

# JAVA PROGRAMMING COURSE

## EXERCISE

# INTERFACES IN JAVA

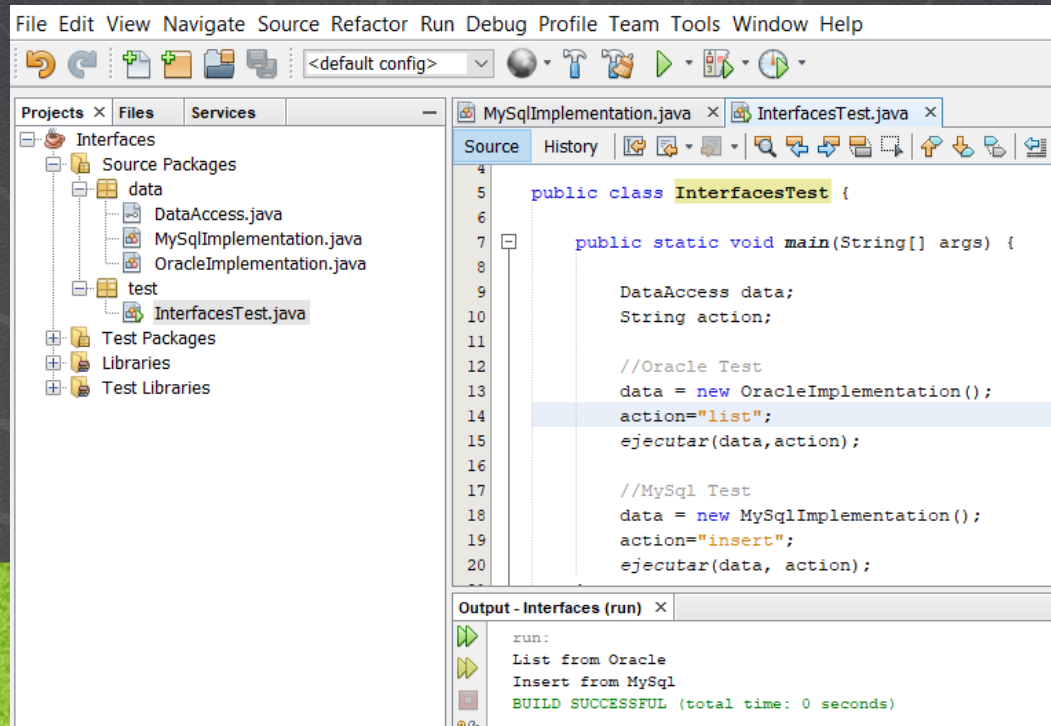


JAVA PROGRAMMING COURSE

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

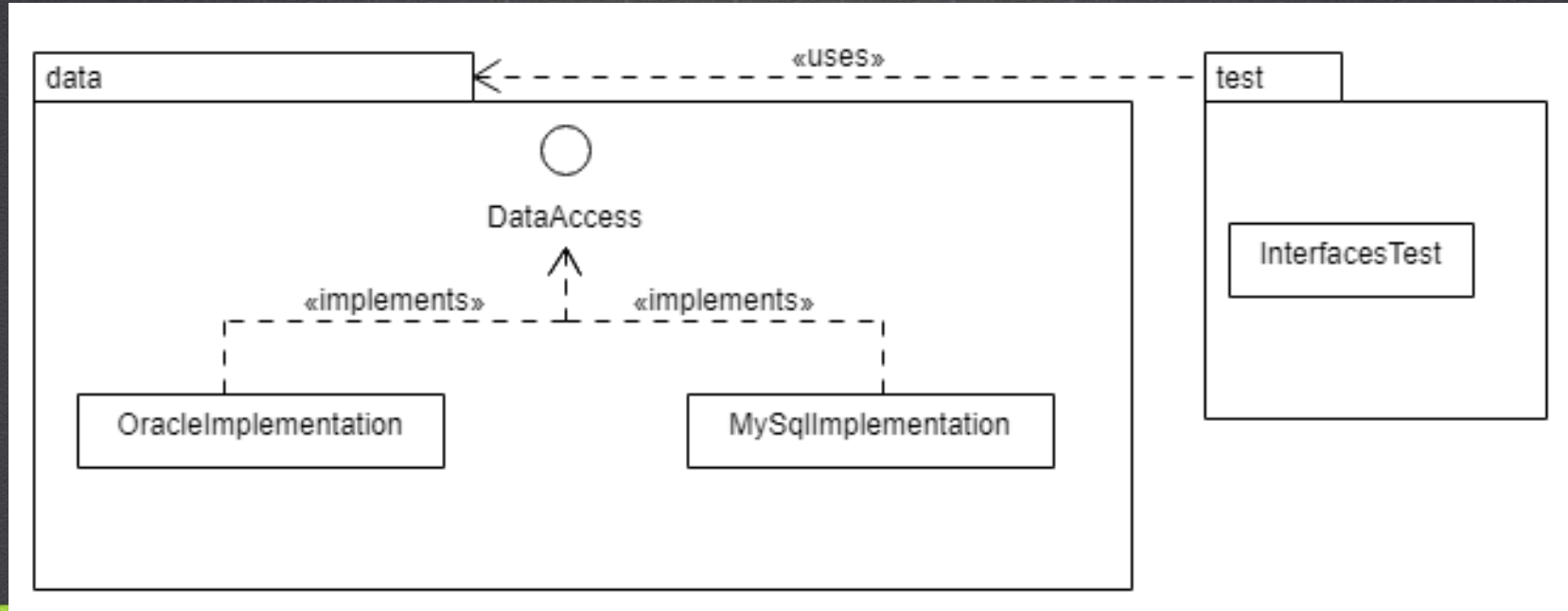
# EXERCISE OBJECTIVE

Put into practice the concept of interfaces in Java. At the end we should observe the following:



