DATA LOGGER

Technical Documentation

Mihika Kumar
MIHIKA RAGHU KUMAR SHREYA BHAT

Introduction:

This document is to outline the technical outline of the code enclosed within the Java project, outlining their purpose and what each line of code does, detailing everything from algorithm to execution.

Algorithm:

MYFORMATTER CLASS:

//Import with package keyword

Import package epita.fr.fundamental.project;

//Import with import keyword

//Import from text package in Java

Import java.text.SimpleDateFormat;

//Import from util package in Java

Import java.util.Date;

Import java.util.logging.Formatter;

Import java.util.logging.Handler;

Import java.util.logging.Level;

Import java.util.logging.LogRecord;

Initiaite class MyFormatter which extends Formatter

```
{
Initiate function public String format(LogRecord rec)
{



Assign StringBuffer buf = new StringBuffer(1000);
Invoke buf.append("<tr>\n");

// To perform WARNING in red

Initialize if loop with arguments (rec.getLevel().intValue() >= Level.WARNING.intValue())
 {
Invoke buf.append("\t<td style=\"color:red\">");
Invoke buf.append("<b>");
Invoke buf.append(rec.getLevel());
Invoke buf.append("</b>");
}
Initialize else loop
{
Invoke buf.append("\t<td>");
Invoke buf.append(rec.getLevel());
}
Invoke buf.append("</td>\n");
Invoke buf.append("\t<td>");
Invoke buf.append(calcDate(rec.getMillis()));
Invoke buf.append("</td>\n");
Invoke buf.append("\t<td>");
Invoke buf.append(formatMessage(rec));
Invoke buf.append("</td>\n");
Invoke buf.append("</tr>\n");
```

```
return buf.toString() when function is complete
}


Initialize function private String calcDate(long millisecs)
{
Assign SimpleDateFormat date_format = new SimpleDateFormat("MMM dd,yyyy HH:mm");
Assign Date resultdate = new Date(millisecs);
return date_format.format(resultdate) when function is complete
}

// this method is called just after the handler using this
// formatter is created

Invoke function public String getHead(Handler h) {
return "<!DOCTYPE html>\n<head>\n<style>\n"+ "table { width: 100% }\n"+ "th { font:bold 10pt Tahoma; }\n"+ "td { font:normal 10pt Tahoma; }\n"+ "h1 {font:normal 11pt Tahoma;}\n"+ "</style>\n"+ "</head>\n"+ "<body>\n"+ "<h1>" + (new Date()) + "</h1>\n"+ "<table border=\"0\" cellpadding=\"5\" cellspacing=\"3\">\n"+ "<tr align=\"left\">\n"+ "\t<th style=\"width:10%\">Loglevel</th>\n"+ "\t<th style=\"width:15%\">Time</th>\n"+ "\t<th style=\"width:75%\">Log Message</th>\n"+ "</tr>\n" upon function call
}

// this method is called just after the handler using this
// formatter is closed

Invoke public String getTail(Handler h)
{
return "</table>\n</body>\n</html>" upon function call
}
}
```

MYLOGGER CLASS:

```
//Import with package keyword
Import package epita.fr.fundamental.project;

//Import with import keyword
//Import from io package in java
Import  java.io.IOException;

//Import from util package in java
Import java.util.logging.FileHandler;
Import java.util.logging.Formatter;
Import java.util.logging.Level;
Import java.util.logging.Logger;
Import java.util.logging.SimpleFormatter;
Import java.util.logging.Handler;
Import java.util.logging.ConsoleHandler;

Initialize public class MyLogger
{
//Variable definitions inherited from already-defined classes

Initialize staticprivate FileHandler fileTxt;
Initialize static private SimpleFormatter formatterTxt;
Initialize static private FileHandler fileHTML;
Initialize static private Formatter formatterHTML;
```

Initialize function static public void setup() throws IOException {

// get the global logger to configure it

Initialize and Assign Logger logger =
Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);

//suppress the logging output to the console

Initialize and Assign Logger rootLogger = Logger.getLogger("");

Initialize and Assign Handler[] handlers = rootLogger.getHandlers();

Initialize if loop with argument (handlers[0] instanceof ConsoleHandler)

