Kermeta Day'09 – Rennes 2009/04/02

Implementing a singleton with aspects in Kermeta

INRIA

IRISA

Agenda

- Intent
- Applicability
- Structure and sample
- Limitations

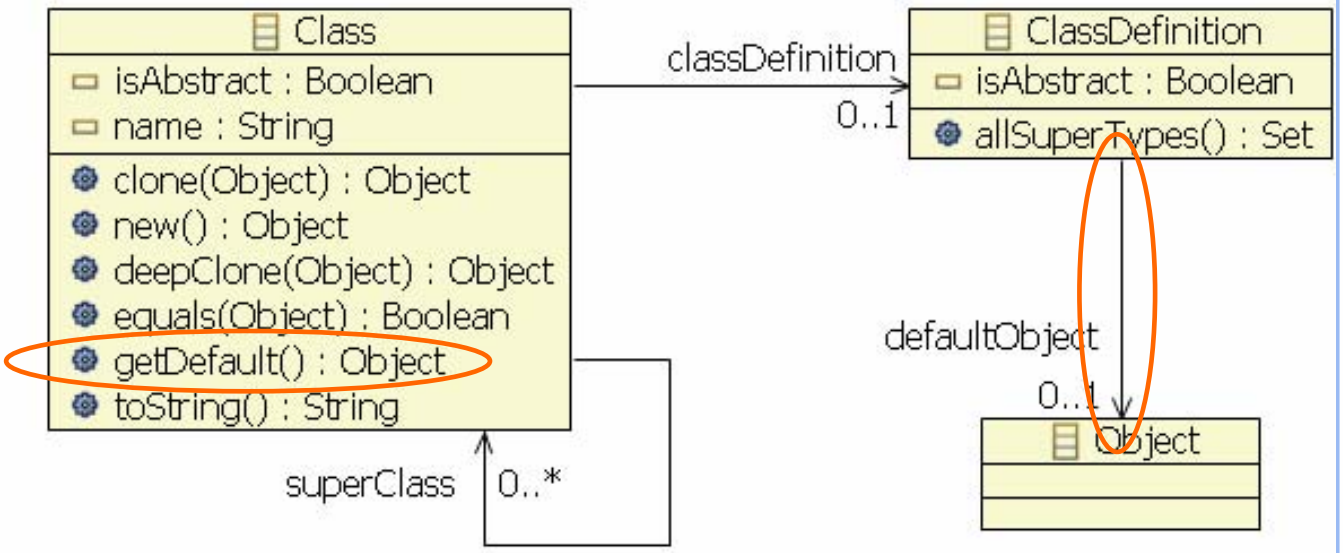
Intent

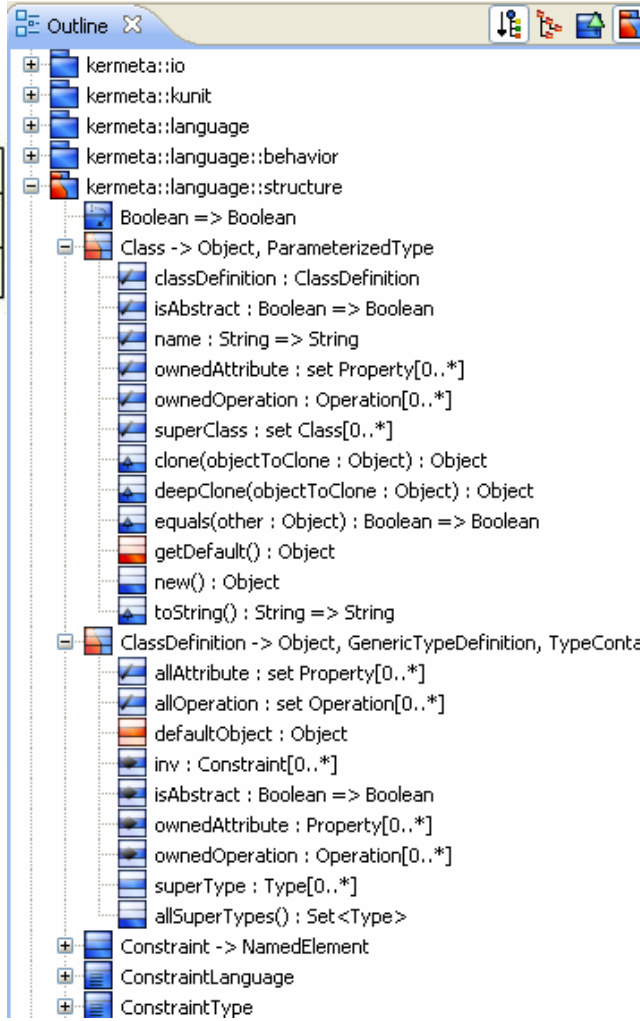
- No static operation in kermeta
  - how can I share an instance across the application ? Especially if it is a crosscuting data like a logger.
- Show how kermeta aspect can be used to implement the classical singleton design pattern :
  - Ensure a class only has one instance, and provide a global point of access to it.

Applicability

- Use the Singleton pattern when
  - there must be exactly one instance of a class, and it must be accessible to clients from a well-known access point.
  - when the sole instance should be extensible by subclassing, and clients should be able to use an extended instance without modifying their code.

Structure





Ensures creation and access point

Store the instance

Implementation


```
package kermeta::language::structure;

aspect class ClassDefinition{
   /** Internal object returned by getDefault on a Class whose
     * typeDefinition is self. Should not be accessed by other means */
   reference defaultObject : Object
}

aspect class Class {
   /* Get or create a default object instance of this Class */
   operation getDefault() : Object is do
       if self.typeDefinition.isInstanceOf(ClassDefinition) then
           var cd : ClassDefinition
           cd ?= self.typeDefinition
           if cd.defaultObject.isVoid then
               // create the default object
               cd.defaultObject := self.new
           end
           result := cd.defaultObject
       else
           var exception : NotImplementedException
           exception := NotImplementedException.new
           exception.message := "getDefault works only with ClassDefinition"
           raise exception
       end
   end
}
```

Sample code


```
require
    "platform:/plugin/org.kermeta.language.mdk/src/kmt/language/extension/SingletonSupport.kmt"

…
// configure the logger for your application
var logger : MySingletonSimpleLogger
logger ?=  MySingletonSimpleLogger.getDefault()
logger.level := 2
// you may also replace the default instance by your own subclass
…

MySingletonSimpleLogger.getDefault().asType(MySingletonSimpleLogger).printMessage("Hello world")
…
MySingletonSimpleLogger.getDefault().asType(MySingletonSimpleLogger).printError("an error msg")
…



class MySingletonSimpleLogger {
    /* 0 =  no messages; 1 = errors only; 2 = all messages */
    attribute level : Integer
    operation ensureDefaultLevel() is do
         if level.isVoid then level := 1 end
    end

    operation printMessage(msg : String) is do
         ensureDefaultLevel
         if level > 1 then
              stdio.writeln(msg)
         end
    end
    operation printError(msg : String) is do
         ensureDefaultLevel
         if level > 0 then
              stdio.writeln(msg)
         end
    end
}
```

Use it anywhere in your code provided you have the require

Limitations

- Not a strict singleton : Kermeta cannot forbid the access to the instruction "new"
  - So we cannot ensure that the program hasn't created another instance somewhere else.
  - Programmers should know that they must use the getDefault operation

Conlusion

- Kermeta aspects applied to extend Kermeta itself

- There is definitively no need for statics in Kermeta