TÉCNICO LISBOA

Multiple Dispatch in Java

Professor:

António Menezes Leitão

André Rodrigues nº69998
Gonçalo Castilho nº75305
Sílvia Timóteo nº75770

TÉCNICO LISBOA

Introdução ao problema

- CLOS
  - Multiple dispatch
- Java
  - Single dispatch + overloading

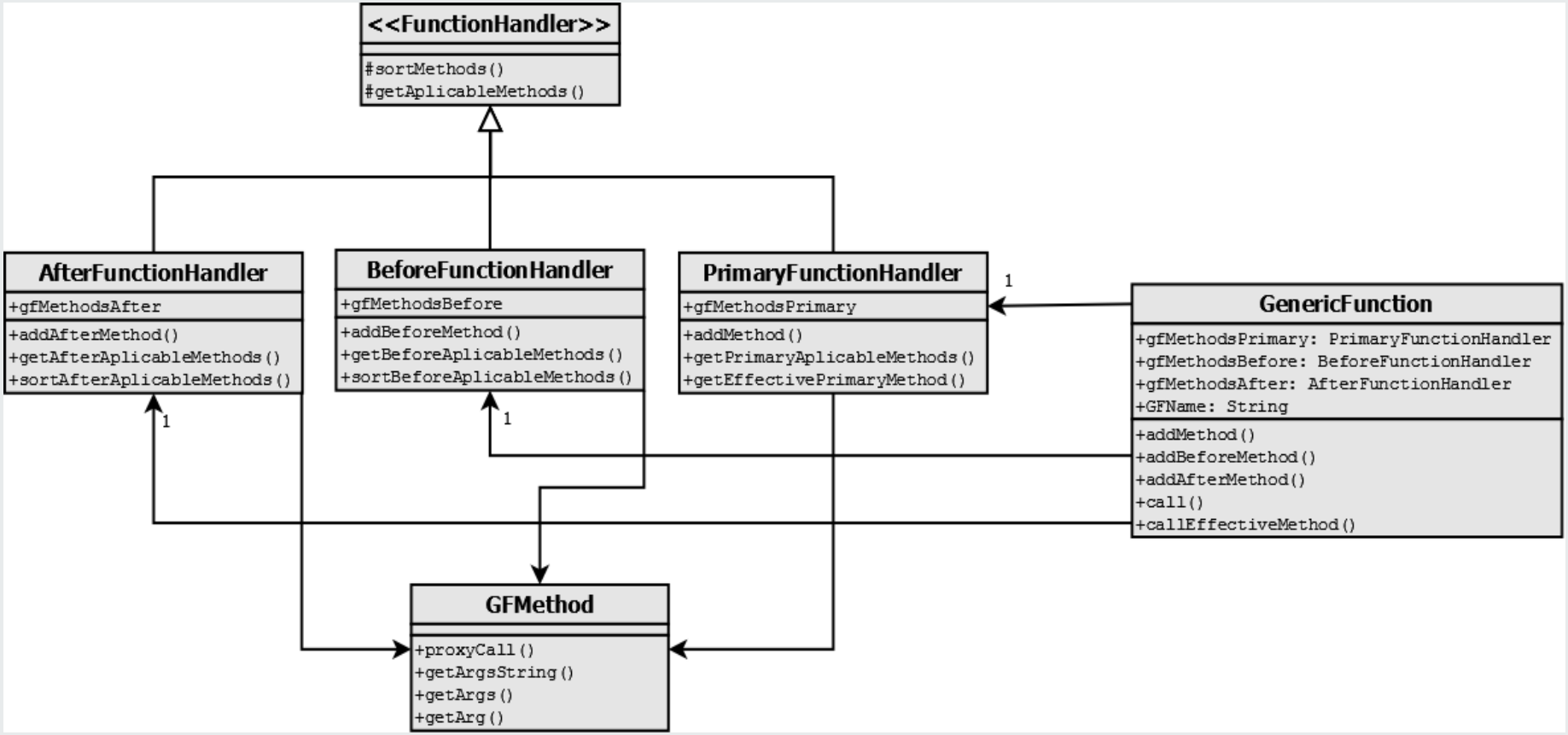
TÉCNICO LISBOA

Objectivo

- Permitir Multiple Dispatch em java.