{

Invoke rootLogger.removeHandler(handlers[0]);

}

Invoke logger.setLevel(Level.INFO);

Assign fileTxt = new FileHandler("Logging.txt");

Assign fileHTML = new FileHandler("Logging.html");

// create a TXT formatter

Assign formatterTxt = new SimpleFormatter();

Invoke fileTxt.setFormatter(formatterTxt);

Invoke logger.addHandler(fileTxt);

// create an HTML formatter

Assign formatterHTML = new MyFormatter();

Assign fileHTML.setFormatter(formatterHTML);

Invoke logger.addHandler(fileHTML);

}

}

Create_Identity CLASS:

//Import with package keyword

Import epita.fr.fundamental.project.MyLogger;

Import package epita.fr.fundamental.project;

//Import from sql package in java

Import java.sql.Connection;

Import java.sql.PreparedStatement;

Import java.sql.ResultSet;

Import java.sql.SQLException;

Import java.sql.DriverManager;

//Import from util package in Java

Import java.util.List;

Import java.util.Scanner;

Import java.util.logging.Level;

Import java.util.logging.Logger;

Initialize class CreateIdentity

Initialize function main() of type public static void with arguments String[] args

Declaring constants:

Declare String DB_DRIVER as final, assigned to "org.apac"org.apache.derby.jdbc.EmbeddedDriver".

Declare String DB_CONNECTION as final, assigned to"jdbc:derby:MyDB;create=true";

Declare String DB_USER as final, assigned to "shreya";

Declare String DB_PASSWORD as final, assigned to "amit05";

Declaring table identity fields:

Declare String displayName ;

Declare String uid;

Declare String email;

Taking input from user;

Output "Enter display name :” through System.out.println

Take user input through variable a of type Scanner

Assign value of a to displayName

Output "Enter display name :” through System.out.println

Take user input through variable b of type Scanner

Assign value of b to uid.

Output "Enter display name :” through System.out.println

Take user input through variable c of type Scanner

Assign value of c to email.

Close a

Close b

Close c

Initialize dbConnection of type Connection as null;

Initialize preparedStatement of type PreparedStatement as null;

Create DataBase:

Initialize StringcreateTableSQL as "CREATE TABLE IDENTITIES" + " (displayName VARCHAR(255)," + " uid VARCHAR(255) PRIMARY KEY," + "email VARCHAR(255)";

Insert into Database:

Initialize String insertTableSQL as "INSERT INTO IDENTITIES (displayName, uid, email " + "values (?, ?, ?)";

Initialize try loop

Register JDBC driver by invoking forName(DB_DRIVER) in Class

Open connection

Output "Connecting to a selected database..." through System.out.println

Assign DriverManager.getConnection(DB_CONNECTION, DB_USER, DB_PASSWORD) to dbConnection

Output "Connected database successfully..." through System.out.println

Creating the DB Table

Assigning preparedStatement = dbConnection.prepareStatement(createTableSQL);

Assigning preparedStatement.setString(1,displayName);

Assigning preparedStatement.setString(2,uid);

Assigning preparedStatement.setString(3,email);

Execute insert SQL statement

Invoke executeUpdate() in preparedStatement
Invoke commit() in dbConnection

Output “Record is inserted in the DB table!" in System.out.println

Catch SQLException by providing catch loop with se of type SQLException

Handle errors for JDBC
Invoke printStackTrace() from se

Handle errors for Class.forName
Catch Exception by providing catch loop with e of type Exception
Invoke printStackTrace() from e

Invoke finally block used to close resources
Invoke try loop
Invoke if loop with argument preparedStatement!=null
Close dbConnection
Catch SQLException by providing catch loop with se of type SQLException
Do nothing in loop
Close catch loop
Invoke try loop
Invoke if loop with argument dbConnection!=null
Close dbConnection
Close if loop

Catch SQLException by providing catch loop with se of type SQLException

Invoke printStackTrace() of se

Close catch loop

Close try loop

Output "Class Insert execution complete" with System.out.println

End main

Select_Identity CLASS:

//Import via package keyword

Import package epita.fr.fundamental.project

Import epita.fr.fundamental.project.MyLogger;

