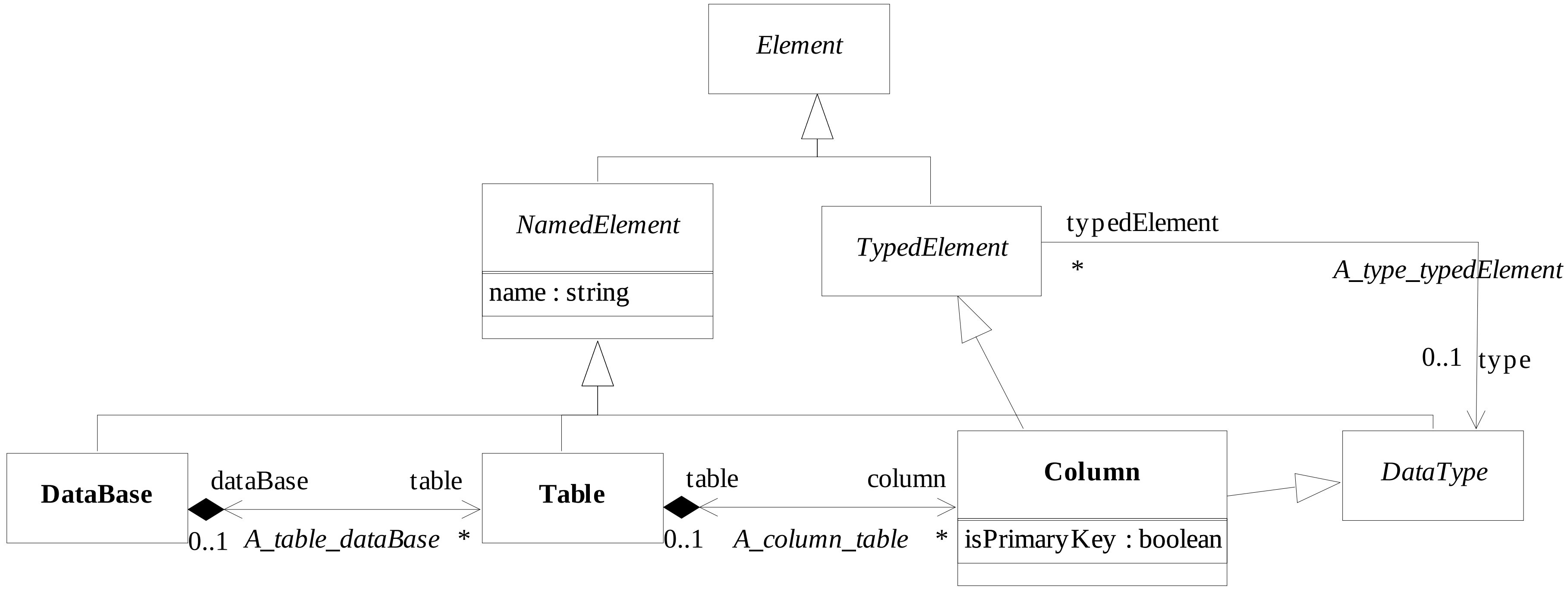
MOF : Generalities

- MOF (Meta Object Facility) is THE meta metamodel
  - The goal is to describe real world
  - Real world abstracted in a model
  - Different kinds of model
  - One way to describe a metamodel is enough !
- It is self-described !
  - The meta meta metamodel is the MOF meta metamodel
- Standard from the OMG



MOF : Layers

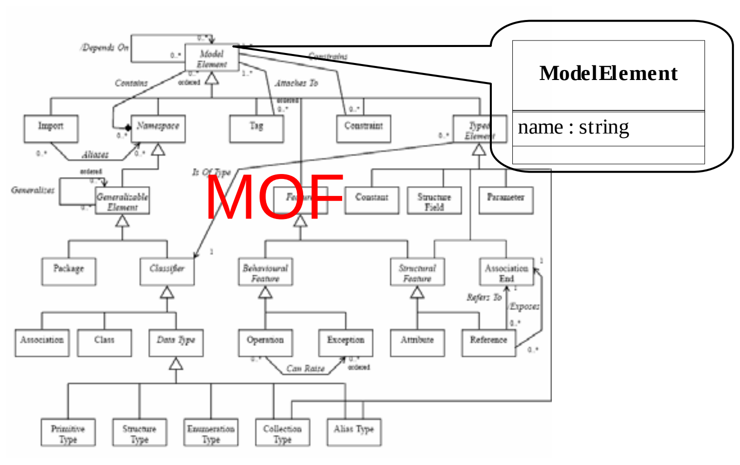
Let’s take a simple example of metamodel



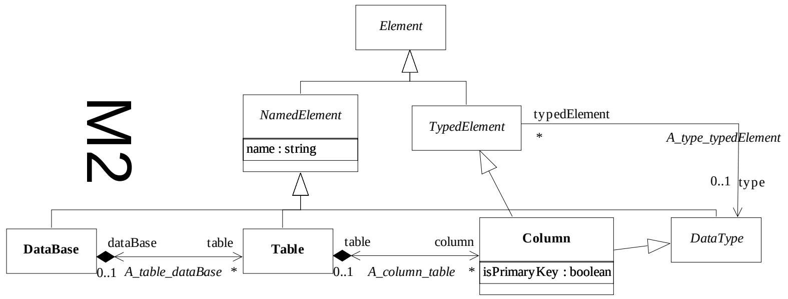
MOF : Layers

The 4 layers architecture (here missing M0 = the real world ; imagine your favorite team as a winner !)

