



NAOUR
BOUTAINA

CHAPITRE 1 : INVERSION DE CONTROLE ET INJECTIONS DES DEPENDANCES

4-ème année IIR G21

Emsi les
orangers

Compte-rendu



Introduction



Énoncé



Conception



Code source



Captures écran de l'exécution



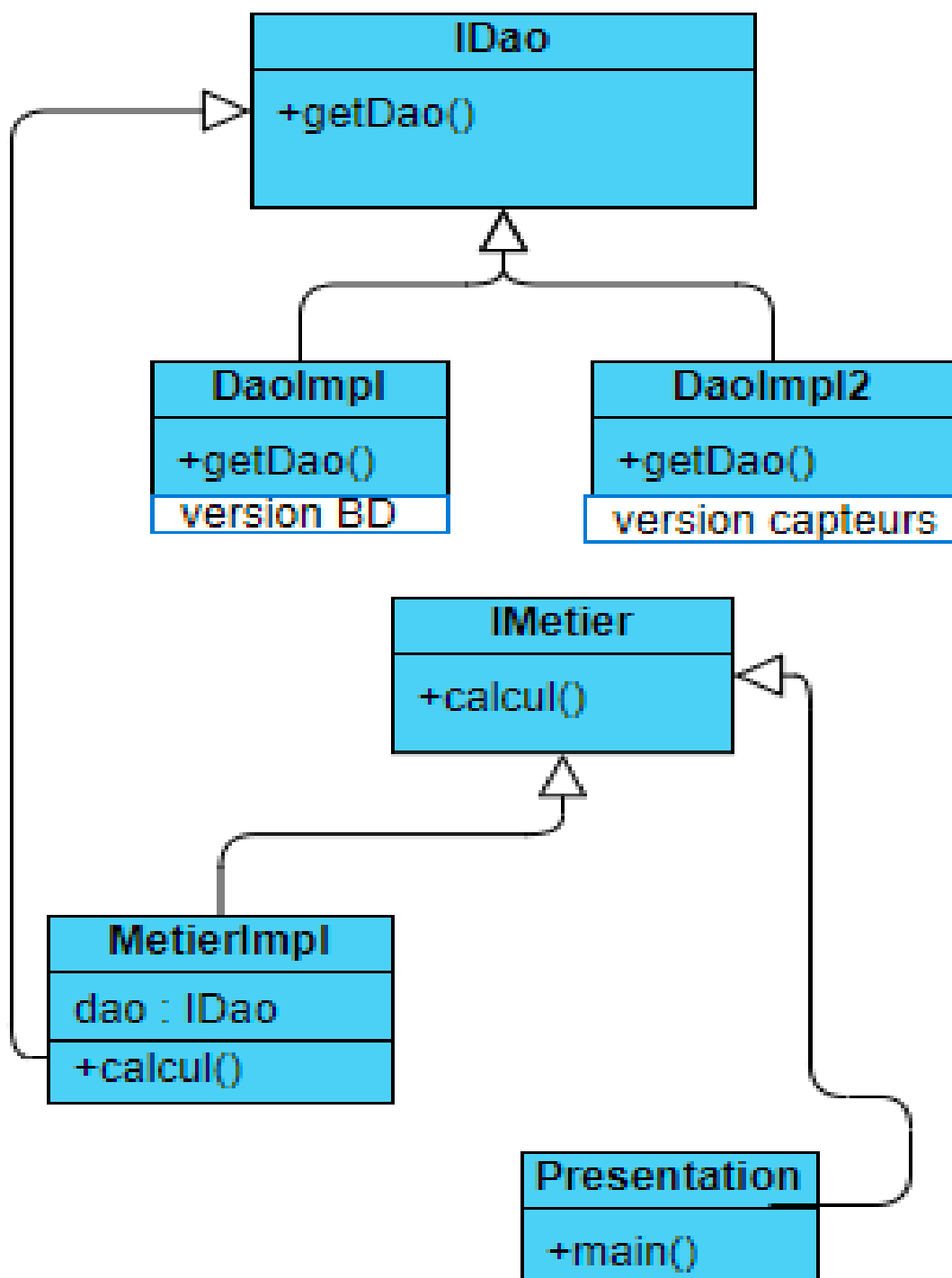
Conclusion

En programmation orientée objet, le principe ouvert/fermé affirme qu'une classe doit être à la fois ouverte (à l'extension) et fermée (à la modification). « Ouverte » signifie qu'elle a la capacité d'être étendue. « Fermée » signifie qu'elle ne peut être modifiée que par extension, sans modification de son code source. L'idée est qu'une fois qu'une classe a été approuvée via des revues de code, des tests unitaires et d'autres procédures de qualification, elle ne doit plus être modifiée mais seulement étendue.

Comment créer une application fermée à la modification et ouverte à l'extension ?

Injection des dépendances :

1. Instanciation statique
2. Instanciation dynamique
3. Par Spring Framework (XML et Annotations)



CODE SOURCE

Lien du code source en GitHub :

https://github.com/boutaina/jee_tp

PARTIE 1 : (JAVA)

https://github.com/boutaina/jee_tp/tree/main/emsi_l oc

PARTIE 2 : (SPRING)

https://github.com/boutaina/jee_tp/tree/main/emsi-ioc-2

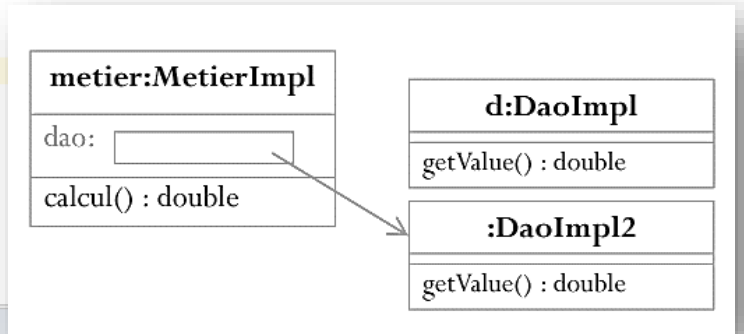
SCREEN SHOTS

A. Injection des dépendances par instanciation statique :

```
1 package pres;
2
3 import dao.DaoImpl;
4 import ext.DaoImpl2;
5 import metier.MetierImpl;
6
7 public class Presentation {
8     public static void main(String[] args) {
9         /*