Solution

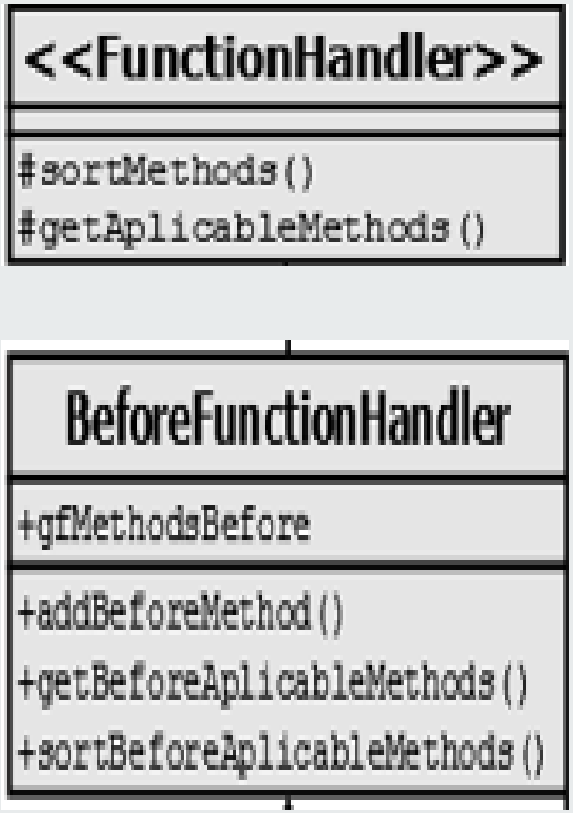
UML


<<FunctionHandler>>
#sortMethods()
#getAplicableMethods()
AfterFunctionHandler
+gfMethodsAfter
+addAfterMethod()
+getAfterAplicableMethods()
+sortAfterAplicableMethods()
BeforeFunctionHandler
+gfMethodsBefore
+addBeforeMethod()
+getBeforeAplicableMethods()
+sortBeforeAplicableMethods()
PrimaryFunctionHandler
+gfMethodsPrimary
+addMethod()
+getPrimaryAplicableMethods()
+getEffectivePrimaryMethod()
1
GenericFunction
+gfMethodsPrimary: PrimaryFunctionHandler
+gfMethodsBefore: BeforeFunctionHandler
+gfMethodsAfter: AfterFunctionHandler
+GFName: String
+addMethod()
+addBeforeMethod()
+addAfterMethod()
+call()
+callEffectiveMethod()
1
1
GFMethod
+proxyCall()
+getArgsString()
+getArgs()
+getArg()

Function Handler

- Guarda os métodos de um tipo (***Around***, ***Primary***, ***Before*** e ***After***) para uma função genérica.

- Selecciona os métodos aplicáveis e ordena-os de acordo com a chamada efectuada.

