

JAVA PROGRAMMING COURSE

EXERCISE

EXCEPTION HANDLING 2

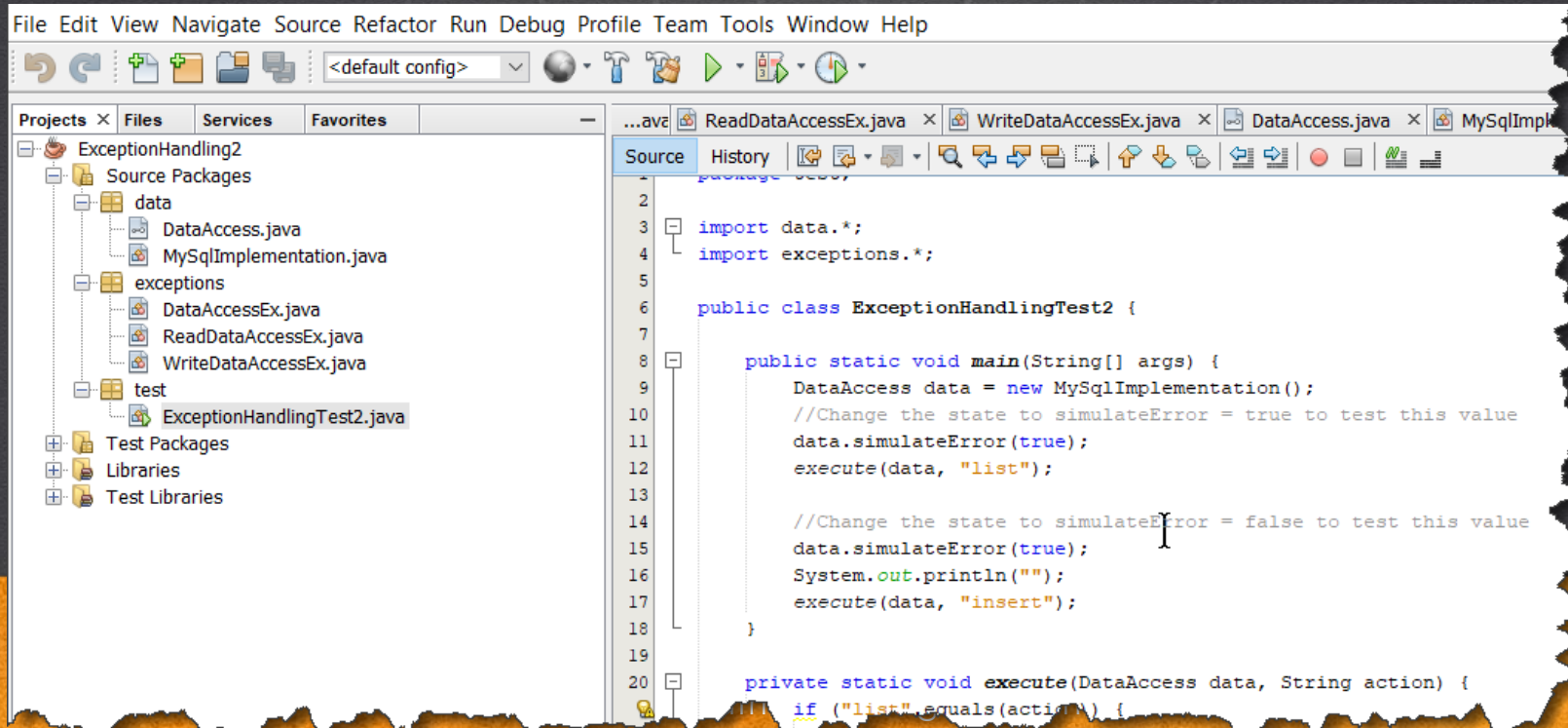


JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create an exercise for more advanced use of exceptions. At the end we should observe the following:



