/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Model class to hold MySQL employee data.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **class** Employee {

**private** Integer employeeId;

**private** String firstName;

**private** String lastName;

**private** Integer jobCode;

**private** Integer payCode;

**private** String salary;

**private** String email;

**private** String telephone;

// Constructor.

**public** Employee(

Integer id,

String fName,

String lName,

Integer jCode,

Integer pCode,

Double pay,

String mail,

String tele

) {

employeeId = id;

firstName = fName;

lastName = lName;

jobCode = jCode;

payCode = pCode;

email = mail;

telephone = tele;

salary = String.*format*( "$%.2f", pay );

}

// Getters.

**public** Integer getEmployeeId() { **return** employeeId; }

**public** String getFirstName() { **return** firstName; }

**public** String getLastName() { **return** lastName; }

**public** Integer getJobCode() { **return** jobCode; }

**public** Integer getPayCode() { **return** payCode; }

**public** String getSalary() { **return** salary; }

**public** String getTelephone() { **return** telephone; }

**public** String getEmail() { **return** email.isEmpty() ? "No email address provided." : email; }

// Setters.

**public** **void** setEmployeeId( Integer id ) { employeeId = id; }

**public** **void** setFirstName( String name ) { firstName = name; }

**public** **void** setLastName( String name ) { lastName = name; }

**public** **void** setJobCode( Integer code ) { jobCode = code; }

**public** **void** setPayCode( Integer code ) { payCode = code; }

**public** **void** setTelephone( String tele ) { telephone = tele; }

**public** **void** setEmail( String mail ) { email = mail; }

// Non-class member getters.

**public** **final** String getPayFrequencyDescription() {

**return** EmployeeTableViewUtility.*getPayFrequency*( payCode );

}

**public** **final** String getJobDescription() {

**return** EmployeeTableViewUtility.*getJobDescription*( jobCode );

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Title: Employee Table View Utility

\* File: EmployeeTableViewUtility.java

\* Author: James Eli

\* Date: 2/23/2017

\*

\* This JavaFX class provides utility functions for viewing a table of

\* employee data held in a MySQL database.

\*

\* Notes:

\* (1) Compiled with java SE JDK 8, Update 121 (JDK 8u121) and JavaFX

\* version 8.0.121-b13.

\*

\* Submitted in partial fulfillment of the requirements of PCC CIS-279.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Change Log:

\* 02/23/2017: Initial release. JME

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.control.TableColumn;

import javafx.scene.control.cell.PropertyValueFactory;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class EmployeeTableViewUtility {

private static List<String> jobDescriptions = new ArrayList<String>();

private static List<String> payFrequencies = new ArrayList<String>();

private static Connection connection;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Open MySQL database.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static boolean openDB( String url, String userId, String password ) {

// Open mysql database connection.

try {

//Class.forName( "com.mysql.jdbc.Driver" ); // Not required.

connection = DriverManager.getConnection( url, userId, password );

return true;

} catch ( Exception ex ) {

return false;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve a list of MySQL Employee database department fields.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static List<String> getDepartments() {

List<String> departments = new ArrayList<>();

try ( CallableStatement statement = connection.prepareCall( "{ call sp\_all\_departments() }" ) ) {

ResultSet resultSet = statement.executeQuery();

while ( resultSet.next() )

departments.add( resultSet.getString( "department\_name" ) );

} catch ( SQLException ex ) {

System.err.println( "Department exception" );

System.err.println( ex.getMessage() );

}

return departments;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve MySQL Employee database job type fields.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void getJobTypes() {

try ( Statement statement = connection.createStatement() ) {

// execute our query, and get a java resultset

ResultSet resultSet = statement.executeQuery( "SELECT \* FROM job\_type" );

while ( resultSet.next() )

jobDescriptions.add( resultSet.getString( "Job\_type\_description" ) );

} catch ( SQLException ex ) {

System.err.println( "Job type exception" );

System.err.println( ex.getMessage() );

}

}

public static String getJobDescription( int jCode ) {

return jobDescriptions.get( jCode );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve MySQL Employee database pay frequency fields.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void getPayFrequencies() {

try ( Statement statement = connection.createStatement() ) {

// execute our query, and get a java resultset

ResultSet resultSet = statement.executeQuery( "SELECT \* FROM pay\_frequency" );

while ( resultSet.next() )

payFrequencies.add( resultSet.getString( "pay\_freq\_description" ) );

} catch ( SQLException ex ) {

System.err.println( "Pay frequency exception" );

System.err.println( ex.getMessage() );

}

}

public static String getPayFrequency( int pCode ) {

return payFrequencies.get( pCode );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve MySQL Employee fields and insert into collection

\* of (observableList) of the Employee class.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static ObservableList<Employee> EmployeeGetList( int departmentNumber ) {

ObservableList<Employee> employeeList = FXCollections.<Employee>observableArrayList();

try ( CallableStatement statement = connection.prepareCall( "{ call sp\_employees\_in\_dept( ? ) }" ) ) {

statement.setString( 1, String.valueOf( departmentNumber ) ); // Insert department number to fetch.