10          * INJECTION DES DÉPENDANCE PAR
11          * INSTANCIATION STATIQUE => NEW
12          */
13         //DaoImpl dao=new DaoImpl();
14         DaoImpl2 dao=new DaoImpl2();
15         MetierImpl metier=new MetierImpl();
16         metier.setDao(dao);
17         System.out.println("Résultat="+ metier.calcul());
18     }
19 }
20
```

Run: Presentation x

"C:\Program Files\Java\jdk1.8.0_301\bin\java.exe" ...
Version Capteurs
Résultat=3240000.0
Process finished with exit code 0



B. Injection des dépendances par instanciation dynamique :

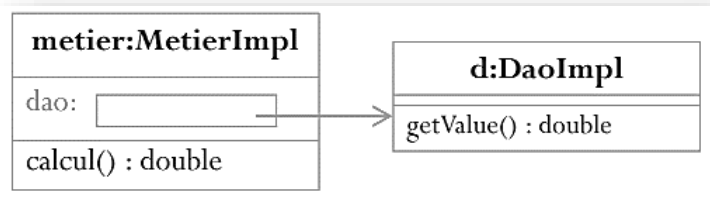
```
3 import dao.IDao;
4 import metier.IMetier;
5
6 import java.io.File;
7 import java.lang.reflect.Method;
8 import java.util.Scanner;
9
10 public class Pres2 {
11     public static void main(String[] args) throws Exception{
12         Scanner scanner=new Scanner(new File( pathname: "config.txt"));
13
14         String daoClassName=scanner.nextLine();
15         Class cDao=Class.forName(daoClassName);
16         IDao dao = (IDao) cDao.newInstance();
17
18         String metierClassName=scanner.nextLine();
19         //forName charger une classe en memoire en format Class
20         Class cMetier=Class.forName (metierClassName);
21         IMetier metier=(IMetier) cMetier.newInstance ();
22
23         Method method = cMetier.getMethod( name: "setDao", IDao.class);
24         //metier.setDao(dao);
25         method.invoke(metier,dao);
26         //invoke cad execute moi la méthode method
27         System.out.println("Résultat=>"+metier.calcul());
28     }
29 }
30
```

