Java Inside - Résumé du lab3

Thiebaud Jeremy

INFO2 2020-2021

MethodHandle

classe MethodHandle : pointeur de fonction typé à l'exécution

Appeler un MethodHandle (du + au – performant) :

- invocation exacte avec .invokeExact() : arguments d'appel et paramètre de la méthode doivent être identique (aucune conversion ne sera faite).
  Lance une WrongMethodTypeException si mauvais paramètres.

- invocation "comme en Java" avec .invoke : conversions habituelles autorisées (type primitive, sous-typage, boxing, varargs).

- invocation "comme java.lanf.reflect.Method.invoke" avec .invokeWithArguments() : prend en paramètre soit un tableau soit une liste, il y a deux surcharges.

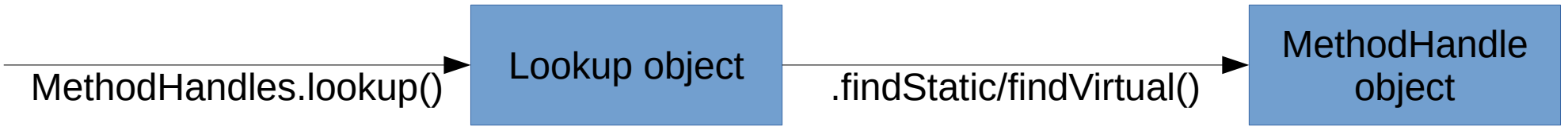
Lookup

Créer un objet Lookup : MethodHandles.lookup()

Methodes de Lookup :

- FindStatic : cherche une méthode statique
- FindVirtual : cherche une méthode d’instance

Paramètres : (Class, "methodName", methodType(returnClass, argumentClass, ...))

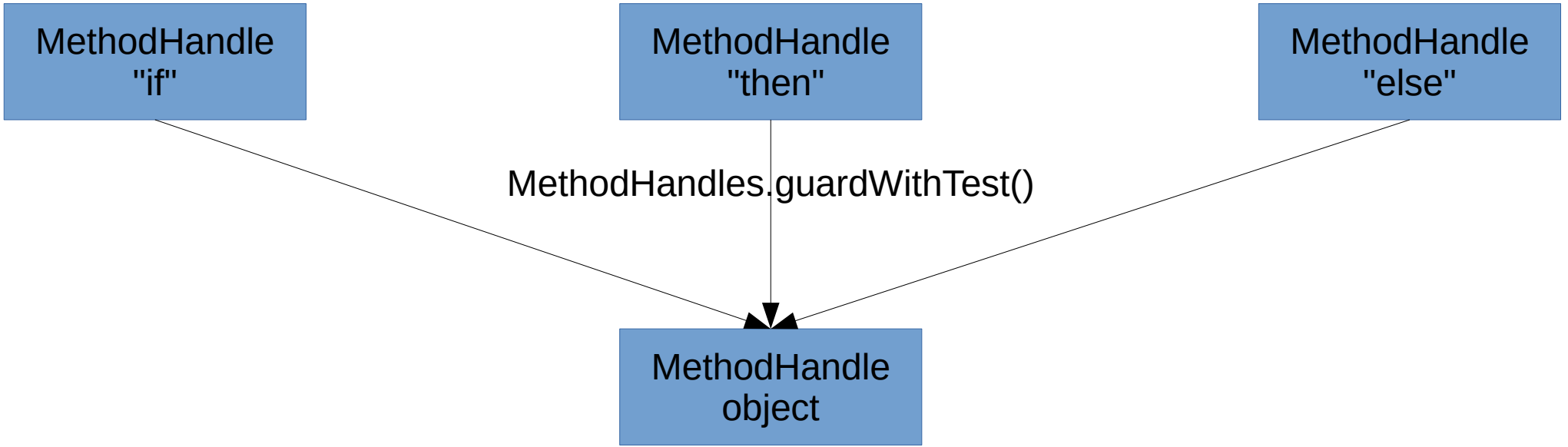


Méthodes importantes

- MethodHandles.insertArguments() : insérer une valeur à la place d'un paramètre à un MethodHandle existant en renvoyant un nouveau MethodHandle avec un paramètre de moins.
- MethodHandles.dropArguments() : supprimer la valeur d'un argument en renvoyant un nouveau MethodHandle avec un paramètre supplémentaire.
- MethodHandle.asType() : prend en paramètre un type de fonction MethodType et renvoie un MethodHandle qui va convertir ses arguments pour pouvoir appeler le MethodHandle sur lequel asType a été appelé
- MethodHandles.constant() : crée un MethodHandle qui renvoie toujours une constante
- MethodHandle.bindTo() : méthode d’instance similaire à MethodHandles.insertArgument. L’argument est toujours ajouté en position 0

MethodHandles.guardWithTest

prend trois MethodHandles en paramètre et renvoie un MethodHandle qui si il est appelé, appel le premier MethodHandle qui doit renvoyer un boolean et en fonction de la valeur du booléen appel soit le deuxième MethodHandle si le booléen est vrai ou le troisième MethodHandle si le booléen est faux agissant comme une sorte de if.