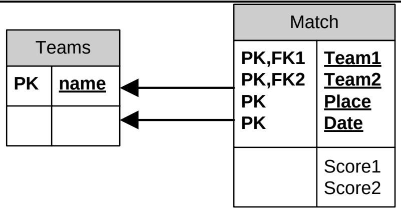
M3



M2



M1



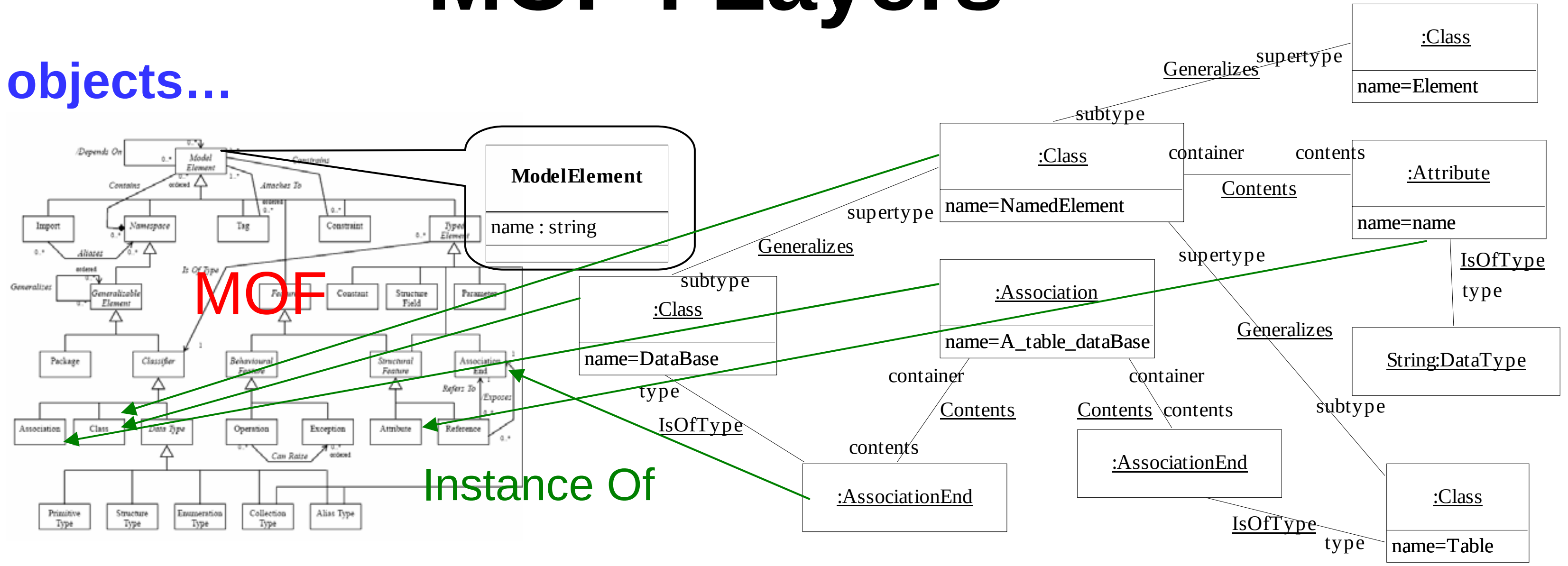
Teams	
PK	<u>name</u>

Match	
PK,FK1 PK,FK2 PK PK	**<u>Team1</u>** **<u>Team2</u>** **<u>Place</u>** **<u>Date</u>**
	Score1 Score2

