Activité 1 :

Injection des dépendances

Réaliser par :

Mohamed amine KHAMMOUR

Professeur :

Mr. Mohamed YOUSSFI

- Injection des dépendances sans utiliser le fremwork Spring :

La class DaoImpl :

```
package dao;

public class DaoImpl implements IDao{
    @Override
    public double getData() {
        //connextion base de donne pour recupirer la temperature
        System.out.println("version base de donnes");
        return Math.random()*40;
    }
}
```

L'interface IDao :

```
package dao;

public interface IDao {
      double getData();


}
```

L'interface **Imetier :**

```
package metier;

public interface Imetier {
     double calcule();


}
```

la class **ImetierImpl**

```
package metier;

import dao.IDao;
public class ImetierImpl implements Imetier {
    private IDao dao; //couplage faible

    @Override
    public double calcule() {
        return dao.getData()*540/Math.cos(Math.PI*40);
    }
//injecter dans la variable dao un objet d'une clas qui implement
l'interface dao
    public void setDao(IDao dao) {
        this.dao = dao;
    }
}
```

- ✓ la Class presntation1

```
package presentation;

import extention.DaoImpl2;
import metier.ImetierImpl;

public class presentation {
    public static void main(String[] args) {
        DaoImpl2 dao= new DaoImpl2();
        ImetierImpl metier=new ImetierImpl(dao);
        System.out.println(metier.calcule());
    }
}
```

```
E:\Java\bin\java.exe ...
version capture
2160000.0
```

- ✓ la classe prenstation2 :dans cette class on travail avec le couplage faible c'est-à-dire sans l'utilisation de new et on a utilisé le fichier de config.txt

```
package presentation;

import dao.IDao;
import metier.Imetier;

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

public class presentation2 {
    public static void main(String[] args) throws FileNotFoundException,
ClassNotFoundException, InstantiationException, IllegalAccessException,
NoSuchMethodException, InvocationTargetException {
        Scanner scanner=new Scanner(new File("config.txt"));
        String daocalssName=scanner.nextLine();//lire le premier line de
fichier text
        Class cDao=Class.forName(daocalssName);
        IDao dao=(IDao) cDao.newInstance();
        System.out.println(dao.getData());
        String metiercalssName=scanner.nextLine();//lire le premier line de
fichier text
        Class cmetier=Class.forName(metiercalssName);
        Imetier metier= (Imetier) cmetier.newInstance();
        Method method=cmetier.getMethod("setDao",IDao.class);
        method.invoke(metier,dao);
        System.out.println(metier.calcule());


    }
}
```

```
E:\Java\bin\java.exe ...
version base de donnes
23.112147962386025
version base de donnes
18013.62671108871

Process finished with exit code 0
```

- L'utilisation du Framework Spring :

✓ La version XML

Le fichier pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>InversionDuControle</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-
core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.3.16</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-
context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.16</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-
beans -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>5.3.16</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
```

```
            <version>4.12</version>
            <scope>test</scope>
        </dependency>

    </dependencies>



</project>
```

Le fichier configure.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="d" class="extention.DaoImpl2"></bean>
    <bean id="metier" class="metier.ImetierImpl">
        <property name="Dao" ref="d"></property>
    </bean>
</beans>
```

La class **PresSpringXml**

```
package presentation;

import metier.Imetier;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class PresSpringXml {
    public static void main(String[] args) {
        ApplicationContext ctx=new
ClassPathXmlApplicationContext("configure.xml");
        // Imetier metier=(Imetier) ctx.getBean("metier");
        Imetier metier=ctx.getBean(Imetier.class);
        System.out.println(metier.calcule());

    }
}
```

```
E:\Java\bin\java.exe ...
version capture
2160000.0

Process finished with exit code 0
```

- ✓ La version annotation :

La classe DaoImpl avc l'annotation @Component

```
package dao;

import org.springframework.stereotype.Component;

@Component()
public class DaoImpl implements IDao{
    @Override
    public double getData() {
        //connextion base de donne pour recupirer la temperature
        System.out.println("version base de donnes");
        return Math.random()*40;
    }
}
```

La Class ImetierImpl :avec l'annotation @Autowired

```
package metier;

import dao.IDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class ImetierImpl implements Imetier {
    private IDao dao; //couplage faible
@Autowired


    @Override
    public double calcule() {
        return dao.getData()*540/Math.cos(Math.PI*40);
    }
//injecter dans la variable dao un objet d'une clas qui implement
l'interface dao
    public void setDao(IDao dao) {
        this.dao = dao;
    }
}
```

la calss Presentation_Anotation

```
package presentation;

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

//import org.spring
public class Presentation_Anotation
{
    public static void main(String[] args) {
ApplicationContext ctx=new
AnnotationConfigApplicationContext("extention","metier");//ajouter les
pakedges qu'il faut scaner
        Imetier metier=ctx.getBean(Imetier.class);
```

```
        System.out.println(metier.calcule());
    }}
```

```
E:\Java\bin\java.exe ...
version capture
2160000.0

Process finished with exit code 0
```

- ✓ Il y a aussi la partie injection avec constructeur avec paramètre

```
On élevé l' annotation @Autowired et on ajout un constructeur avec
paramètre dans la class  ImetierImpl
```

```
package metier;

import dao.IDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class ImetierImpl implements Imetier {
    private IDao dao; //couplage faible

    public ImetierImpl(IDao dao) {
        this.dao = dao;
    }

    @Override
    public double calcule() {
        return dao.getData()*540/Math.cos(Math.PI*40);
    }
//injecter dans la variable dao un objet d'une clas qui implement
l'interface dao
    public void setDao(IDao dao) {
        this.dao = dao;
    }
}
```

- ✓ La partie xml si on veut utiliser un constructeur avec paramètre

```
On va éliminer la balise <property name="Dao" ref="d"></property>
et on fait  <constructor-arg ref="d"></constructor-arg>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="d" class="extention.DaoImpl2"></bean>
    <bean id="metier" class="metier.ImetierImpl">
        <!--<property name="Dao" ref="d"></property>-->
        <constructor-arg ref="d"></constructor-arg>
    </bean>
</beans>
```

- ✓ La partie test :

La classe calcule_Test

```java
package metier;

import org.junit.Assert;
import org.junit.Test;
public class calcule_test {
    private metier.calcule calcule;
    @Test
    public void somme_test(){
        calcule=new calcule();
        double a=5,b=14;
        double expected=19;
        double res= calcule.somme(a,b);
        Assert.assertTrue(res==expected);
    }
}
```