ResultSet resultSet = statement.executeQuery();

while ( resultSet.next() ) {

int id = resultSet.getInt( "employee\_id" );

String lName = resultSet.getString( "last\_name" );

String fName = resultSet.getString( "first\_name" );

int jCode = resultSet.getInt( "job\_type\_code" );

String email = resultSet.getString( "email\_address" );

String tele = resultSet.getString( "telephone" );

double pay = resultSet.getInt( "pay" );

int pCode = resultSet.getInt( "pay\_freq\_code" );

employeeList.add( new Employee( id, fName, lName, jCode, pCode, pay, email, tele ) );

}

} catch ( SQLException ex ) {

System.err.println( "SQL exception" );

System.err.println( ex.getMessage() );

}

return employeeList;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Callbacks for the tableview columns.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static TableColumn<Employee, Integer> getIdColumn() {

TableColumn<Employee, Integer> idCol = new TableColumn<>( "Id" );

idCol.setPrefWidth( 50 );

idCol.setStyle( "-fx-alignment: CENTER;" );

idCol.setCellValueFactory( new PropertyValueFactory<>( "EmployeeId" ) );

return idCol;

}

public static TableColumn<Employee, String> getFirstNameColumn() {

TableColumn<Employee, String> fNameCol = new TableColumn<>( "First Name" );

fNameCol.setPrefWidth( 100 );

fNameCol.setSortable( false );

fNameCol.setCellValueFactory( new PropertyValueFactory<>( "firstName" ) );

return fNameCol;

}

public static TableColumn<Employee, String> getLastNameColumn() {

TableColumn<Employee, String> lNameCol = new TableColumn<>( "Last Name" );

lNameCol.setPrefWidth( 100 );

lNameCol.setCellValueFactory( new PropertyValueFactory<>("lastName" ) );

return lNameCol;

}

// Nested columns.

public static TableColumn<Employee, ?> getJobTypeColumn() {

TableColumn<Employee, ?> jobTypeCol = new TableColumn<>( "Job Type" );

TableColumn<Employee, Integer> jobTypeCodeCol = new TableColumn<>( "Code" );

TableColumn<Employee, String> jobTypeCodeDescriptionCol = new TableColumn<>( "Description" );

//@SuppressWarnings( "unchecked" ) jobTypeCol.getColumns().addAll( jobTypeCodeCol, jobTypeCodeDescriptionCol );

jobTypeCol.getColumns().add( jobTypeCodeCol );

jobTypeCol.getColumns().add( jobTypeCodeDescriptionCol );

jobTypeCodeCol.setPrefWidth( 50 );

jobTypeCodeCol.setSortable( false );

jobTypeCodeCol.setStyle( "-fx-alignment: CENTER;" );

jobTypeCodeCol.setCellValueFactory( new PropertyValueFactory<>( "jobCode" ) );

jobTypeCodeDescriptionCol.setPrefWidth( 100 );

jobTypeCodeDescriptionCol.setSortable( false );

jobTypeCodeDescriptionCol.setCellValueFactory( new PropertyValueFactory<>( "jobDescription" ) );

return jobTypeCol;

}

// Nested columns.

public static TableColumn<Employee, ?> getPayFrequencyColumn() {

TableColumn<Employee, ?> payFrequencyCol = new TableColumn<>( "Pay Frequency" );

TableColumn<Employee, Integer> payFrequencyCodeCol = new TableColumn<>( "Code" );

TableColumn<Employee, String> payFrequencyDescriptionCol = new TableColumn<>( "Description" );

//@SuppressWarnings( "unchecked" ) payFrequencyCol.getColumns().addAll( payFrequencyCodeCol, payFrequencyDescriptionCol );

payFrequencyCol.getColumns().add( payFrequencyCodeCol );

payFrequencyCol.getColumns().add( payFrequencyDescriptionCol );

payFrequencyCodeCol.setPrefWidth( 50 );

payFrequencyCodeCol.setStyle( "-fx-alignment: CENTER;" );

payFrequencyCodeCol.setSortable( false );

payFrequencyCodeCol.setCellValueFactory( new PropertyValueFactory<>( "payCode" ) );

payFrequencyDescriptionCol.setPrefWidth( 200 );

payFrequencyDescriptionCol.setSortable( false );

payFrequencyDescriptionCol.setCellValueFactory( new PropertyValueFactory<>( "payFrequencyDescription" ) );

return payFrequencyCol;

