TP 1 : Utilisation de la Programmation Dynamique avec Java

Pr. Mohamed LACHGAR

Objectif

L'objectif de ce TP consiste à comprendre l'utilisation de la programmation dynamique en Java pour instancier des classes, injecter des dépendances et invoquer des méthodes via la réflexion. Cela permet de charger dynamiquement des classes et de configurer une application sans dépendre de la configuration statique.

Contexte

Dans ce TP, la création d'une application utilisant des interfaces et des classes pour le calcul d'une valeur est requise. L'application devra utiliser la réflexion pour charger les implémentations de ces interfaces à partir d'un fichier de configuration, permettant ainsi une flexibilité dans la gestion des dépendances.

Étapes du TP

1. Création de l'interface `IDao`

Créer un package nommé `dao` et ajouter l'interface `IDao`. Cette interface doit définir une méthode `getValue` qui retourne un nombre de type `double`.

Listing 1: Interface `IDao`

```
package dao;

public interface IDao {
   double getValue();
}
```

2. Création de l'implémentation `DaoImpl`

Ajouter une classe `DaoImpl` dans le même package qui implémente l'interface `IDao`. La méthode `getValue` doit retourner une valeur fixe, par exemple, `100.0`.

Listing 2: Classe `DaoImpl`

```
package dao;

public class DaoImpl implements IDao {
   @Override
   public double getValue() {
      return 100.0;
   }
}
```

3. Création de l'interface `IMetier`

Créer un package `metier` et y ajouter l'interface `IMetier`. Cette interface doit définir une méthode `calcul` qui retourne également un `double`.

Listing 3: Interface `IMetier`

```
package metier;

public interface IMetier {
   double calcul();
}
```

4. Création de l'implémentation `MetierImpl`

Ajouter une classe `MetierImpl` dans le package `metier` qui implémente l'interface `IMetier`. Cette classe doit inclure une variable de type `IDao` et une méthode `setDao` pour injecter l'instance de `IDao`. La méthode `calcul` doit appeler `getValue` de `IDao` et retourner le double de cette valeur.

Listing 4: Classe `MetierImpl`

```
package metier;

import dao.IDao;

public class MetierImpl implements IMetier {
```

```
    private IDao dao;

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

    @Override
    public double calcul() {
        return dao.getValue() * 2;
    }
}
```

5. Création de la classe `Presentation2`

Dans un package `presentation`, créer la classe `Presentation2` qui utilisera la réflexion pour charger les classes et injecter les dépendances. Cette classe lira les noms des classes depuis un fichier de configuration et les instanciera dynamiquement.

Listing 5: Classe `Presentation2`

```
package presentation;

import dao.IDao;
import metier.IMetier;

import java.io.File;
import java.lang.reflect.Method;
import java.util.Scanner;

public class Presentation2 {
    public static void main(String[] args) throws Exception {
        // Lecture du nom de la classe DAO depuis le fichier de
            configuration
        Scanner scanner = new Scanner(new File("config.txt"));
        String daoClassName = scanner.nextLine();

        // Utilisation de la réflexion pour charger la classe DAO
            et créer une instance
        Class<?> cDao = Class.forName(daoClassName);
        IDao dao = (IDao)
            cDao.getDeclaredConstructor().newInstance();
```

```
        // Lecture du nom de la classe Métier depuis le fichier
            de configuration
        String metierClassName = scanner.nextLine();
        Class<?> cMetier = Class.forName(metierClassName);
        IMetier metier = (IMetier)
            cMetier.getDeclaredConstructor().newInstance();

        // Injection de la DAO dans le Métier àl'aide de la
            réflexion
        Method setDaoMethod = cMetier.getMethod("setDao",
            IDao.class);
        setDaoMethod.invoke(metier, dao);

        // Invocation d'une méthode sur l'instance de Métier et
            affichage du résultat
        System.out.println("Résultats = " + metier.calcul());

        scanner.close();
    }
}
```

Configuration

Créer un fichier `config.txt` dans le répertoire racine du projet. Ce fichier doit contenir les noms complets des classes `DaoImpl` et `MetierImpl`, chacun sur une ligne :

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

Exécution du projet

Compiler et exécuter la classe `Presentation2`. Si tout est correctement configuré, le résultat attendu dans la console sera :

```
Résultats = 200.0
```

Conclusion

Ce TP permet de comprendre les principes de la programmation dynamique en Java. L'utilisation de la réflexion permet de créer des applications plus flexibles et modulaires. L'injection de dépendances via la réflexion facilite également la maintenance et l'évolution du code.