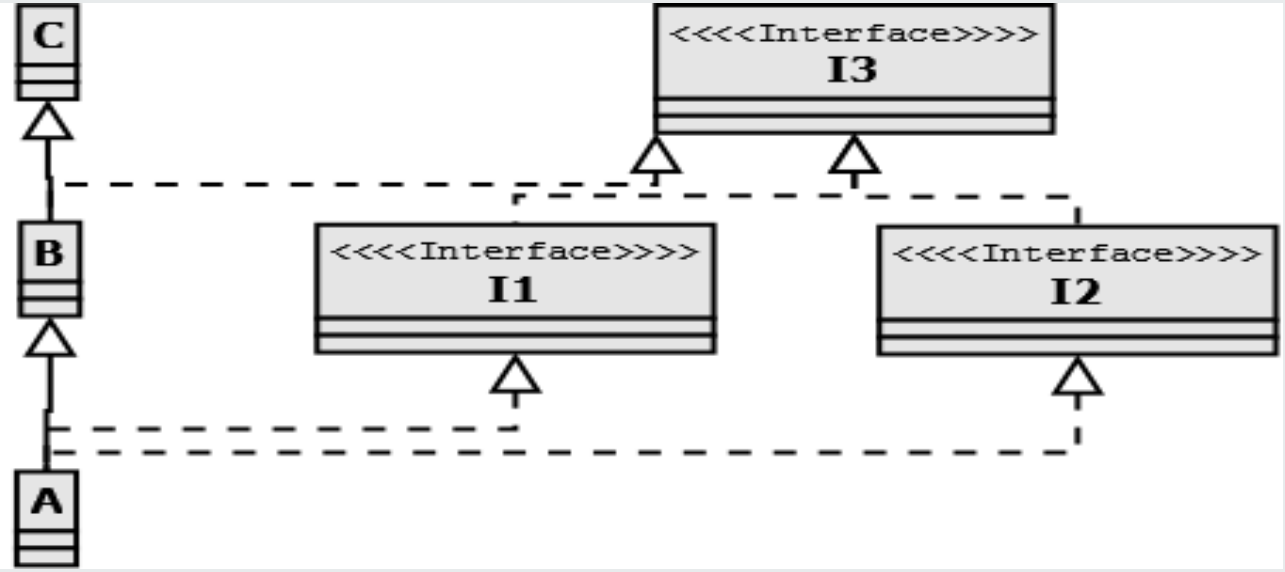


TÉCNICO LISBOA

Precedência de classes

- Foi utilizada uma procura ***BFS*** para gerar a lista de precendias de classes

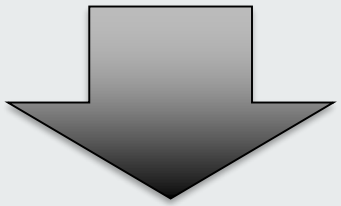
Exemplo de ordenação



A -> B -> I1 -> I2 -> C -> I3

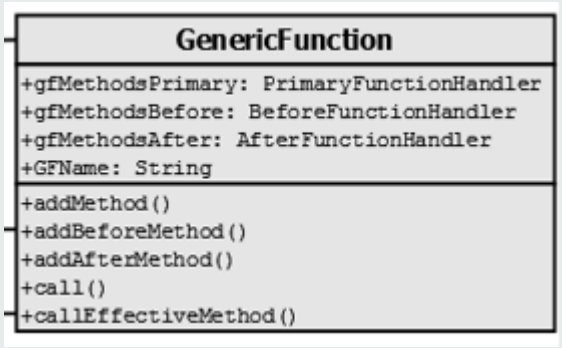
Precedência de classes

Critério de desempate entre classes e interfaces ao mesmo nivel de hierarquia:



As classes têm precedência sobre as interfaces e a ordem das interfaces é aquela pela qual sao declaradas.

Generic Function



- GFName – string representativo do nome da função.
- 3 *handlers*, um correspondente aos métodos ***primários***, outro aos ***after*** e outro aos ***before***.