# CLASS DIAGRAM OF THE EXERCISE

This is the class diagram of the exercise:





# 1. CREATE A NEW PROJECT

Create a new project:

**New Java Application**

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ **Create Main Class**

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## 2. CREATE A NEW CLASS

Create a new class:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

### 3. MODIFY THE CODE

DataAccess.java:

```
package data;

public interface DataAccess {

    public static int MAX_RECORDS = 10;

    public abstract void insert();

    public abstract void list();

}
```



## 4. CREATE A NEW CLASS

Create a new class:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name: OracleImplementation

Project: Interfaces

Location: Source Packages

Package: data

Created File: C:\Courses\JavaProgramming\Lesson15\Interfaces\src\data\OracleImplementation.java

< Back   Next >   **Finish**   Cancel   Help

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## 5. MODIFY THE CODE

### OracleImplementation.java:

```
package data;

public class OracleImplementation implements DataAccess{

    @Override
    public void insert() {
        System.out.println("Insert from Oracle");
    }

    @Override
    public void list() {
        System.out.println("List from Oracle");
    }

}
```



## 6. CREATE A NEW CLASS

Create a new class:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# 7. MODIFY THE CODE

## MySqlImplementation.java:

```
package data;

public class MySqlImplementation implements DataAccess{

    @Override
    public void insert() {
        System.out.println("Insert from MySql");
    }

    @Override
    public void list() {
        System.out.println("List from MySql");
    }
}
```

## 8. CREATE A NEW CLASS

Create a new class:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



# 9. MODIFY THE CODE

## InterfacesTest.java:

```
package test;

import data.*;

public class InterfacesTest {

    public static void main(String[] args) {

        DataAccess data = null;
        String action = null;

        //Oracle Test
        data = new OracleImplementation();
        action="list";
        ejecutar(data,action);

        //MySQL Test
        data = new MySQLImplementation();
        action="insert";
        ejecutar(data, action);
    }
}
```

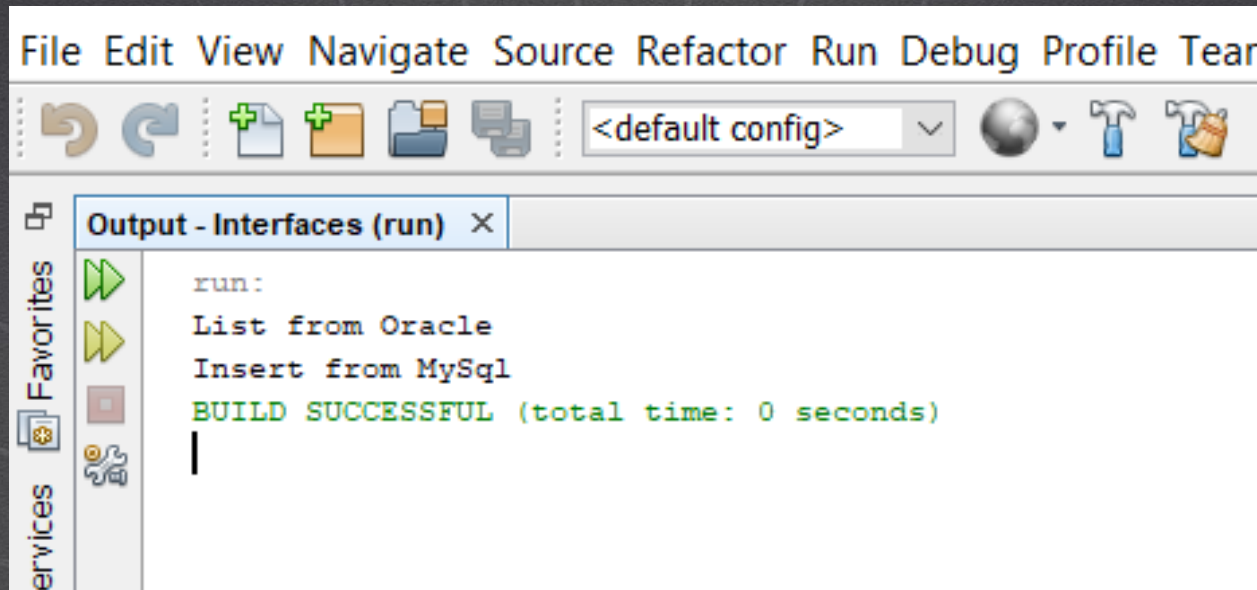
## 9. MODIFY THE CODE

### InterfacesTest.java:

```
private static void ejecutar(DataAccess data, String action){  
    if("list".equals(action)){  
        data.list();  
    }  
    else if("insert".equals(action)){  
        data.insert();  
    }  
}
```

# 10. EXECUTE THE PROJECT

The result is as follows:



**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



# EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of interfaces in Java.
- We have seen how the concept of interfaces in Java is similar to abstract classes, however, we can implement multiple interfaces, instead we can only extend a class in Java, these are just some of the differences and as we go we will see more the use of interfaces in Java.