Introspection

Programmation avancée – Java

F5 – ISIMA 2020/2021

Olivier Goutet
o.goutet@openium.fr

3 novembre 2020

Introduction – Introspection

- Introquoi ?

Introduction – Introspection

- Introquoi ?
- Capacité d'un programme à examiner, et éventuellement à modifier, ses structures internes de haut niveau (par exemple ses objets) lors de son exécution (Wikipedia)

Introduction – Introspection

- Introquoi ?
- Capacité d'un programme à examiner, et éventuellement à modifier, ses structures internes de haut niveau (par exemple ses objets) lors de son exécution (Wikipedia)
- Mécanisme pour découvrir de manière dynamique les informations d'une classe Java.

Plan

Introspection

Introspection

- Permet d'inspecter et d'intéragir avec les objets
- Tout objet hérite de Object
- `getClass()`
  - Connaitre à l'éxécution tout ce qu'est, propose, hérite et implemente une classe

Class

```
String chaine = "Coucou";
Class c = chaine.getClass();

system.out.println(c.getName());

try{
   String s2 = (String) c.newInstance();
} catch (InstantiationException e){
   ...
}catch (IllegalAccessException e){
   ...
}
try{
   Class maClasse = Class.forName("MaClasse");
} catch (ClassNotFoundException e) {
   ...
}
```

Reflexion

- Permet d'examiner un objet
  - Attributs
    - `getFields()` / `getDeclaredFields()`
  - Méthodes
    - `getMethods()` / `getDeclaredMethods()`
  - Constructeurs
    - `getConstructors()` / `getDeclaredConstructors()`

Exemple

Récupérer les noms de toutes les méthodes publiques

```
Method [] methodes = MaClasse.class.getMethods();
for (int i=0;i<methodes.length;i++){
  System.out.println( methodes[i] );
}

Method[] methodes = monObjetInconnu.getClass().getMethods();
```

Modification d'attributs

```
class Loto{
  private double montantDuGain;
}
 - - - - - - - - - - - - - - - - - - - - -
Loto monLoto = new Loto();
try{
  Field champ = monLoto.getClass().getField("montantDuGain");
  // lecture
  double montantDuGain = champ.getDouble(monLoto);
  // ecriture
  champ.setDouble(monLoto,montantDuGain+43);
} catch(NoSuchFieldException e){
  // pas de champ "montantDuGain"
} catch (IllegalAccessException e2){
  // pas le droit de modifier le champ
}
```

Accès aux méthodes

- `getMethods()`
- `invoke(Object obj,Object[] args)`
  - obj : objet sur lequel invoquer
    - Null si méthode statique
  - args : arguments de la méthode

Exemple

```
class Invoke{
  public static void main(String [] args){
    try{
      Class c = Class.forName(args[0]);
      Method m = c.getMethod(args[1], new Class[] {} );
      Object ret = m.invoke(null,null);
      System.out.println("Methode␣Statique␣invoquee:␣" +
      args[1] + "␣de␣la␣classe␣" + args[0] + "␣retour="+ret);
    }catch (ClassNotFoundException e){}
    catch (IllegalAccessException e2){}
    catch (NoSuchMethodException e3){}
    catch (InvocationTargetException e4){}
  }
}
```

```
java Invoke java.lang.System currentTimeMillis
Methode statique invoquee : currentTimeMillis de
la classe java.lang.System retour= 876543729871;
```

Utilisations concrètes

- Trace lors d'un plantage
- Mapping Relationnel / Objet
- Analyse de qualité du code
- Test unitaire / mock (objet simulant)
- Injection de dépendances
- shells