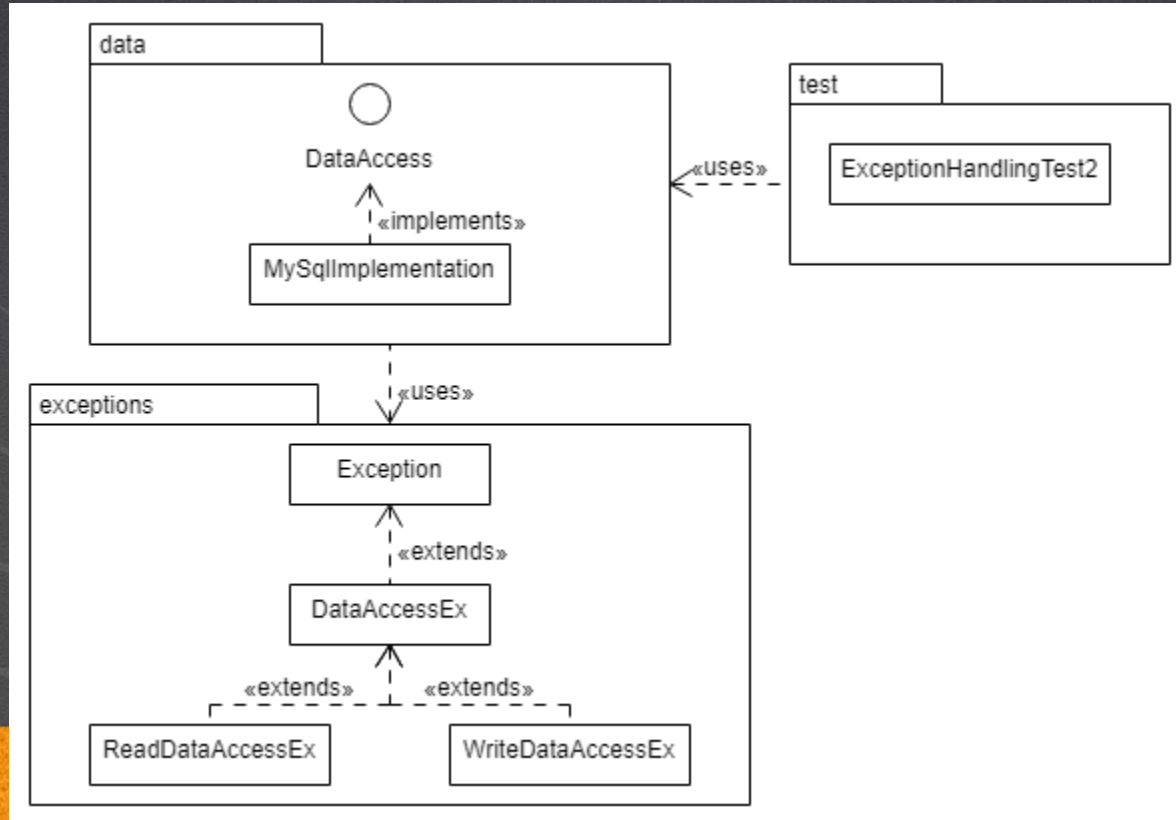
The screenshot shows an IDE window with a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help) and a toolbar. The left sidebar displays a project tree for 'ExceptionHandling2' with the following structure:

- Source Packages
 - data
 - DataSource.java
 - MySQLImplementation.java
 - exceptions
 - DataSourceEx.java
 - ReadDataSourceEx.java
 - WriteDataSourceEx.java
 - test
 - ExceptionHandlingTest2.java
- Test Packages
- Libraries
- Test Libraries

The main editor window shows the 'Source' view of 'ExceptionHandlingTest2.java'. The code is as follows:

```
1 package com.example;
2
3 import data.*;
4 import exceptions.*;
5
6 public class ExceptionHandlingTest2 {
7
8     public static void main(String[] args) {
9         DataSource data = new MySQLImplementation();
10        //Change the state to simulateError = true to test this value
11        data.simulateError(true);
12        execute(data, "list");
13
14        //Change the state to simulateError = false to test this value
15        data.simulateError(true);
16        System.out.println("");
17        execute(data, "insert");
18    }
19
20    private static void execute(DataSource data, String action) {
21        if ("list".equals(action)) {
```

DIAGRAMA DE CLASES DEL EJERCICIO



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: ExceptionHandling2

Project Location: C:\Courses\JavaProgramming\Lesson17 Browse...