//Import via import keyword:

//Import from sql package in java

Import java.sql.Connection;

Import java.sql.PreparedStatement;

Import java.sql.ResultSet;

Import java.sql.SQLException;

Import java.sql.DriverManager;

//Import from util package in java

Import java.util.List;

```
Import java.util.Scanner;
Import java.util.logging.Level;
Import java.util.logging.Logger;
Import DriverManager from sql package in java
Initialize class Select_Identity of type public
{
Initialize function main() of type public static void with arguments String[] args
{

Declaring DB Connection constants

Initialize String DB_DRIVER of type final
Assign String DB_DRIVER = "org.apache.derby.jdbc.EmbeddedDriver";
Initialize String DB_CONNECTION of type final
Assign DB_CONNECTION = "jdbc:derby:MyDB;create=true";
Initialize String DB_USER of type final
Assign DB_USER="shreya";
Initialize String DB_PASSWORD of type final
Assign DB_PASSWORD = "amit05";

Declaring identity table fields
Initialize String uid;
 Initialize Connection dbConnection = null;
Initialize PreparedStatement preparedStatement = null;

DB select
Initialize String selectTableSQL = "SELECT * FROM IDENTITIES WHERE uid = ?";
Initialize try loop
 {
```

Register JDBC driver

Invoke Class.forName(DB_DRIVER);

Open a connection

Output "Connecting to a selected database..." through System.out

Assign dbConnection = DriverManager.getConnection(DB_CONNECTION, DB_USER, DB_PASSWORD);

Output "Connected database successfully..." through System.out

Assign preparedStatement = dbConnection.prepareStatement(selectTableSQL);

Execute select SQL statement

Invoke preparedStatement.executeUpdate();

Invoke dbConnection.commit();

//Getting multiple rows from the DB table

Assign ResultSet rs= preparedStatement.executeQuery();

Invoke dbConnection.commit();

Start while loop with argument (rs.next())

{

Output (rs.getString(1)+" "+rs.getString(2)) through System.out

}

Output ("Records selected from the DB table!") through System.out

}

```
//Handle errors for JDBC
Start catch loop with arguments (SQLException se)

{
Invoke se.printStackTrace();
 }

//Handle errors for Class.forName
Start catch loop with arguments (Exception e)
{
Invoke e.printStackTrace();
}
//finally block used to close resources
Invoke finally loop
{
Invoke try loop
{
Invoke if loop with arguments (preparedStatement!=null)
Close dbConnection
}
Invoke catch loop with arguments(SQLException se)
{ //do nothing
}
Invoke try loop
{
Invoke if loop with arguments (dbConnection!=null)
Close dbConnection
}
Invoke catch loop with arguments (SQLException se)
```

{

Invoke se.printStackTrace();

}

}

Output ("Class select execution complete") via System.out

}//end main

}

Update_Identity CLASS:

Import package epita.fr.fundamental.project with the package keyword

Import the following packages with the import keyword:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Scanner;

Initialize class Update_Identitiy of type public

{

Initialize function main() of type public static void with arguments String[] args

{

//Declaring DB Connection constants

Initialize String DB_DRIVER of type final

Assign String DB_DRIVER = "org.apache.derby.jdbc.EmbeddedDriver";

Initialize String DB_CONNECTION of type final

Assign DB_CONNECTION = "jdbc:derby:MyDB;create=true";

Initialize String DB_USER of type final

Assign DB_USER="shreya";

Initialize String DB_PASSWORD of type final

Assign DB_PASSWORD = "amit05";

Declaring identity table fields

Initialize String uid;

Initialize Connection dbConnection = null;

Initialize PreparedStatement preparedStatement = null;

//Taking input from user

Output ("Enter display name :”) through System.out.println

Take user input through variable a of type Scanner

Assign a=displayName

Output ("Enter display name :”) through System.out.println

Take user input through variable b of type Scanner

Assign b =uid.

Output ("Enter display name :”) through System.out.println

Take user input through variable c of type Scanner

Assign c=email.