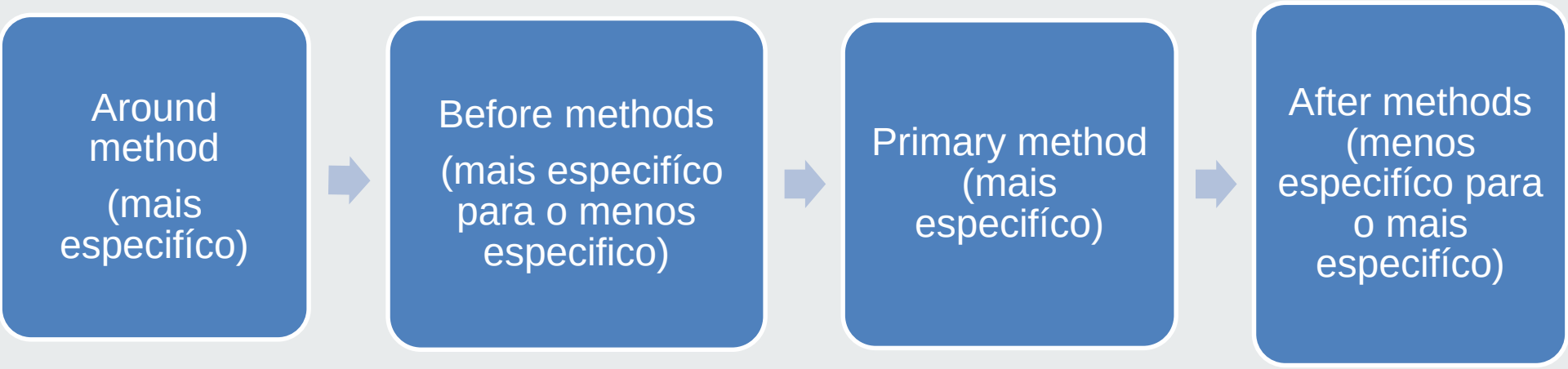
TÉCNICO LISBOA

Generic Function

- Para ser executada a função generic é calculado ***effective method*** para os argumentos da chamada.

