

Activité Pratique N°1

Inversion de contrôle et Injection des dépendances



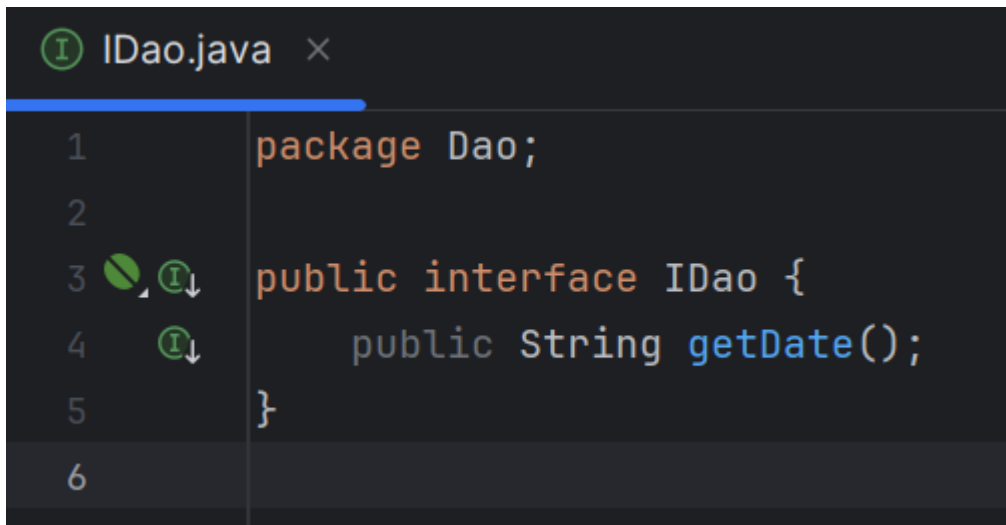
Réalisé par :

Zakariae ABDOUNI

Encadré par :

Mohammed YOUSSEFI

- Créer l'interface IDao avec une méthode getDate



The screenshot shows a code editor window titled 'IDao.java'. The code defines a package 'Dao' and a public interface 'IDao' with a single method 'getDate()' that returns a 'String'. The interface is implemented by the 'DaoImp' class, which is shown in the next screenshot. The code is as follows:

```
1 package Dao;
2
3 public interface IDao {
4     public String getDate();
5 }
6
```

- Créer une implémentation de cette interface



The screenshot shows a code editor window with the implementation of the 'IDao' interface. The code defines a package 'Dao', imports 'org.springframework.stereotype.Component', and creates a public class 'DaoImp' that implements 'IDao'. The 'getDate()' method returns a new 'Date()' object converted to a string. The code is as follows:

```
package Dao;

import org.springframework.stereotype.Component;

@Component("dao")
public class DaoImp implements IDao{
    public String getDate() {
        return new java.util.Date().toString();
    }
}
```

- Créer l'interface IMetier avec une méthode calcul

```
package Metier;  
  
public interface IMetier {  
    public int calcul();  
}
```

- Créer une implémentation de cette interface en utilisant le couplage faible

```
package Metier;  
  
import ...  
  
@Component("metier")  
public class MetierImp implements IMetier{  
  
    private IDao dao; //couplage faible  
  
    public MetierImp(IDao dao) { this.dao = dao; }  
  
    public void setDao(IDao dao) { this.dao = dao; }  
    public int calcul() {  
        String date = dao.getDate();  
  
        return 30 - Integer.parseInt(date.substring(8, 10));  
    }  
  
    // Injection dans la variable dao un objet d'une class qui implémente l'interface IDao  
    public void SetDao(IDao dao) { this.dao = dao; }  
  
}
```

- Faire l'injection des dépendances :

1- Par instantiation statique

```
package Presentation;

import Dao.DaoImp;
import Metier.MetierImp;
💡

public class Main {
    public static void main(String[] args) {

        DaoImp dao = new DaoImp();
        MetierImp metier = new MetierImp(dao);

        String days = metier.calcul() + " jours restants";
        System.out.println("resultat avec injection statique = "
            + days);
    }
}
```

2- Par instantiation dynamique

```
package Presentation;

import ...
💡

public class Main2 {
    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(new File("src/main/resources/config.txt"));
            String daoClassName = scanner.nextLine();
            Class cDao = Class.forName(daoClassName);
            IDao dao = (IDao) cDao.newInstance();
            String metierClassName = scanner.nextLine();
            Class cMetier = Class.forName(metierClassName);
            IMetier metier = (IMetier) cMetier.getConstructor(IDao.class).newInstance(dao);

            Method method = cMetier.getMethod("setDao", IDao.class);
            method.invoke(metier, dao);
            String days = metier.calcul() + " jours restants";
            System.out.println("resultat avec injection dynamique = " + days);
        } catch (FileNotFoundException | ClassNotFoundException | InstantiationException | IllegalAccessException |
            InvocationTargetException | NoSuchMethodException e) {
            throw new RuntimeException(e);
        }
    }
}
```

3- En utilisant le Framework Spring

```
package Presentation;

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

public class SpringXML {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext( configLocation: "applicationContext.xml");
        IMetier metier = (IMetier) context.getBean( name: "metier");
        String days = metier.calcul() + " jours restants";
        System.out.println("resultat avec Spring XML = " + days);
        ((ClassPathXmlApplicationContext) context).close();
    }
}
```

```
package Presentation;

import Dao.IDao;
import Metier.IMetier;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class SpringAnnotation {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext( ...basePackages: "Dao", "Metier");
        IMetier metier = context.getBean(IMetier.class);
        String days = metier.calcul() + " jours restants";
        System.out.println("resultat avec Spring Annotation = " + days);
    }
}
```