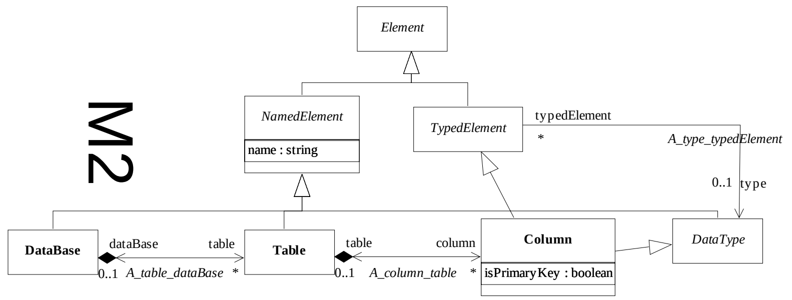
MOF : Layers

M3 objects…

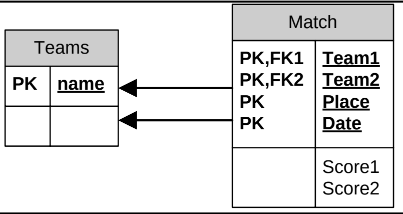
M3



M2

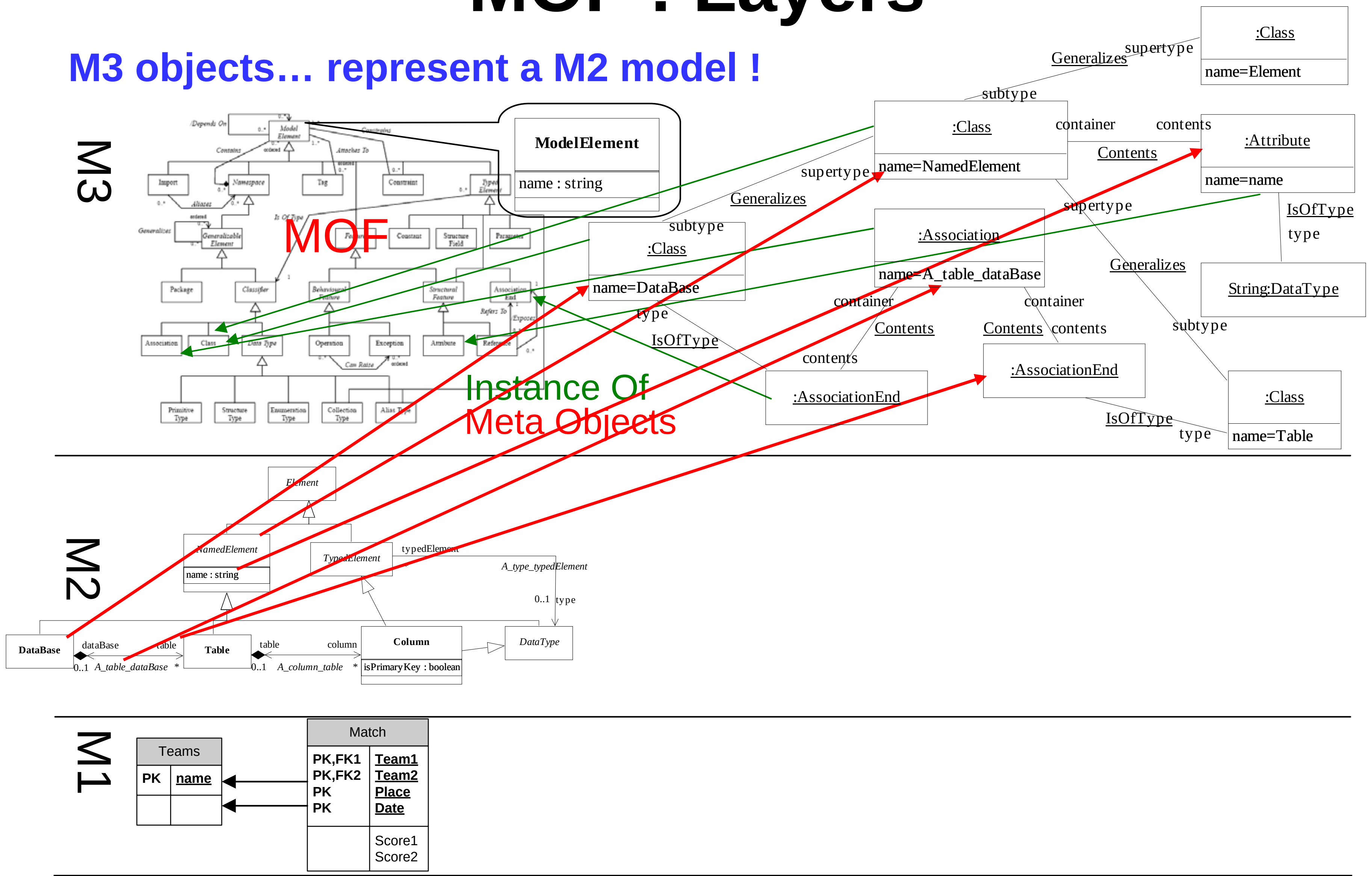


M1



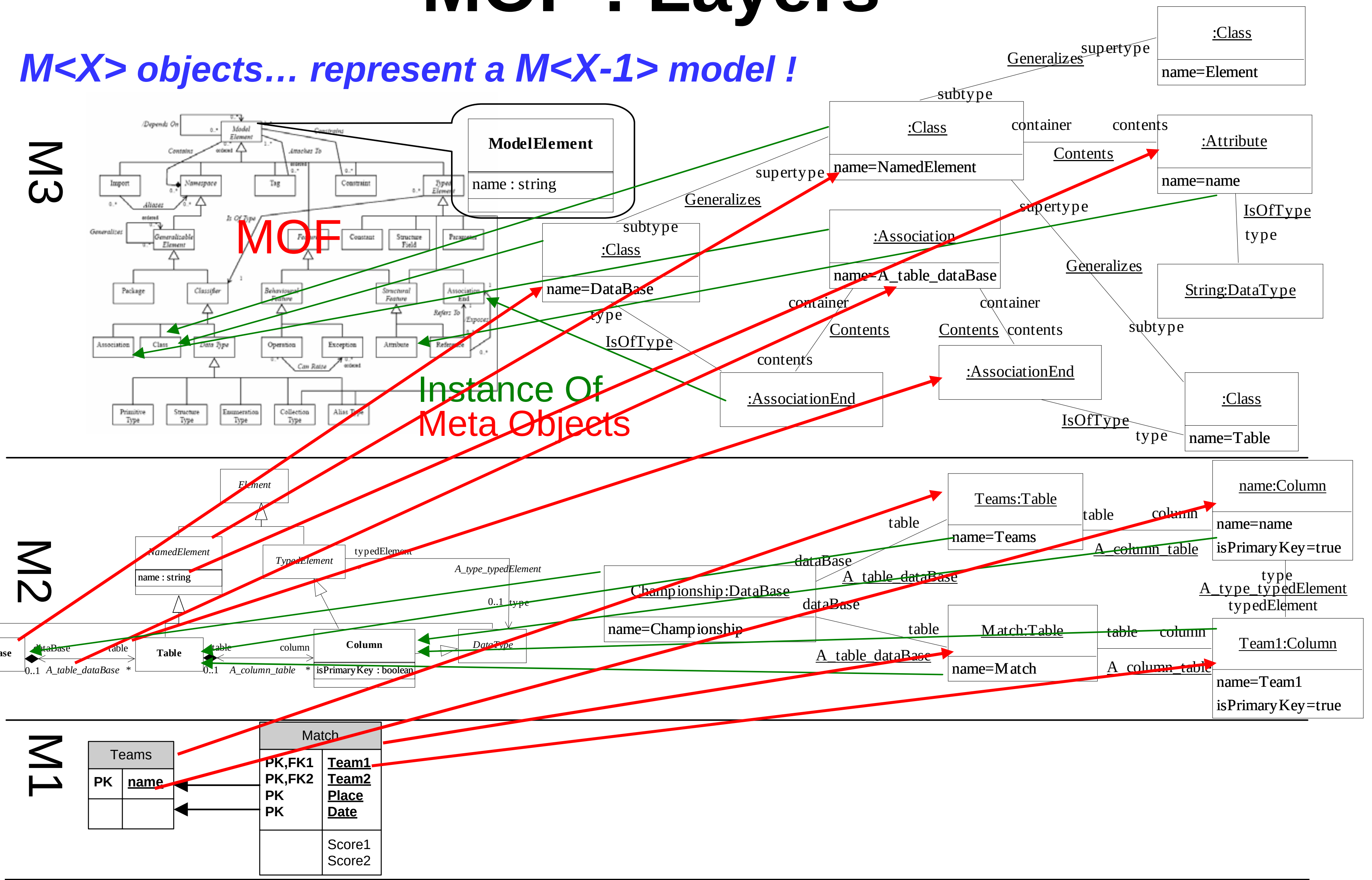
MOF : Layers

M3 objects… represent a M2 model !



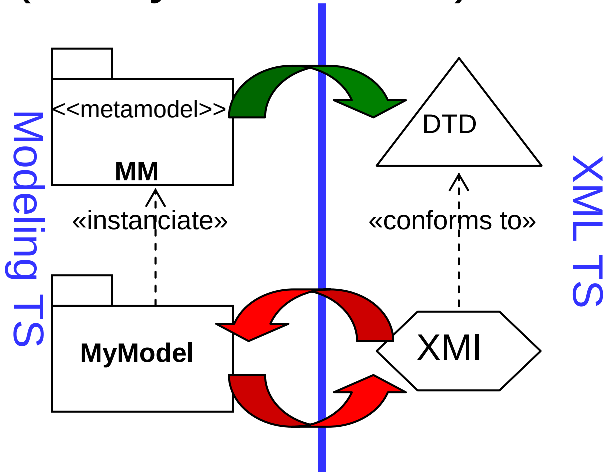
MOF : Layers

M<X> objects… represent a M<X-1> model !



MOF : Model interchange (XMI)

- Model interchange is standardized
- Should take into account models of any kinds (of any metamodel)
- XMI is XML => it needs a schema
- Schema is given by the M(X+1) level !
  - Tools generate a DTD from a metamodel
  - Tools load / store models from / to XMI
  - An “XMI model” is valid for a given metamodel
    - XMI is a language template



