JAVA INSIDE
TP 03 - Invocation API java.lang.invoke

- Lookup
- MethodHandle

Lien GitHub : https://github.com/MelissaDaCosta/java-inside

Mélissa DA COSTA

API Réflective

API de réflexion :

java.lang.reflect.Method.invoke

API d’invocation :

java.lang.invoke.MethodHandle.invokeExact

Utilisation de l'API d'invocation

La classe Lookup permet de connaître les visibilités des champs de la classe courantes :

```
var lookup = MethodHandles.lookup(); // donne le lookup courant
```

La méthode privateLookupIn permet d'accéder aux méthodes privés (=setAccessible) :

```
var lookupPrivate = MethodHandles.privateLookupIn(Example.class, lookup);
```

Appeler une méthode

La méthode findStatic permet de trouver une méthode static :

```
var methodHandle = lookupPrivate.findStatic(Example.class, "aStaticHello",
methodType(String.class, int.class));
```

Elle retourne un pointeur de fonction sur la méthode trouvée.

Pour appeler cette méthode, il faut utiliser invokeExact :

```
assertEquals("question 4", (String) methodHandle.invokeExact(4));
```

Il faut donner le type de retour de la méthode appelée car invokeExact vérifie qu'elle a la même signature.

findVirtual

Pour appeler une méthode autre que static il faut utiliser findVirtual :

```
var methodHandle = lookupPrivate.findVirtual(Example.class,
"anInstanceHello", methodType(String.class, int.class));
```

FindVirtual permet de trouver une méthode virtuelle, c'est-à-dire une méthode d'instance ou d'interface en utilisant le polymorphisme.

insertArguments

La méthode insertArguments permet d'insérer des arguments à l'appel d'une méthode :

```
var methodHandleInsert =
MethodHandles.insertArguments(methodHandle, 1, 6);
```

Dans l'exemple ci-dessus, le int 6 est inséré en argument de la méthodHandle à la position 1.

Sachant que la position 0 des arguments est occupée par le nom de la classe.

dropArguments

La méthode dropArguments permet de supprimer des arguments à l'appel d'une méthode :

```
var methodeHandleDrop =
MethodHandles.dropArguments(methodHandle, 0, Example.class);
```

Dans l'exemple ci-dessus, l'argument à la position 0 est supprimé (la classe).

Pour appeler la méthode il faut donc lui rajouter un argument lors de l'appel qui correspond à celui retiré :

```
assertEquals("question 7", (String)
methodeHandleDrop.invokeExact(new Example(), new Example(), 7));
```

asType et constant

AsType permet de gérer l'un-boxing :

```
var methodHandleAsType = methodHandle.asType(methodType(String.class,
Example.class, Integer.class));
```

En appelant la méthode avec un integer en argument, la différence entre methodHandleAsType et le methodHandle original va être faite pour convertir l'Integer reçu en int.

Pour créer une constante :

```
var methodHandleConstant = MethodHandles.constant(int.class, 9);
```

Créée une méthode qui ne prend rien en argument et renvoie une constante int qui vaut 9.