Generic Function

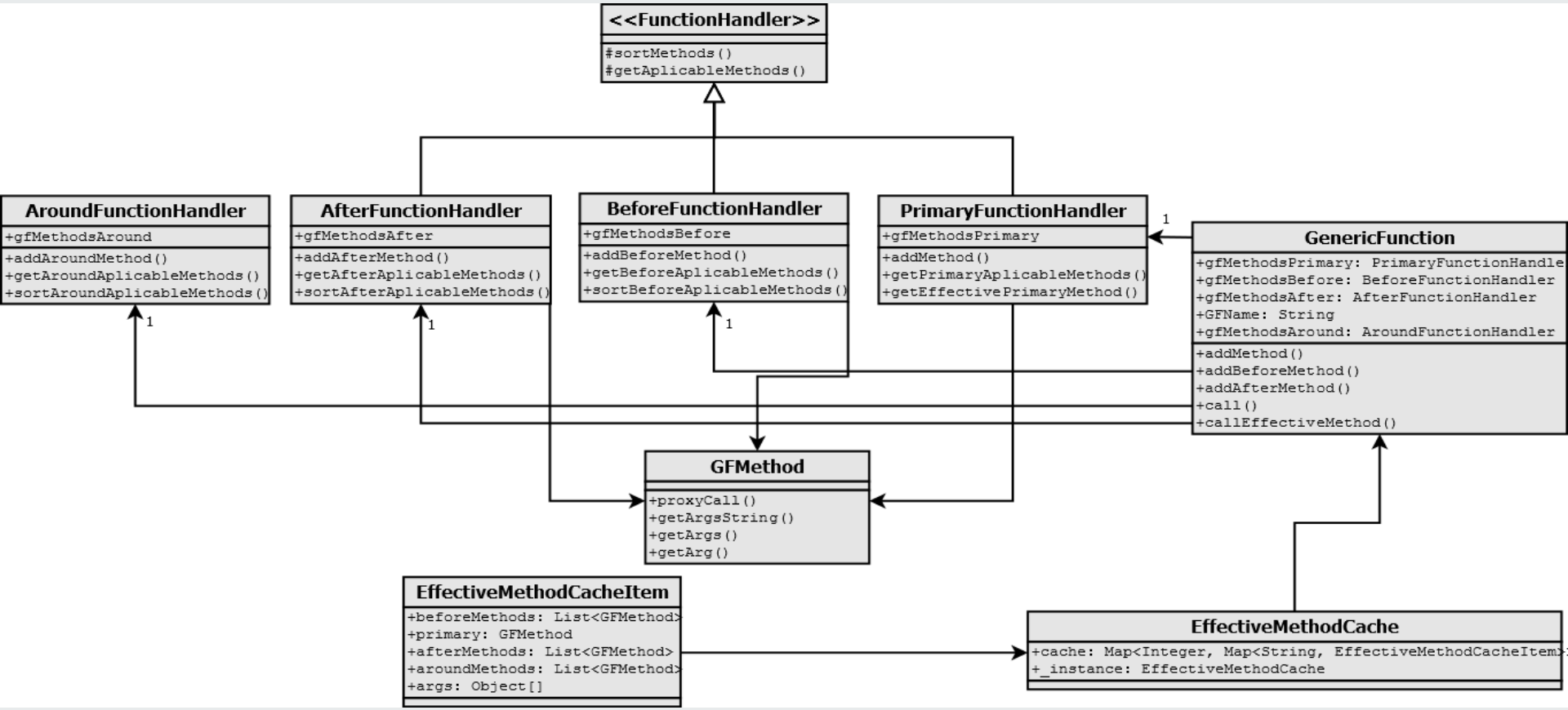
- ***Effective method***



TÉCNICO LISBOA

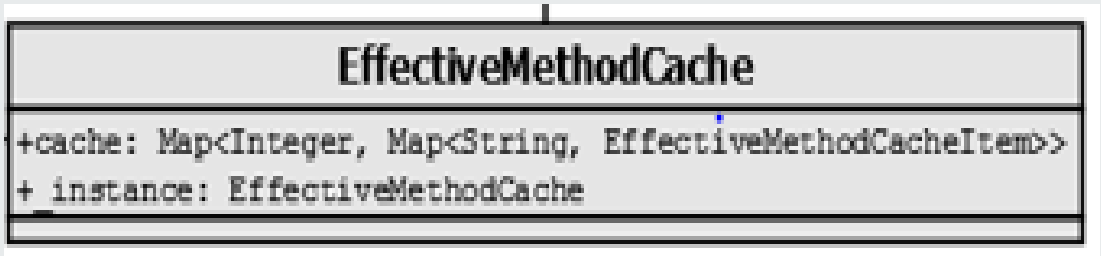
Extensões

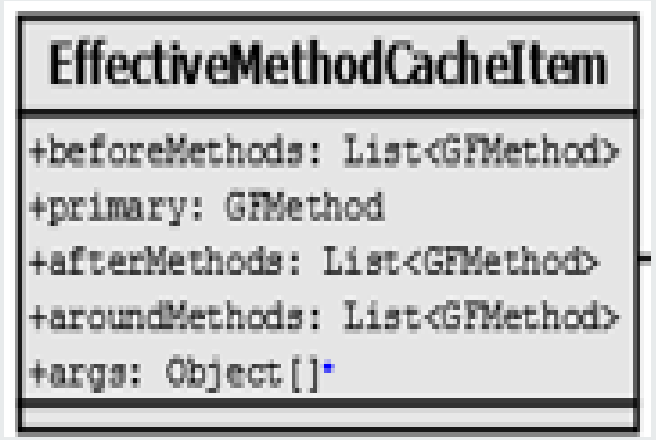
UML


<<FunctionHandler>>
#sortMethods()
#getAplicableMethods()
AroundFunctionHandler
+gfMethodsAround
+addAroundMethod()
+getAroundAplicableMethods()
+sortAroundAplicableMethods()
AfterFunctionHandler
+gfMethodsAfter
+addAfterMethod()
+getAfterAplicableMethods()
+sortAfterAplicableMethods()
BeforeFunctionHandler
+gfMethodsBefore
+addBeforeMethod()
+getBeforeAplicableMethods()
+sortBeforeAplicableMethods()
PrimaryFunctionHandler
+gfMethodsPrimary
+addMethod()
+getPrimaryAplicableMethods()
+getEffectivePrimaryMethod()
GenericFunction
+gfMethodsPrimary: PrimaryFunctionHandle
+gfMethodsBefore: BeforeFunctionHandler
+gfMethodsAfter: AfterFunctionHandler
+GFName: String
+gfMethodsAround: AroundFunctionHandler
+addMethod()
+addBeforeMethod()
+addAfterMethod()
+call()
+callEffectiveMethod()
GFMethod
+proxyCall()
+getArgsString()
+getArgs()
+getArg()
EffectiveMethodCacheItem
+beforeMethods: List<GFMethod>
+primary: GFMethod
+afterMethods: List<GFMethod>
+aroundMethods: List<GFMethod>
+args: Object[]
EffectiveMethodCache
+cache: Map<Integer, Map<String, EffectiveMethodCacheItem>
+_instance: EffectiveMethodCache
1
1
1
1
1

Cache

- Guarda o cálculo do effectiveMethod para uma dada chamada.





TÉCNICO LISBOA

Métodos Around e Call-next-method

- Foi implementada a possibilidade de existirem métodos ***around***, bem como ***call-next-method*** nos métodos around com comportamento semelhante ao do CLOS.

TÉCNICO LISBOA

Questões?