Project Folder: C:\Courses\JavaProgramming\Lesson17\ExceptionHandling2

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

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

☐ Create Main Class exceptionhandling2.ExceptionHandling2

< Back Next > **Finish** Cancel Help

JAVA PROGRAMMING COURSE

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

3. MODIFY THE CODE

DataAccessEx.java:

```
package exceptions;

public class DataAccessEx extends Exception{

    public DataAccessEx(String message){
        super(message);
    }
}
```


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:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

5. MODIFY THE CODE

ReadDataAccessEx.java:

```
package exceptions;

public class DataAccessEx extends Exception{

    public DataAccessEx(String message){
        super(message);
    }
}
```


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

7. MODIFY THE CODE

WriteDataAccessEx.java :

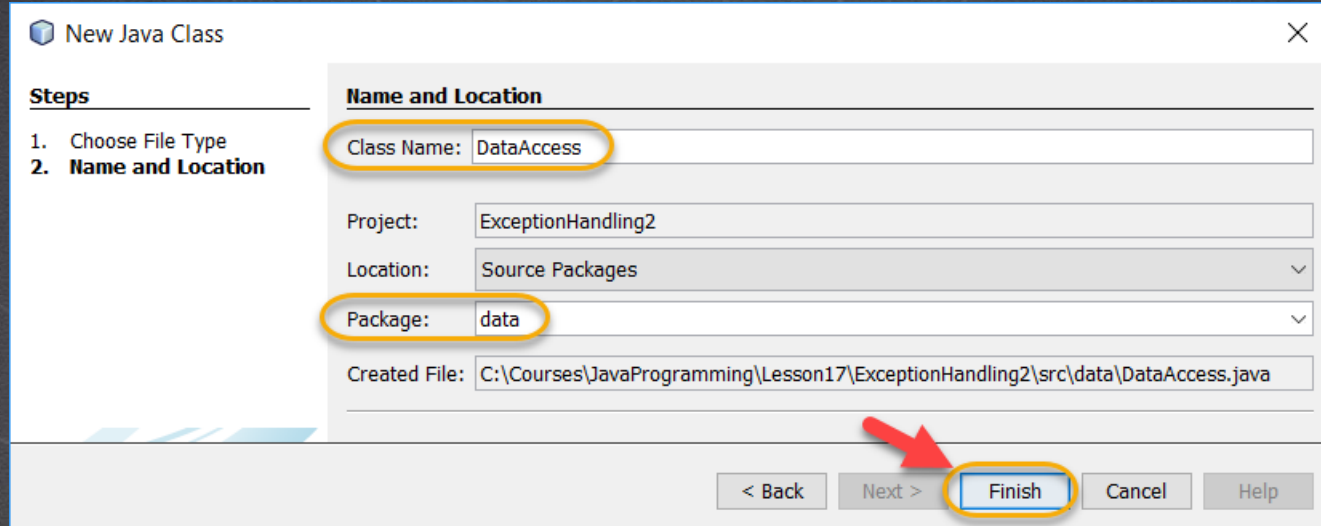
```
package exceptions;

public class WriteDataAccessEx extends DataAccessEx{

    public WriteDataAccessEx(String message) {
        super(message);
    }
}
```