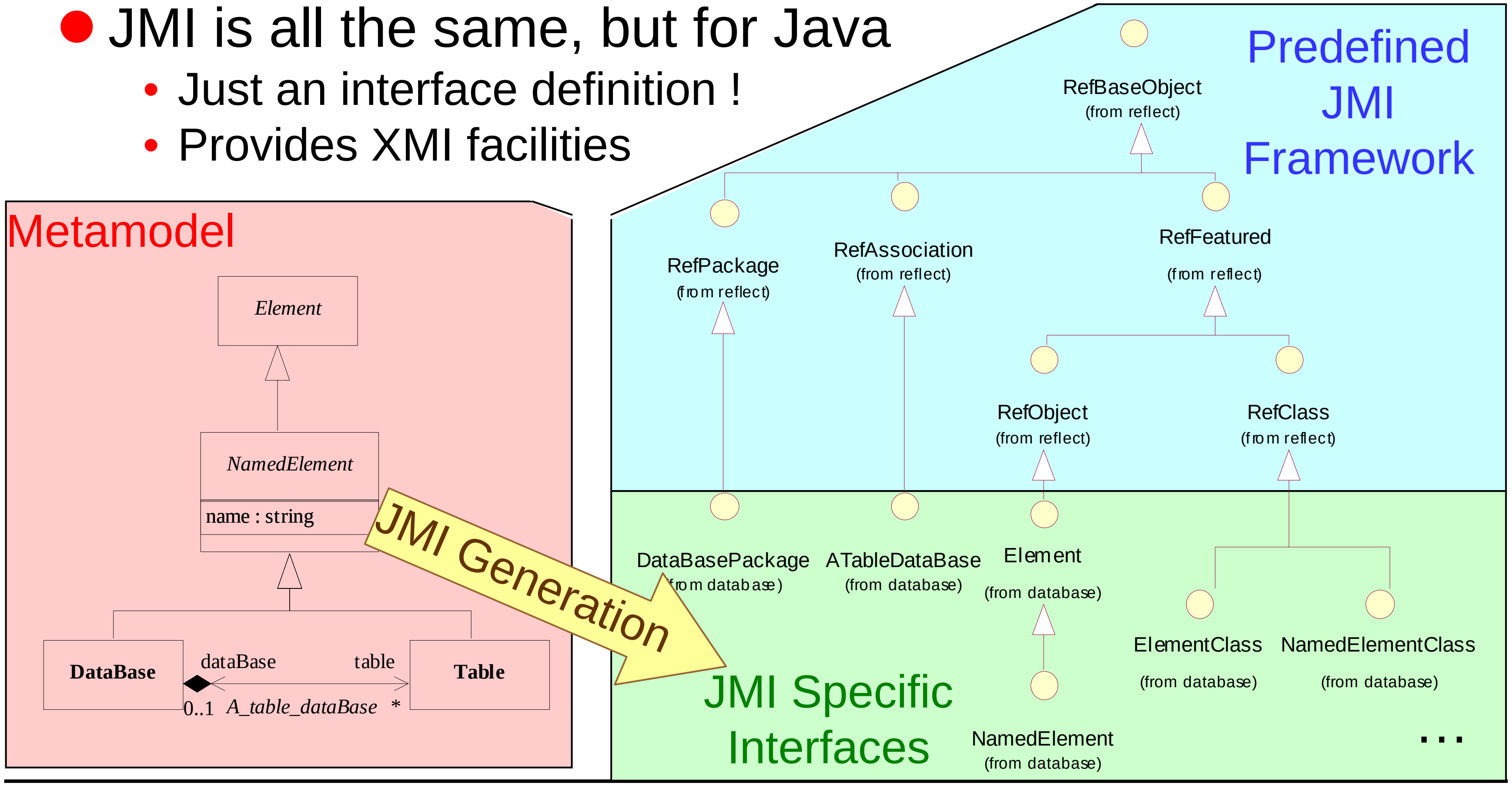
- Remark
  - 3 versions of XMI (1.0, 1.1, 1.2)
  - Many versions of metamodel (UML : 0.9, 1.0, 1.3, 1.4, 2.0…)
  - Tools interpret the XMI standard as they wish !
  - XMI possibilities for a same model (of a given metamodel) : Cartesian product of
    - XMI version
    - Metamodel version
    - Tool

Contents

- About MOF
- About JMI
  - A MOF mapping for Java
  - Reflective facilities
  - Generated interfaces
- The MDR tool
- Demo
- Outlook

