**Rapport de Activité pratique N°1 :**

**Soukaina ELKAMOUNI – II-BDCC2**

1. Créer l'interface IDao :

package dao;  
  
public interface IDao {  
 double getTemp();  
  
}

1. Créer une implémentation de cette interface :

package dao;  
  
public class DaoImpl implements IDao{  
 @Override  
 public double getTemp() {  
 /\*  
 Se connecter à la BD pour récupérer la température  
 \*/  
 double data=24;  
 return data;  
 }  
}

1. Créer l'interface IMetier :

package metier;  
  
public interface IMetier {  
 double calcul();  
}

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

package metier;  
  
import dao.IDao;  
  
public class MetierImpl implements IMetier{  
  
  
 // Couplage faible  
 private IDao dao;  
  
 @Override  
 public double calcul() {  
 double temp = dao.getTemp();  
 double res = temp\*12;  
 return res;  
 }  
  
 public IDao getDao() {  
 return dao;  
 }  
  
 public void setDao(IDao dao) {  
 this.dao = dao;  
 }  
}

1. Faire l'injection des dépendances :
   1. Par instanciation statique :

package Pres;  
  
import dao.DaoImpl;  
import metier.MetierImpl;  
  
public class Presentation {  
 public static void main(String[] args) {  
 /\*  
 ========= injection des dépendances ===========  
  
 \*/  
 DaoImpl dao = new DaoImpl();  
 MetierImpl metier = new MetierImpl();  
 metier.setDao(dao);  
 System.*out*.println(metier.calcul());  
  
 }  
}

* 1. Par instanciation dynamique :

package Pres;  
  
import dao.IDao;  
import metier.IMetier;  
  
import java.io.File;  
import java.lang.reflect.Method;  
import java.util.Scanner;  
  
public class PresentationV2 {  
 public static void main(String[] args) throws Exception {  
  
 /\*  
 Invocation Dynamique  
 \*/  
 Scanner sc = new Scanner(new File("config.txt"));  
 String daoClassName = sc.nextLine();  
 //System.out.println(daoClassName);  
 Class cDao = Class.*forName*(daoClassName);  
 IDao dao = (IDao) cDao.newInstance(); // sous-casting  
 //System.out.println(dao.getTemp());  
  
 String metierClassName = sc.nextLine();  
 System.*out*.println(metierClassName);  
 Class cMetier = Class.*forName*(metierClassName);  
 IMetier metier = (IMetier) cMetier.newInstance();  
  
 Method m = cMetier.getMethod("setDao", IDao.class); // si elle contient plusieur parametre, faire ("setDao", new Class[]{IDao.class, ...})  
 m.invoke(metier, dao); // appeler la methode m dans l'objet metier, et on lui transmet l'objet dao comme parametre  
 System.*out*.println(metier.calcul());  
  
 }  
}

* 1. En utilisant le Framework Spring :
     1. Version XML :

package Pres;  
  
import metier.IMetier;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class PresSpring {  
 public static void main(String[] args){  
 ApplicationContext context=new ClassPathXmlApplicationContext("configurationSpring.xml");  
  
 IMetier metier= (IMetier) context.getBean("metier"); // recupere l'objet qui a l'id metier depuis le fichier config.xml  
  
 System.*out*.println(metier.calcul());  
  
 }  
}

* + 1. Version annotations :

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("dao", "metier","ext");  
 IMetier metier = context.getBean(IMetier.class);  
 System.*out*.println(metier.calcul());  
 }  
}