Cas d’utilisation

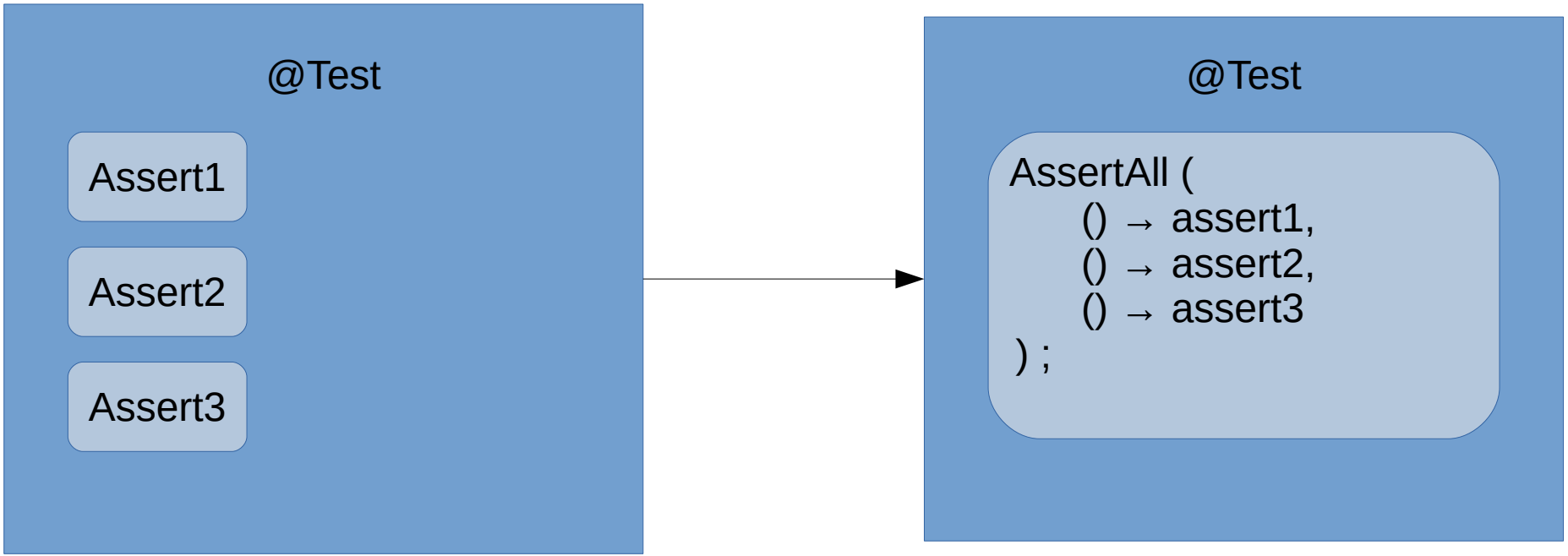
A quoi sert MethodHandles.dropArguments ?

```
private MethodHandle match(String str) throws NoSuchMethodException, IllegalAccessException {
    var equals = LOOKUP.findVirtual(
            String.class,
            "equals",
            methodType(boolean.class, Object.class));
    return MethodHandles.guardWithTest(
            MethodHandles.insertArguments(equals, 1, str),
            MethodHandles.dropArguments(MethodHandles.constant(int.class, 1), 0, String.class),
            MethodHandles.dropArguments(MethodHandles.constant(int.class, -1), 0, String.class)
    );
}
```

MethodHandles.guardWithTest demande aux MethodHandleS d’avoir la même signature (Excepté pour le test qui renvoi un booléen et qui peut prendre moins de paramètres)

Utiliser MethodHandles.dropArguments pour avoir une signature acceptable.

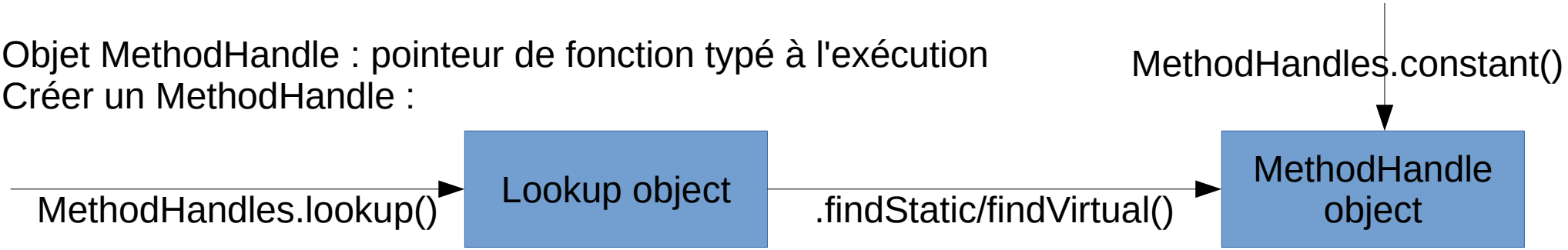
JUNIT5 - assertAll



Conclusion

Objet MethodHandle : pointeur de fonction typé à l'exécution

Créer un MethodHandle :



Utiliser un MethodHandle :