}

public static TableColumn<Employee, String> getPayColumn() {

TableColumn<Employee, String> payCol = new TableColumn<>( "Pay" );

payCol.setPrefWidth( 100 );

payCol.setStyle( "-fx-alignment: CENTER-RIGHT;" );

payCol.setCellValueFactory( new PropertyValueFactory<>( "salary" ) );

return payCol;

}

public static TableColumn<Employee, String> getEmailColumn() {

TableColumn<Employee, String> emailCol = new TableColumn<>( "Email" );

emailCol.setPrefWidth( 200 );

emailCol.setSortable( false );

emailCol.setCellValueFactory( new PropertyValueFactory<>( "email" ) );

return emailCol;

}

public static TableColumn<Employee, String> getTelephoneColumn() {

TableColumn<Employee, String> telephoneCol = new TableColumn<>( "Telephone" );

telephoneCol.setPrefWidth( 100 );

telephoneCol.setSortable( false );

telephoneCol.setCellValueFactory( new PropertyValueFactory<>( "telephone" ) );

return telephoneCol;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Title: Employee Table View Utility

\* File: EmployeeTableViewUtility.java

\* Author: James Eli

\* Date: 2/23/2017

\*

\* This JavaFX class provides utility functions for viewing a table of

\* employee data held in a MySQL database.

\*

\* Notes:

\* (1) Compiled with java SE JDK 8, Update 121 (JDK 8u121) and JavaFX

\* version 8.0.121-b13.

\*

\* Submitted in partial fulfillment of the requirements of PCC CIS-279.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Change Log:

\* 02/23/2017: Initial release. JME

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.control.TableColumn;

import javafx.scene.control.cell.PropertyValueFactory;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class EmployeeTableViewUtility {

private static List<String> jobDescriptions = new ArrayList<String>();

private static List<String> payFrequencies = new ArrayList<String>();

private static Connection connection;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Open MySQL database.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static boolean openDB( String url, String userId, String password ) {

// Open mysql database connection.

try {

//Class.forName( "com.mysql.jdbc.Driver" ); // Not required.

connection = DriverManager.getConnection( url, userId, password );

return true;

} catch ( Exception ex ) {

return false;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve a list of MySQL Employee database department fields.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static List<String> getDepartments() {

List<String> departments = new ArrayList<>();

try ( CallableStatement statement = connection.prepareCall( "{ call sp\_all\_departments() }" ) ) {

ResultSet resultSet = statement.executeQuery();

while ( resultSet.next() )

departments.add( resultSet.getString( "department\_name" ) );

} catch ( SQLException ex ) {

System.err.println( "Department exception" );

System.err.println( ex.getMessage() );

}

return departments;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve MySQL Employee database job type fields.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void getJobTypes() {

try ( Statement statement = connection.createStatement() ) {

// execute our query, and get a java resultset

ResultSet resultSet = statement.executeQuery( "SELECT \* FROM job\_type" );

while ( resultSet.next() )

jobDescriptions.add( resultSet.getString( "Job\_type\_description" ) );

} catch ( SQLException ex ) {

System.err.println( "Job type exception" );

System.err.println( ex.getMessage() );

}

}

public static String getJobDescription( int jCode ) {

return jobDescriptions.get( jCode );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve MySQL Employee database pay frequency fields.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void getPayFrequencies() {

try ( Statement statement = connection.createStatement() ) {

// execute our query, and get a java resultset

ResultSet resultSet = statement.executeQuery( "SELECT \* FROM pay\_frequency" );

while ( resultSet.next() )

payFrequencies.add( resultSet.getString( "pay\_freq\_description" ) );

} catch ( SQLException ex ) {

System.err.println( "Pay frequency exception" );

System.err.println( ex.getMessage() );

}

}

public static String getPayFrequency( int pCode ) {

return payFrequencies.get( pCode );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Retrieve MySQL Employee fields and insert into collection

\* of (observableList) of the Employee class.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static ObservableList<Employee> EmployeeGetList( int departmentNumber ) {

ObservableList<Employee> employeeList = FXCollections.<Employee>observableArrayList();

try ( CallableStatement statement = connection.prepareCall( "{ call sp\_employees\_in\_dept( ? ) }" ) ) {

statement.setString( 1, String.valueOf( departmentNumber ) ); // Insert department number to fetch.

