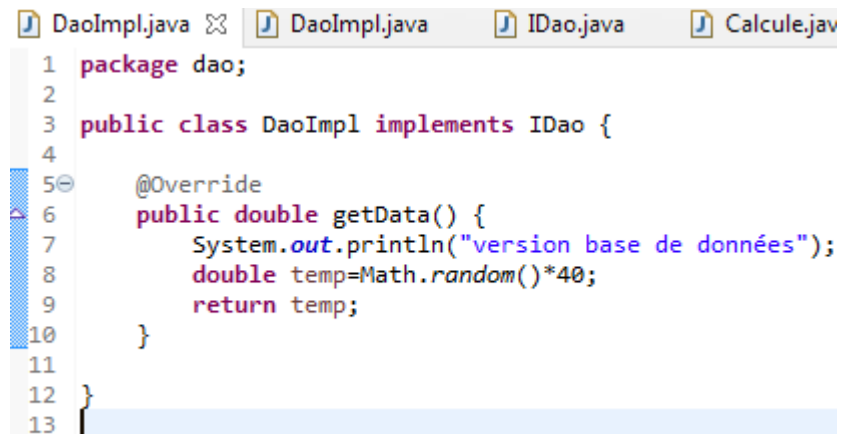


1. Créer l'interface IDao

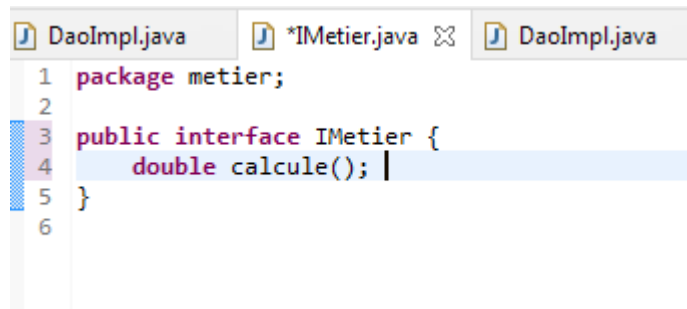
```
1 package dao;  
2  
3 public interface IDao {  
4     double getData();  
5 }  
6
```

2. Créer une implémentation de cette interface



```
DaoImpl.java x DaoImpl.java IDao.java Calcule.jav  
1 package dao;  
2  
3 public class DaoImpl implements IDao {  
4  
5     @Override  
6     public double getData() {  
7         System.out.println("version base de données");  
8         double temp=Math.random()*40;  
9         return temp;  
10    }  
11  
12 }  
13
```

3. Créer l'interface IMetier



```
DaoImpl.java IMetier.java x DaoImpl.java  
1 package metier;  
2  
3 public interface IMetier {  
4     double calcule();  
5 }  
6
```

4. Créer une implémentation de cette interface en utilisant le couplage faible

```

DaoImpl.java  IMetier.java  *MetierImpl...  DaoImpl.java  IDao.j
1  package metier;
2
3  import dao.IDao;
4
5  public class MetierImpl implements IMetier {
6      private IDao dao; //couplage faible pas de new ( couplage fort)
7
8      public IDao getDao() {
9          return dao;
10     }
11
12     public void setDao(IDao dao) {
13         this.dao = dao;
14     }
15     @Override
16     public double calcule() {
17         double tmp = dao.getData();
18         double res = tmp*540/Math.cos(tmp*Math.PI);
19         return res;
20     }
21

```

5. Faire l'injection des dépendances :
- Par instantiation statique

```

DaoImpl.java  IMetier.java  MetierImpl.java  *Presentati...  DaoImpl.java  IDao.java  »14
1  presentation;
2
3  dao.DaoImpl;
4
5  class Presentation {
6
7      public static void main(String[] args) {
8          injection des dependances par instantiation statique => new => couplage fort => vrai prob de maintenance
9
10
11      DaoImpl dao = new DaoImpl();
12      MetierImpl metier = new MetierImpl();
13      metier.setDao(dao);
14      System.out.println(metier.calcule());
15

```

- Par instantiation dynamique

```

public class Presentation2 {
    public static void main(String[] args) throws Exception
    {
        //instanciation dynamique
        Scanner scanner = new Scanner(new File("config.txt"));
        String daoClassName= scanner.nextLine(); //savoir le nom de la classe
        Class cDao= Class.forName(daoClassName); // je demande de charger une classe dans la memoire

        IDao dao=(IDao) cDao.newInstance(); //cree un objet metier qui peut etre diaImp ou diaoimpl2... dan

        //creer un objet metier
        String metierClassName = scanner.nextLine();
        Class cMetier= Class.forName(metierClassName);
        IMetier metier= (IMetier) cMetier.newInstance(); //cree un objet metier
        //injection

        Method method= cMetier.getMethod("setDao", IDao.class); // je cherche une fonction setDao qui a un
        method.invoke(metier, dao); //metier.setDao(dao)
        System.out.println("Resultat=>"+metier.calcule());
    }
}

```

c. En utilisant le Framework Spring

- Version XML

Le fichier applicationContext.xml :

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="
5         http://www.springframework.org/schema/beans
6         http://www.springframework.org/schema/beans/spring-beans.xsd
7         http://www.springframework.org/schema/tx
8         http://www.springframework.org/schema/tx/spring-tx.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context.xsd
11        http://www.springframework.org/schema/aop
12        http://www.springframework.org/schema/aop/spring-aop.xsd">
13 <bean id="dao" class="dao.DaoImpl"></bean>
14 <bean id="metier" class="metier.MetierImpl">
15   <property name="dao" ref="dao"></property>
16 </bean>
17 </beans>
```

La classe présentation :

```
package presentation;

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

import metier.IMetier;

public class PresentationSpringXML {
    public static void main(String[] args) {
        ApplicationContext context=
            new ClassPathXmlApplicationContext("applicationContext.xml");
        IMetier metier= (IMetier) context.getBean("metier"); //donne moi un bean qui s'appel metier
        System.out.println("Resultat=>"+metier.calculer());
    }
}
```

- Version annotations

La classe DaoImpl :

```
package dao;

import org.springframework.stereotype.Component;

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

    public double getData() {
        System.out.println("version base de données");
        double temp=Math.random()*40;
        return temp;
    }
}
```

La classe MetierImpl :

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

import dao.IDao;
@Component("metier")
public class MetierImpl implements IMetier {

    @Autowired //au mement ou spring vas instancier la classe metier
    @Qualifier("dao") //injecter une instance précise lorsqu on a ;
    private IDao dao; //couplage faible pas de new ( couplage fort)

    public IDao getDao() {
        return dao;
    }

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

    //injecter dans la variable dao un objet d'une classe qui imple
    public void setDao(IDao dao) {
        this.dao = dao;
    }

    public double calcule() {
        double tmp = dao.getData();
        double res = tmp*540/Math.cos(tmp*Math.PI);
        return res;
    }
}
```

La classe présentation :

```
package presentation;

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

import metier.IMetier;

public class PresentationSpringAnnotations {
    public static void main(String[] args) {
        //dao et metier sont les noms des packages qu'il faut analyser
        ApplicationContext context = new AnnotationConfigApplicationContext("dao","metier");
        IMetier metier = context.getBean(IMetier.class); //demander un bean qui implemente l'interface IMetier
        System.out.println(metier.calcule());
    }
}
```