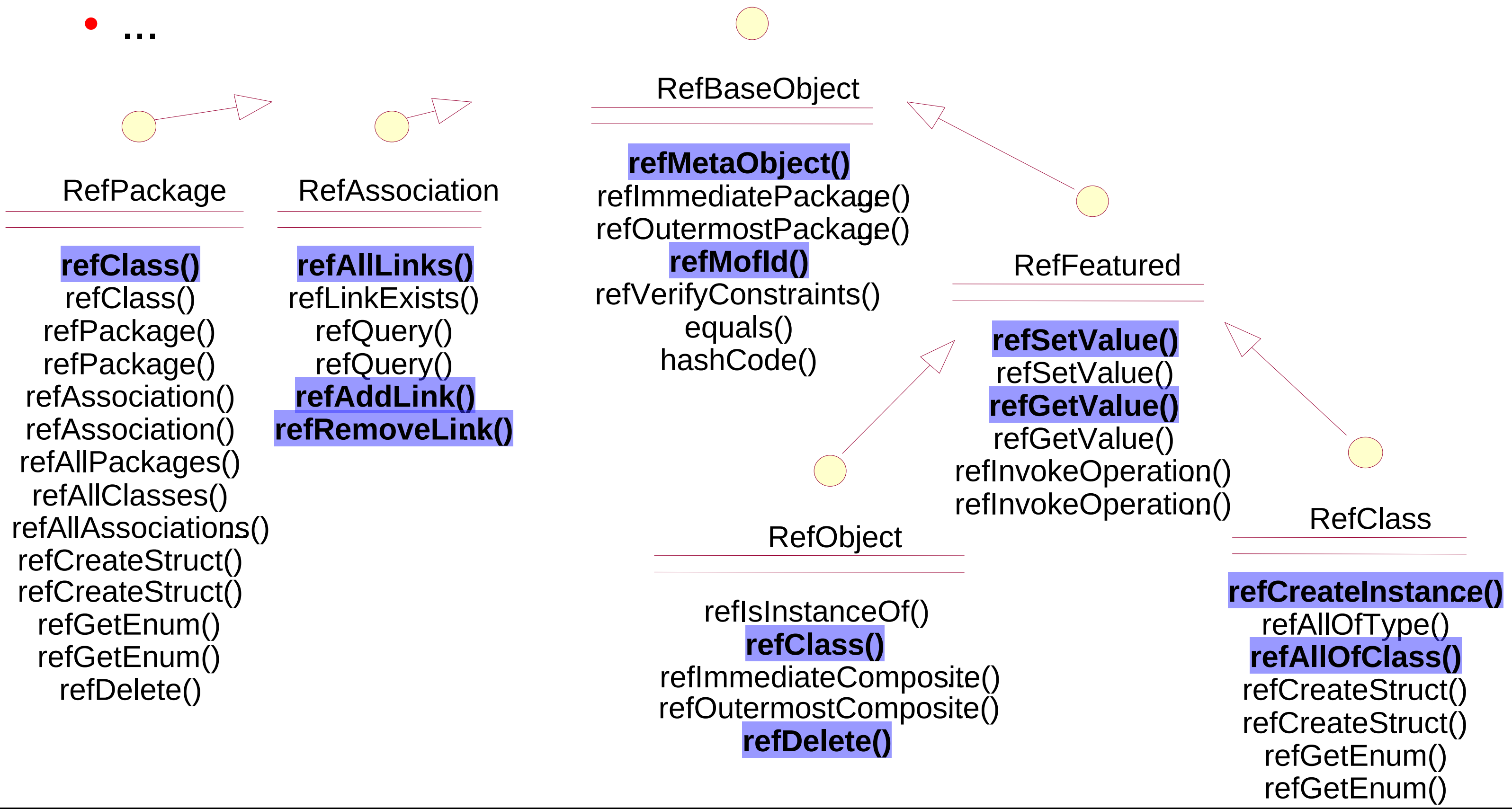
JMI : A MOF mapping for Java

- « MOF to IDL mapping » chapter
  - Concept part of the MOF standard
  - Made to access and to manipulate the model through CORBA
- JMI is all the same, but for Java
  - Just an interface definition !
  - Provides XMI facilities

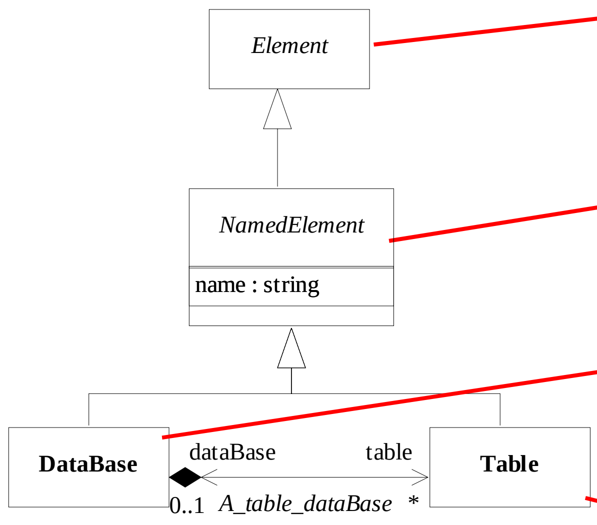

Predefined JMI Framework
RefBaseObject (from reflect)
RefPackage (from reflect)
RefAssociation (from reflect)
RefFeatured (from reflect)
RefObject (from reflect)
RefClass (from reflect)
Metamodel
Element
NamedElement
name : string
DataBase
Table
dataBase
table
0..1
A_table_dataBase
*
JMI Generation
JMI Specific Interfaces
DataBasePackage (from database)
ATableDataBase (from database)
Element (from database)
NamedElement (from database)
ElementClass (from database)
NamedElementClass (from database)
...

JMI : Reflective facilities

- As defined in the MOF, it is possible to
  - Access the metatype of an object
  - Asks a metatype for each one of its instances
  - Access a feature of an object (with name of meta element)
  - ...