Run: Pres2 x

"C:\Program Files\Java\jdk1.8.0_301\bin\java.exe" ...
version base de donnée
Résultat=>-20563.078068381335
Process finished with exit code 0

Fichier texte de configuration : config.txt

```
1 dao.DaoImpl
2 metier.MetierImpl
```



Onget 5

SCREEN SHOTS

C. Injection des dépendances avec Spring : L'idée en résumé est de déporter la responsabilité de la liaison des composants du programme dans un Framework afin de pouvoir facilement changer ces composants ou leur comportement.

Injection des dépendances dans une application java standard :

The screenshot shows an IDE with two main windows. The left window displays an XML file named 'applicationContext.xml' with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/xsdt
                           http://www.springframework.org/schema/xsdt">
  <bean id="dao" class="ext.DaoImplVWS"></bean>
  <bean id="metier" class="metier.MetierImpl">
    <constructor-arg ref="dao"></constructor-arg>
  </bean>
</beans>
```

The right window displays a Java file named 'PresSpringXML.java' with the following content:

```
package pres;

import metier.IMetier;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class PresSpringXML {

    public static void main(String[] args) {
        ApplicationContext context=
            new ClassPathXmlApplicationContext(
                configLocation: "applicationContext.xml");
        IMetier metier=(IMetier) context.getBean( name: "metier");
        System.out.println("Resultat=>"+metier.calcul());
    }
}
```

Below these windows is a 'Run' console window showing the execution of 'PresSpringXML'. The output is:

```
"C:\Program Files\Java\jdk1.8.0_301\bin\java.exe" ...
Version web service
Resultat=>48600.0
Process finished with exit code 0
```

➤ Spring Version Annotations

The screenshot shows an IDE with three main windows. The left window displays a Java file named 'DaoImpl.java' with the following content:

```
package dao;

import org.springframework.stereotype.Component;

@Component("dao")
public class DaoImpl implements IDao{

    @Override
    public double getData() {
        /*
        connect to data base to get température
        */
        System.out.println("Version base de donnée");

        double temp=Math.random()*40;
        return temp;
    }
}
```

The middle window displays a Java file named 'MetierImpl.java' with the following content:

```
import dao.IDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class MetierImpl implements IMetier {
    // Couplage faible

    private IDao dao;

    public MetierImpl(IDao dao) {
        this.dao = dao;
    }

    @Override
    // ...
}
```

The right window displays a Java file named 'PresSpringAnnotations.java' with the following content:

```
package pres;

import metier.IMetier;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class PresSpringAnnotations {

    public static void main(String[] args) {
        ApplicationContext context=
            new AnnotationConfigApplicationContext(
                (...basePackages: "dao","metier","ext");
        IMetier metier=context.getBean(IMetier.class);
        System.out.println(metier.calcul());
    }
}
```

Below these windows is a 'Run' console window showing the execution of 'PresSpringAnnotations'. The output is:

```
"C:\Program Files\Java\jdk1.8.0_301\bin\java.exe" ...
version base de donnée
21649.9042821449
Process finished with exit code 0
```

Onget 6

SCREEN SHOTS

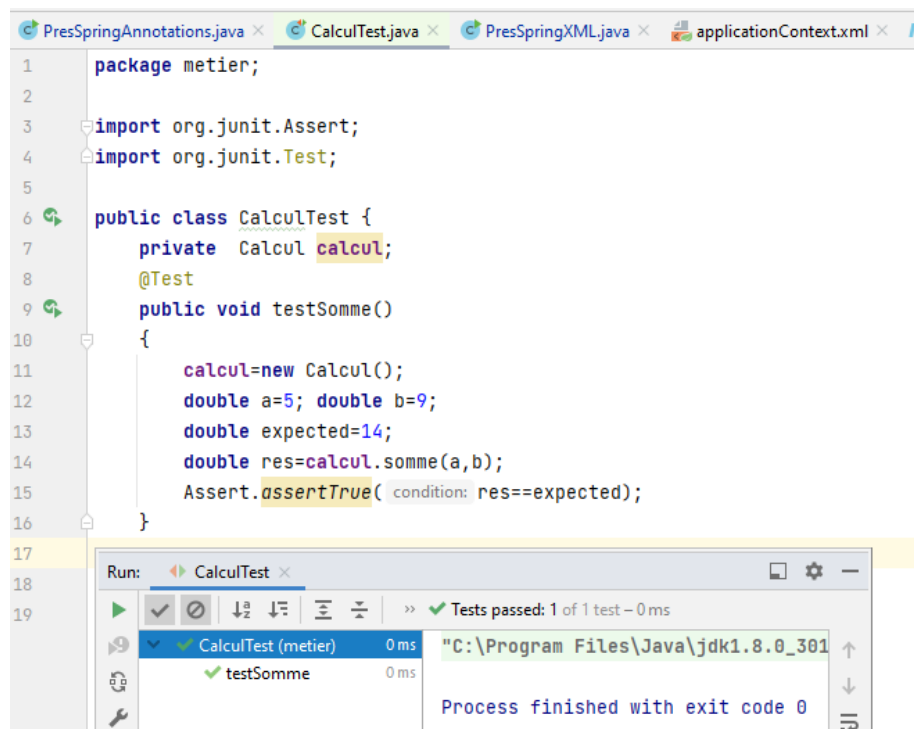
Maven, géré par l'organisation Apache Software Foundation. (Jakarta Project), est un outil pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier. L'objectif recherché est de

- produire un logiciel à partir de ses sources,
- en optimisant les tâches réalisées à cette fin
- et en garantissant le bon ordre de fabrication.

Onglet 7



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>ma.emsi</groupId>
8   <artifactId>emsi-ioc-2</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11   <properties>
12     <maven.compiler.source>8</maven.compiler.source>
13     <maven.compiler.target>8</maven.compiler.target>
14   </properties>
```



```
1 package metier;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 public class CalculTest {
7     private Calcul calcul;
8     @Test
9     public void testSomme()
10    {
11        calcul=new Calcul();
12        double a=5; double b=9;
13        double expected=14;
14        double res=calcul.somme(a,b);
15        Assert.assertTrue( condition: res==expected);
16    }
17 }
```

Run: CalculTest

Tests passed: 1 of 1 test - 0 ms

CalculTest (metier) 0 ms

testSomme 0 ms

"C:\Program Files\Java\jdk1.8.0_301"

Process finished with exit code 0

Le principe d'ouverture et de fermeture est difficile à comprendre, c'est parce que, quels changements de code sont définis comme 'Extension' ? Quels changements de code sont définis comme 'Modifier' ? Qu'est-ce qui constitue une satisfaction ou une violation Principe d'ouverture et de fermeture ? C'est ça ? C'est difficile à comprendre. La raison pour laquelle ce principe est difficile à maîtriser, c'est parce que, Comment faire Ouvert à l'extension, Modifier la fermeture ?

C'est pour ça le « Spring Boot » est un Framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en JAR, totalement autonome. Ce qui nous intéresse particulièrement.