ResultSet resultSet = statement.executeQuery();

while ( resultSet.next() ) {

int id = resultSet.getInt( "employee\_id" );

String lName = resultSet.getString( "last\_name" );

String fName = resultSet.getString( "first\_name" );

int jCode = resultSet.getInt( "job\_type\_code" );

String email = resultSet.getString( "email\_address" );

String tele = resultSet.getString( "telephone" );

double pay = resultSet.getInt( "pay" );

int pCode = resultSet.getInt( "pay\_freq\_code" );

employeeList.add( new Employee( id, fName, lName, jCode, pCode, pay, email, tele ) );

}

} catch ( SQLException ex ) {

System.err.println( "SQL exception" );

System.err.println( ex.getMessage() );

}

return employeeList;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Callbacks for the tableview columns.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static TableColumn<Employee, Integer> getIdColumn() {

TableColumn<Employee, Integer> idCol = new TableColumn<>( "Id" );

idCol.setPrefWidth( 50 );

idCol.setStyle( "-fx-alignment: CENTER;" );

idCol.setCellValueFactory( new PropertyValueFactory<>( "EmployeeId" ) );

return idCol;

}

public static TableColumn<Employee, String> getFirstNameColumn() {

TableColumn<Employee, String> fNameCol = new TableColumn<>( "First Name" );

fNameCol.setPrefWidth( 100 );

fNameCol.setSortable( false );

fNameCol.setCellValueFactory( new PropertyValueFactory<>( "firstName" ) );

return fNameCol;

}

public static TableColumn<Employee, String> getLastNameColumn() {

TableColumn<Employee, String> lNameCol = new TableColumn<>( "Last Name" );

lNameCol.setPrefWidth( 100 );

lNameCol.setCellValueFactory( new PropertyValueFactory<>("lastName" ) );

return lNameCol;

}

// Nested columns.

public static TableColumn<Employee, ?> getJobTypeColumn() {

TableColumn<Employee, ?> jobTypeCol = new TableColumn<>( "Job Type" );

TableColumn<Employee, Integer> jobTypeCodeCol = new TableColumn<>( "Code" );

TableColumn<Employee, String> jobTypeCodeDescriptionCol = new TableColumn<>( "Description" );

//@SuppressWarnings( "unchecked" ) jobTypeCol.getColumns().addAll( jobTypeCodeCol, jobTypeCodeDescriptionCol );

jobTypeCol.getColumns().add( jobTypeCodeCol );

jobTypeCol.getColumns().add( jobTypeCodeDescriptionCol );

jobTypeCodeCol.setPrefWidth( 50 );

jobTypeCodeCol.setSortable( false );

jobTypeCodeCol.setStyle( "-fx-alignment: CENTER;" );

jobTypeCodeCol.setCellValueFactory( new PropertyValueFactory<>( "jobCode" ) );

jobTypeCodeDescriptionCol.setPrefWidth( 100 );

jobTypeCodeDescriptionCol.setSortable( false );

jobTypeCodeDescriptionCol.setCellValueFactory( new PropertyValueFactory<>( "jobDescription" ) );

return jobTypeCol;

}

// Nested columns.

public static TableColumn<Employee, ?> getPayFrequencyColumn() {

TableColumn<Employee, ?> payFrequencyCol = new TableColumn<>( "Pay Frequency" );

TableColumn<Employee, Integer> payFrequencyCodeCol = new TableColumn<>( "Code" );

TableColumn<Employee, String> payFrequencyDescriptionCol = new TableColumn<>( "Description" );

//@SuppressWarnings( "unchecked" ) payFrequencyCol.getColumns().addAll( payFrequencyCodeCol, payFrequencyDescriptionCol );

payFrequencyCol.getColumns().add( payFrequencyCodeCol );

payFrequencyCol.getColumns().add( payFrequencyDescriptionCol );

payFrequencyCodeCol.setPrefWidth( 50 );

payFrequencyCodeCol.setStyle( "-fx-alignment: CENTER;" );

payFrequencyCodeCol.setSortable( false );

payFrequencyCodeCol.setCellValueFactory( new PropertyValueFactory<>( "payCode" ) );

payFrequencyDescriptionCol.setPrefWidth( 200 );

payFrequencyDescriptionCol.setSortable( false );

payFrequencyDescriptionCol.setCellValueFactory( new PropertyValueFactory<>( "payFrequencyDescription" ) );

return payFrequencyCol;

}

public static TableColumn<Employee, String> getPayColumn() {

TableColumn<Employee, String> payCol = new TableColumn<>( "Pay" );

payCol.setPrefWidth( 100 );

payCol.setStyle( "-fx-alignment: CENTER-RIGHT;" );

payCol.setCellValueFactory( new PropertyValueFactory<>( "salary" ) );

return payCol;

}

public static TableColumn<Employee, String> getEmailColumn() {

TableColumn<Employee, String> emailCol = new TableColumn<>( "Email" );

emailCol.setPrefWidth( 200 );

emailCol.setSortable( false );

emailCol.setCellValueFactory( new PropertyValueFactory<>( "email" ) );

return emailCol;

}

public static TableColumn<Employee, String> getTelephoneColumn() {

TableColumn<Employee, String> telephoneCol = new TableColumn<>( "Telephone" );

telephoneCol.setPrefWidth( 100 );

telephoneCol.setSortable( false );

telephoneCol.setCellValueFactory( new PropertyValueFactory<>( "telephone" ) );

return telephoneCol;

}

}