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

9. MODIFY THE CODE

DataAccess.java :

```
package data;

import exceptions.DataAccessEx;

public interface DataAccess {

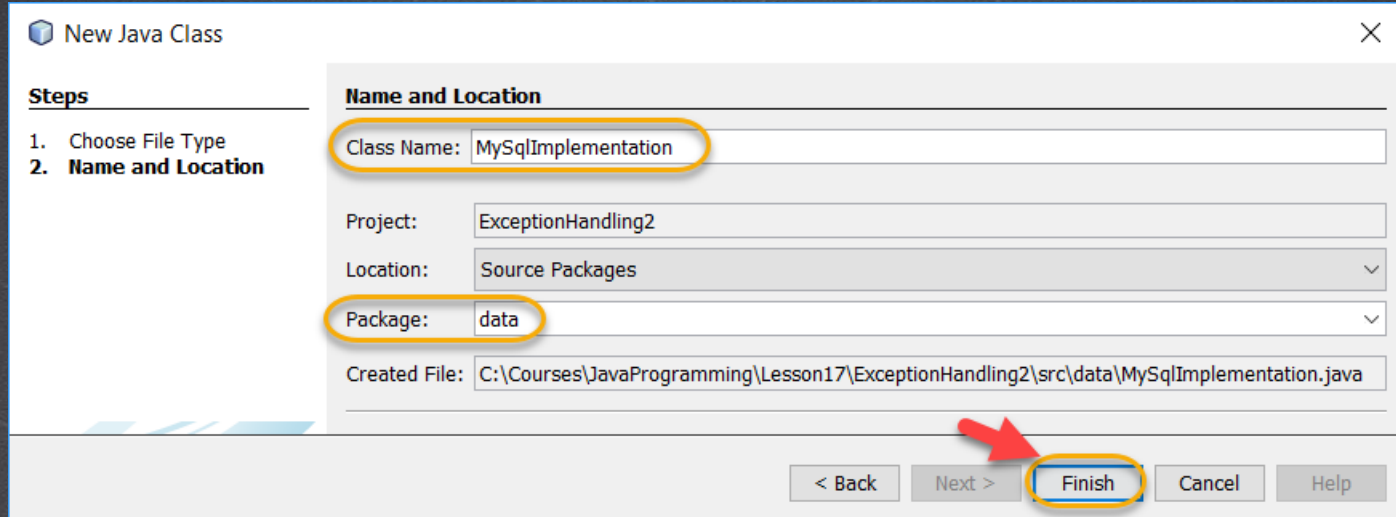
    public abstract void insert() throws DataAccessEx;

    public abstract void list() throws DataAccessEx;

    public abstract void simulateError(boolean simularError);
}
```

10. 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

11. MODIFY THE CODE

MySQLImplementation.java:

```
package data;
import exceptions.*;

public class MySQLImplementation implements DataAccess{
    private boolean simulateError;

    @Override
    public void insert() throws DataAccessEx {
        if (simulateError) {
            throw new WriteDataAccessEx("Data writing error");
        } else {
            System.out.println("Insert from MySQL");
        }
    }

    @Override
    public void list() throws DataAccessEx {
        if (simulateError) {
            throw new ReadDataAccessEx("Data reading error");
        } else {
            System.out.println("List from MySQL");
        }
    }

    public boolean isSimulateError() {
        return simulateError;
    }

    @Override
    public void simulateError(boolean simularError) {
        this.simulateError = simularError;
    }
}
```


12. 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

13. MODIFY THE CODE

ExceptionHandlingTest2.java:

```
package test;

import data.*;
import exceptions.*;

public class ExceptionHandlingTest2 {

    public static void main(String[] args) {
        DataAccess data = new MySqlImplementation();
        //Change the state to simulateError = true to test this value
        data.simulateError(true);
        execute(data, "list");

        //Change the state to simulateError = false to test this value
        data.simulateError(true);
        System.out.println("");
        execute(data, "insert");
    }
}
```

13. MODIFY THE CODE

ExceptionHandlingTest2.java:

```
private static void execute(DataAccess data, String action) {
    if ("list".equals(action)) {
        try {
            data.list();
        } //Si se van a procesar varias excepciones de la misma jerarquia
        //siempre se debe procesar primero la excepcion de menor jerarquia
        //y posteriormente la de mayor jerarquia
        catch (ReadDataAccessEx ex) {
            System.out.println("Read error: Process the most specific exception of data reading");
        } catch (DataAccessEx ex) {
            System.out.println("Data Access Error: Process the most generic exception of access to data");
        } catch (Exception ex) {
            System.out.println("General error");
        } finally {
            System.out.println("Process finally is optional, it will always run regardless of whether there was
an error or not");
        }
    }
}
```