Close a

Close b

Close c

Initialize Connection dbConnection = null;

Initialize PreparedStatement preparedStatement = null;

//DB update

Assign String updateTableSQL = "UPDATE IDENTITIES SET displayName = ? AND email = ?" + " WHERE uid = ?";

Initalize try loop {

//Register JDBC driver

Invoke Class.forName(DB_DRIVER);

//Open a connection

Output ("Connecting to a selected database...") through System.out

Assign dbConnection = DriverManager.getConnection(DB_CONNECTION, DB_USER, DB_PASSWORD);

Output ("Connected database successfully...") through System.out

Assign preparedStatement = dbConnection.prepareStatement(updateTableSQL);

Invoke preparedStatement.setString(1,displayName);

Invoke preparedStatement.setString(2,uid);

Invoke preparedStatement.setString(3,email);

//Execute update SQL statement

Invoke preparedStatement.executeUpdate();

Invoke dbConnection.commit();

Output ("Record is updated in the DB table!") through System.out

}

Initiate catch loop with arguments (SQLException se)

{

//Handle errors for JDBC

Invoke se.printStackTrace();

}

Initiate catch loop with argument(Exception e)

{

//Handle errors for Class.forName

Invoke e.printStackTrace();

}

Initiate finally loop

{

//finally block used to close resources

Initiate try loop

{

Initiate if loop with argument (preparedStatement!=null)

Close dbConnection

}

Initiate catch loop with argument (SQLException se){

// do nothing

}

Initiate try loop

{

Initate if loop with argument(dbConnection!=null)

Close dbConnection

}

Initiate catch loop with argument (SQLException se)

{

Invoke se.printStackTrace();

}//end finally try

}//end try

Output ("Class Update execution complete") through System.out

}//end main

}

Delete_Identity Class

Import package epita.fr.fundamental.project with the package keyword

Import the following packages with the import keyword:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Scanner;

Initialize class Delete_Identity of type public {

public static void main(String[] args) {

//Declaring DB Connection constants

Initialize String DB_DRIVER of type final

Assign String DB_DRIVER = "org.apache.derby.jdbc.EmbeddedDriver";

Initialize String DB_CONNECTION of type final

Assign DB_CONNECTION = "jdbc:derby:MyDB;create=true";

Initialize String DB_USER of type final

Assign DB_USER="shreya";

Initialize String DB_PASSWORD of type final

Assign DB_PASSWORD = "amit05";

//Declaring identity table fields

I

nitialize String uid;

Initialize Connection dbConnection = null;

Initialize PreparedStatement preparedStatement = null;

//Taking input from user

Output ("Enter user id :") through System.out.println

Take user input through variable a of type Scanner

Assign a=uid

Close a

```
Assign Connection dbConnection = null;
Assign PreparedStatement preparedStatement = null;

//DB delete
Assign String deleteTableSQL = "DELETE IDENTITIES WHERE uid = ?";
 Initiate try loop{
 //Register JDBC driver

Invoke Class.forName(DB_DRIVER);

//Open a connection
Output ("Connecting to a selected database...") through System.out
Assign dbConnection = DriverManager.getConnection(DB_CONNECTION,
DB_USER, DB_PASSWORD);
Output("Connected database successfully...") through System.out

 //Creating the DB Table
Assign preparedStatement = dbConnection.prepareStatement(deleteTableSQL);


  //Execute delete SQL statement
Invoke preparedStatement.executeUpdate();
Invoke dbConnection.commit();
Output ("Record is deleted in the DB table!") through System.out
}
Initiate catch loop with argument (SQLException se)
{
//Handle errors for JDBC
Invoke se.printStackTrace();
```

```
 }
Initiate catch loop with argument(Exception e)
{
//Handle errors for Class.forName
Invoke e.printStackTrace();
}

Initiate finally loop
{
//finally block used to close resources
Initiate try loop
{
Initiate if loop with argument (preparedStatement!=null)
Close dbConnection
}

Initiate catch loop with argument (SQLException se){
// do nothing
}
Initiate try loop
{
Initate if loop with argument(dbConnection!=null)
Close dbConnection
}
Initiate catch loop with argument (SQLException se)
{
Invoke se.printStackTrace();

}//end finally try
```

```
  }//end try
Output ("Class Update execution complete") through System.out

}//end main



}
```