JMI : Generated interfaces



```
public interface Element
extends javax.jmi.reflect.RefObject {
}
```

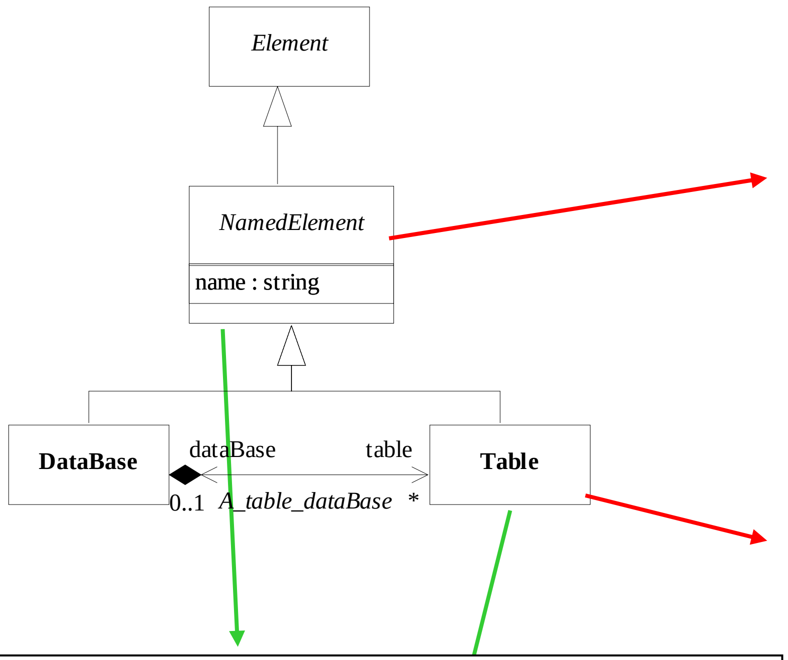
```
public interface NamedElement
extends database.Element {
  public String getName();
  public void setName(String newValue);
}
```

```
public interface DataBase
extends database.NamedElement {
  public Collection getTable();
}
```

```
public interface Table
extends database.NamedElement {
  public DataBase getDataBase();
  public void setDataBase(String newValue);
}
```

- It is possible to access “meta objects” here
  - “Element” interface extends RefObject, so have a RefClass
  - Can access the meta properties of an element
    - Name (any direct “DataBase” instance returns the “DataBase” string)
    - Contents (applied on any NamedElement returns the “name” meta-attribute)
    - …

JMI : Generated interfaces

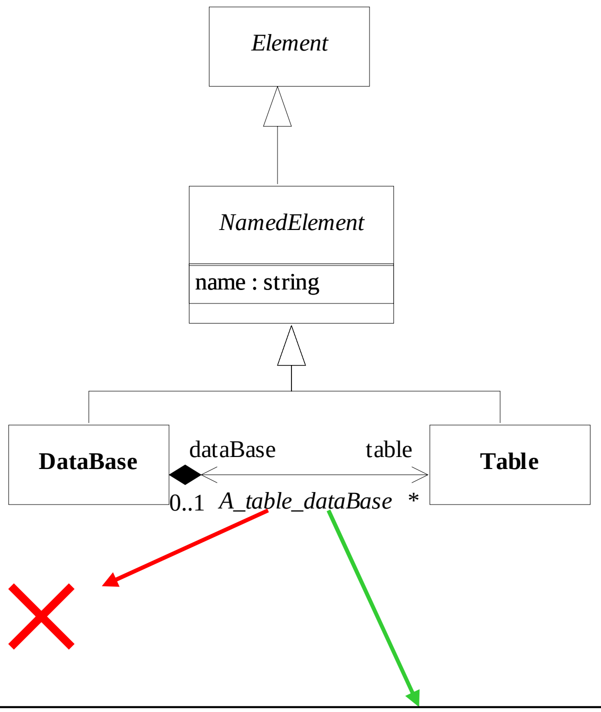


```
public interface NamedElementClass
extends javax.jmi.reflect.RefClass {
}
```

```
public interface TableClass
extends javax.jmi.reflect.RefClass {
  public Table createTable();
  public Table createTable (String name);
}
```

- In order to create an object, you must contact its metaclass
- A metaclass is a *singleton*

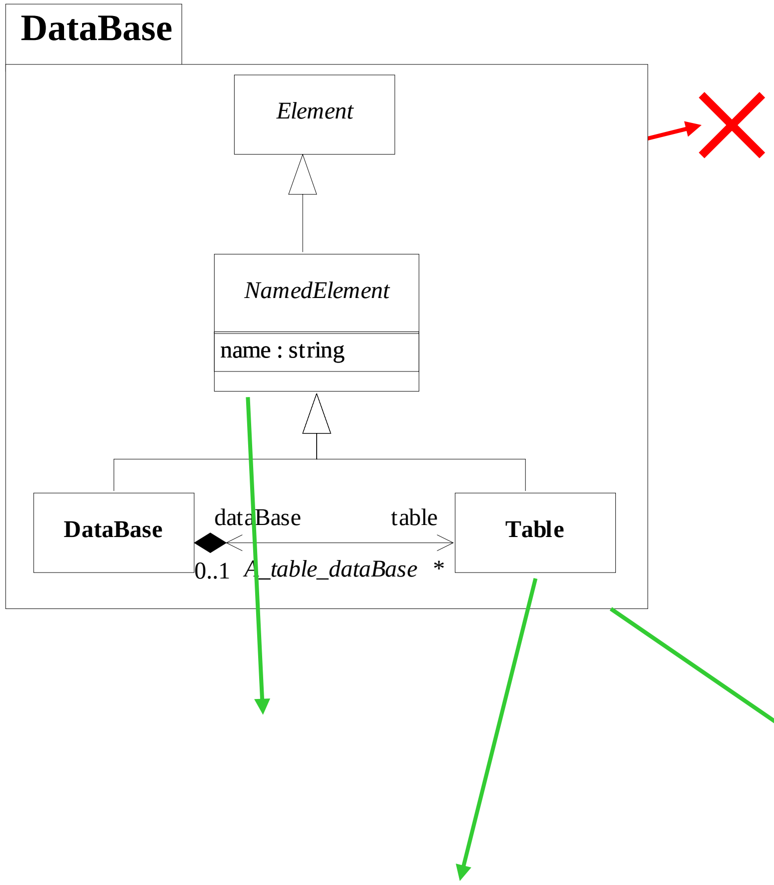
JMI : Generated interfaces



```
public interface ATableDataBase
extends javax.jmi.reflect.RefAssociation {
  public boolean exists(Table table, DataBase dataBase);
  public Collection getTable(DataBase dataBase);
  public DataBase getDataBase(Table table);
  public boolean add(Table table, DataBase dataBase);
  public boolean remove(Table table, DataBase dataBase);
}
```