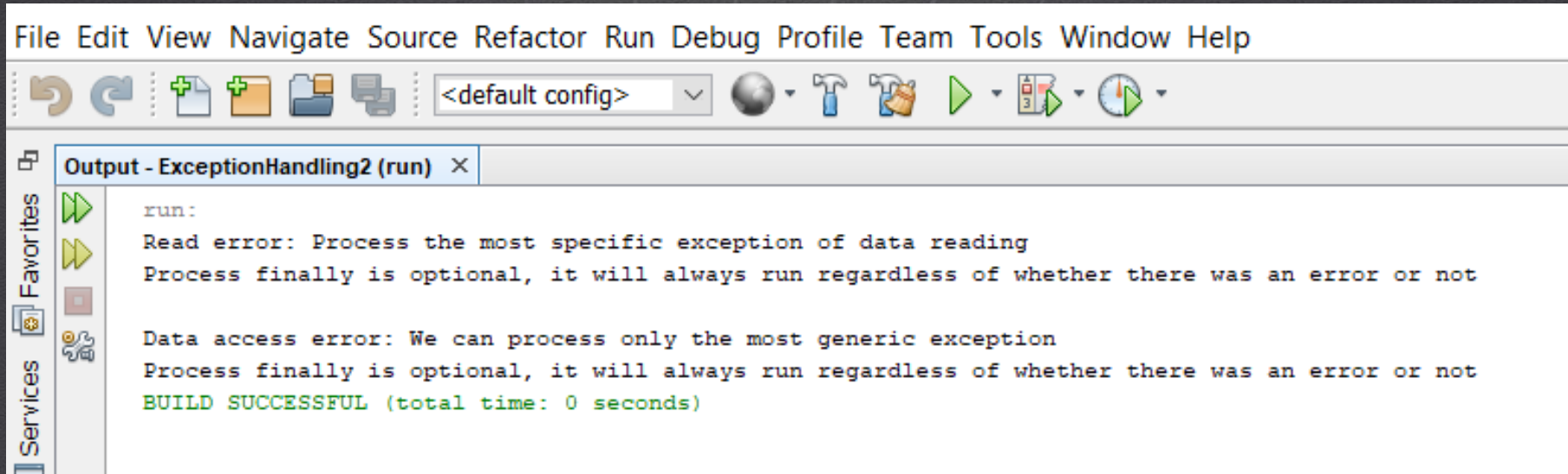

13. MODIFY THE CODE

ExceptionHandlingTest2.java:

```
    } else if ("insert".equals(action)) {
        try {
            data.insert();
        } catch (DataAccessEx ex) {
            System.out.println("Data access error: We can process only the most generic exception");
        } finally {
            System.out.println("Process finally is optional, it will always run regardless of whether there was an error or not");
        }
    } else {
        System.out.println("No known action was provided");
    }
}
```

14. EXECUTE THE PROJECT

The result is as follows:



The screenshot shows an IDE window with the title bar 'File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help'. Below the title bar is a toolbar with icons for undo, redo, new file, new folder, save, copy, paste, and a dropdown menu showing '<default config>'. The main area displays the 'Output - ExceptionHandling2 (run)' window. The output text is as follows:

```
run:  
Read error: Process the most specific exception of data reading  
Process finally is optional, it will always run regardless of whether there was an error or not  
  
Data access error: We can process only the most generic exception  
Process finally is optional, it will always run regardless of whether there was an error or not  
BUILD SUCCESSFUL (total time: 0 seconds)
```

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

EXTRA TASKS

- Test with different values and launch parents and child exceptions to check how the exception handling works according to the hierarchy of exceptions that we have declared.



JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

EXERCISE CONCLUSION

- With this exercise we have put into practice more advanced concepts of exceptions in Java.
- We could observe how to work with Exception type exceptions, it is also possible to convert these exceptions to RuntimeException type, simply by extending this class instead of the Exception class. This can be a good exercise to observe how the compiler does NOT require us to process the RuntimeException type exceptions and thus have a cleaner code.

ONLINE COURSE

JAVA PROGRAMMING

By: Eng. Ubaldo Acosta



JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx