Compte rendu : Injection des dépendances

J2EE

SALMA MAHDAD

4IIR G21

Injection des dépendances et couplage faible

L'injection de dépendances est une technique de programmation qui consiste à fournir les dépendances nécessaires à une classe plutôt que de les créer à l'intérieur de la classe elle-même.

Elle permet de mettre en œuvre le principe du couplage faible en fournissant une méthode pour réduire les dépendances entre les différentes classes d'une application. En utilisant l'injection de dépendances, les classes peuvent être développées de manière indépendante, sans connaître les détails de mise en œuvre des autres classes avec lesquelles elles interagissent.

→Le couplage faible est basé sur les interfaces.

Dans le TP1, nous avons créé une interface "IDao" qui spécifie les comportements que tous les objets DAO doivent avoir.

```
package Dao.dao;

public interface IDao {
 double getData();


}
```

En utilisant cette interface pour modéliser le couplage faible, on peut facilement ajouter plusieurs classes Dao en implémentant simplement cette interface.

```
package Dao.dao;

public class DaoImpl implements IDao{
    @Override
    public double getData() {
       return Math.random()*40;

    }
}
```

Nous avons aussi créé une interface "IMetier" qui spécifie les comportements que de nos objets métier.

```
package metier;

public interface Imetier
{
    double calcul();
```

```
}
```

En utilisant cette interface pour modéliser le couplage faible, on peut facilement ajouter plusieurs classes métiers à notre application en implémentant simplement cette interface.

```
package metier;

import Dao.dao.IDao;

public class MetierImpl implements Imetier {
    private IDao dao;
    @Override
    public double calcul() {
        double tmp=dao.getData();
        double res=tmp*540/Math.cos(tmp*Math.PI);
        return 0;
    }

    public void setDao(IDao dao) {
        this.dao = dao;
    }
}
```

Grace au couplage faible, on peut créer une application plus modulaire et flexible, facilitant ainsi la maintenance et l'extension du code.

L'injection de dépendances peut être effectuée sans Spring ou avec Spring. Dans ce compte rendu, ce sera sans Spring.

➔ **Avec instanciation statique :**

La classe de présentation suivante instancie statiquement les objets des classes de notre application.

```
package Pres;

import Dao.dao.DaoImpl;
import metier.MetierImpl;

public class PresStatique {
    public static void main(String[] args){
        DaoImpl dao= new DaoImpl();

        MetierImpl metier=new MetierImpl();
        metier.setDao(dao);
        System.out.println("resultat="+metier.calcul());
    }
}
```

➔ **Avec instanciation dynamique :**

On crée un fichier config.txt, ou précise les différentes classes de notre application liées par un couplage faible.

```
Dao.dao.DaoImpl
metier.MetierImpl
```

La classe de présentation suivante instancie dynamiquement les objets

```
package Pres;

import Dao.dao.IDao;

import java.io.File;
import java.util.Scanner;

public class PresDynamique {
    public static void main(String[] args) throws Exception {
        Scanner scanner=new Scanner(new File("config.txt"));
        String daoMaclasse=scanner.nextLine();
        Class cDao=Class.forName(daoMaclasse);
        IDao dao =(IDao) cDao.newInstance();
        System.out.println(dao.getData());

    }
}
```