- A meta association is a *singleton*

JMI : Generated interfaces



- In order to create an object, you must contact its metaclass
- A meta element is a *singleton* and provide access to its nested meta elements
- The root meta package is the entry point to access these singletons
- Need to be provided a mechanism to retrieve the root package singleton

```
public interface DataBasePackage
extends javax.jmi.reflect.RefPackage {
  public ElementClass getElement();
  public NamedElementClass getNamedElement();
  public DataBaseClass getDataBase();
  public TableClass getTable();
  public ATableDataBase getATableDataBase();
}
```

Contents

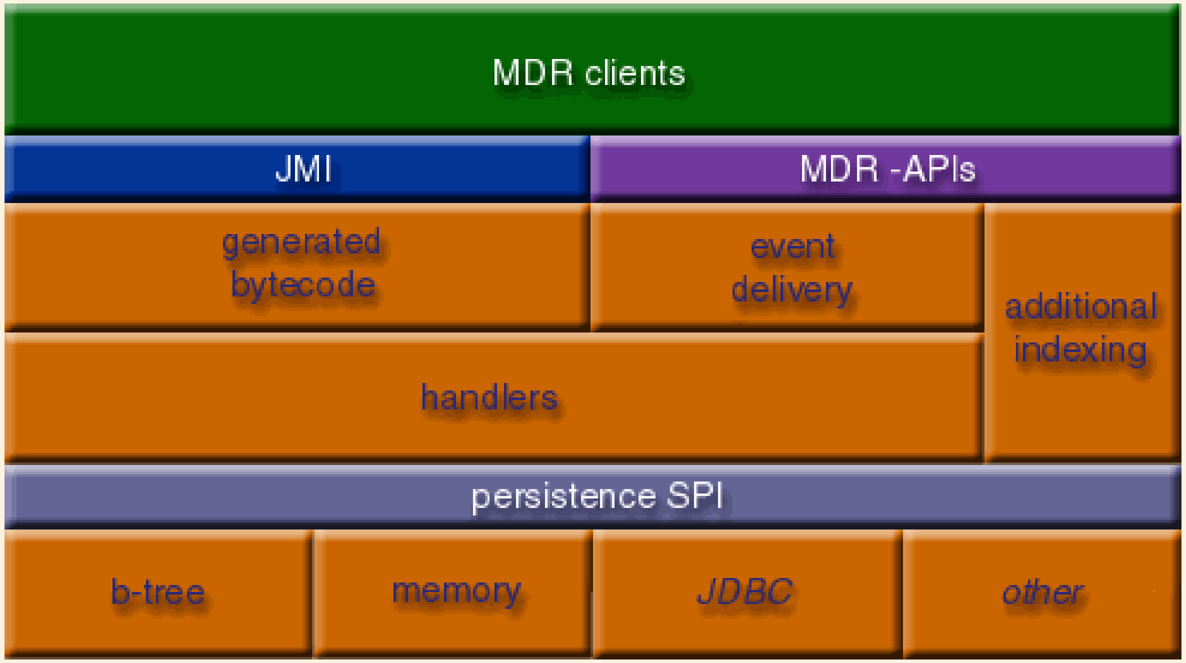
MDR : An implementation of JMI

An open-source tool from Sun

- MDR provides access to the meta package we needed above
- It is able to manage any model of any (MOF) metamodel
- It can generate the JMI interfaces
- It provides an implementation for these interfaces
- It provides XMI support
  - Reader
  - Writer
  - DTD generation

MDR is a model repository

MDR : Architecture



Contents

- About JMI
- The MDR tool
- Demo
- Outlook

Contents

- About JMI
- The MDR tool
- Demo
- Outlook
  - Just describe a metamodel to build a repository
  - Manipulate your models as you manipulate objects
  - No support for profile or constraint…
  - Tricky support for operations and constraints
  - Many tools use MDR
    - The new version of Dresden OCL Toolkit
    - Poseidon
    - The model transformation languages MTL and ATL
    - …

Thank you !

- Any question ?