Justin Pappano

ITMD 510

4/29/2018

ITMD 510

PROJECT PHASE – II

DESCRIPTION OF THE PROJECT

The purpose of this project is to help either businesses or individuals track large amounts of cash flow. This application will process two types of cash flow payments:

* Income
* Expenses

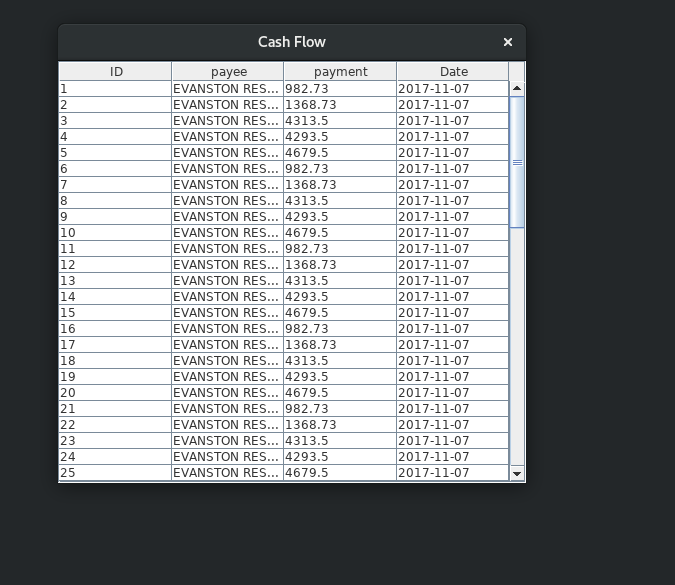
Both types of cash flow are processed the same way but are segregated. Each respective payment type is places into an sqlite database of its given type. The program will process each type of cash flow by effectively identifying:

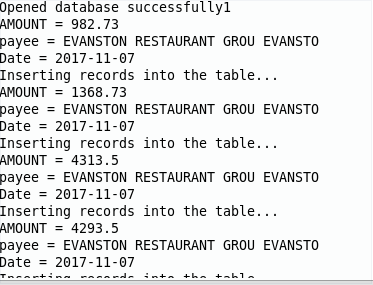
* top 5 payments made (received and expended)
* top 5 Dates where lasrgest transaction sizes have occurred.
* top 5 payees and payers

The app takes information from an sqlite database, skrooge\_sqlite.db, and puts them into James Papadema’s database server. The information in the sqlite database were taken from various sources online. The table, CashFlow, will ultimately contain transactions that are of interest. The goal is for the user to identify simple spending trends for a business for purposes of security and budgeting/

In order to operate this, one must run it in eclipse from the Main.java class.







**package** application;

**import** java.sql.SQLException;

**public** **interface** CashFlow

{

**abstract** **void** amountSize();//fomrs a tally of the top 5 largest expenses

**abstract** **void** topDate() **throws** SQLException; //forms a tally of the top 5 most used categories

**abstract** **void** topPayee(); //forms a tally of the top 5 Payees

}

package application;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;

import com.mysql.jdbc.Statement;

import java.sql.\*;

import models.DaoModelLite;

/\*\*

\* <h1>Expense any negative income comes through here</h1>

\* @author Justin Pappano

\* @lab Final Project

\* @class ITMD510

\*/

public class Expense implements CashFlow

{

static ArrayList<List<String>> array = new ArrayList<>();

static InputStreamReader ir = new InputStreamReader(System.in);

static float amt=0;

// static String qy = "SELECT f\_balance, payee.t\_name, operation.d\_date FROM operationbalance, operation, payee, "

// + "suboperation WHERE f\_balance = f\_balance\_entered AND f\_balance <0 "

// + "AND operation.r\_payee\_id = payee.id AND operation.d\_date=suboperation.d\_date "

// + "AND suboperation.r\_category\_id = payee.r\_category\_id ORDER BY f\_balance ASC;";

//

static int i = 0;

static Connection c = null;

Statement stmt;

DaoModelLite conn;

String qy;

/\*\*

\* This constructor selects every record in the database that is considered to be an expense

\* So, every amount that is negative

\* @throws SQLException

\*/

public Expense() throws SQLException

{

conn= new DaoModelLite();

conn.connect();

amountSize();

}

@Override

//this will count the top largest transactions

public void amountSize()

{

qy = " SELECT f\_balance, payee.t\_name, operation.d\_date FROM operationbalance, operation, "

+ "payee, suboperation WHERE f\_balance = f\_balance\_entered AND f\_balance <0 AND "

+ "operation.r\_payee\_id = payee.id AND operation.d\_date=suboperation.d\_date AND "

+ "suboperation.r\_category\_id = payee.r\_category\_id LIMIT 5; ";

try

{

conn.Selecting(qy);

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

//these are the top 5 dates where transactions have occured

public void topDate()

{

qy = "SELECT f\_balance, t\_name, operation.d\_date, count(operation.d\_date) AS OCCURANCE "

+ "FROM operationbalance, operation, payee, suboperation WHERE f\_balance = f\_balance\_entered "

+ "AND f\_balance <0 AND operation.r\_payee\_id = payee.id AND operation.d\_date=suboperation.d\_date "

+ "AND suboperation.r\_category\_id = payee.r\_category\_id GROUP BY operation.d\_date ORDER "

+ "BY OCCURANCE DESC limit 5;";

try {

conn.Selecting(qy);

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

topPayee();

}

//rank the top 5 payees in order from largest transaction

public void topPayee()

{

qy = "SELECT f\_balance, t\_name, COunt(t\_name) AS OCCURANCE, operation.d\_date FROM operationbalance, operation, "

+ "payee, suboperation WHERE f\_balance = f\_balance\_entered AND f\_balance <0 AND operation.r\_payee\_id = payee.id "

+ "AND operation.d\_date=suboperation.d\_date AND suboperation.r\_category\_id = payee.r\_category\_id ORDER BY payee.t\_name "

+ " DESC limit 5";

try

{

conn.Selecting(qy);

}

catch (SQLException e1)

{

// TODO Auto-generated catch block

e1.printStackTrace();

}

/\*try

{

conn.close();

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}\*/

}

}

package models;

import java.sql.Connection;

import java.sql.Date;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Time;

import java.sql.Timestamp;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

public class DaoModelLite

{

static Connection c = null;

static float payment;

static String payee;

static float payments;

static String payees;

static String df;

//static DaoModel D = new DaoModel();

public static void connect() throws SQLException

{

// Connection c = null;

String url = "jdbc:sqlite:skrooge\_sqlite.db";

try

{

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection(url);

c.setAutoCommit(false);

}

catch ( Exception e )

{

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

System.out.println("Opened database successfully1");

}

public void Selecting(String qy) throws SQLException

{

Statement stmt = null;

try

{

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery(qy);

while ( rs.next() )

{

payment = rs.getFloat("f\_balance");

payee = rs.getString("t\_name");

df = rs.getString("d\_date");

System.out.println( "AMOUNT = " + payment);

System.out.println( "payee = " + payee );

System.out.println( "Date = " + df );

//DaoModel.insertRecords(payment, payee );

DaoModel.insertRecords(payment, payee, df);

}

rs.close();

stmt.close();

c.close();

}

catch ( Exception e )

{

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

System.out.println("Operation done successfully3");

}

}

package application;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;

import com.mysql.jdbc.Statement;

import java.sql.\*;

import models.DaoModelLite;

public class Income implements CashFlow

{

/\*static String qy = "SELECT f\_balance, payee.t\_name, operation.d\_date FROM operationbalance, "

+ "operation, payee, suboperation WHERE f\_balance > 0 AND f\_balance = f\_balance\_entered "

+ "AND f\_balance AND operation.r\_payee\_id = payee.id AND operation.d\_date=suboperation.d\_date A"

+ "ND suboperation.r\_category\_id = payee.r\_category\_id limit 5;";

\*/

static int i = 0;

static Connection c = null;

Statement stmt;

Expense era [] = new Expense[5];

DaoModelLite conn;

static String qy;

public Income() throws SQLException

{

conn= new DaoModelLite();

conn.connect();

amountSize();

}

@Override

public void amountSize()

{

qy = "SELECT f\_balance, payee.t\_name, operation.d\_date FROM operationbalance, operation, payee, "

+ "suboperation WHERE f\_balance > 0 AND f\_balance = f\_balance\_entered AND f\_balance AND"

+ " operation.r\_payee\_id = payee.id AND operation.d\_date=suboperation.d\_date AND "

+ "suboperation.r\_category\_id = payee.r\_category\_id limit 5;";

//topPayee() ;

try

{

conn.Selecting(qy);

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public void topDate()

{

qy = "SELECT f\_balance, t\_name, count(countoperation.d\_date) AS OCCURANCE "

+ "FROM operationbalance, operation, payee, suboperation WHERE f\_balance = f\_balance\_entered "

+ "AND f\_balance >0 AND operation.r\_payee\_id = payee.id AND operation.d\_date=suboperation.d\_date "

+ "AND suboperation.r\_category\_id = payee.r\_category\_id limit 5;";

//topPayee();

try

{

conn.Selecting(qy);

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public void topPayee()

{

qy = "SELECT f\_balance, t\_name, COunt(t\_name) AS OCCURANCE, operation.d\_date FROM operationbalance, operation, "

+ "payee, suboperation WHERE f\_balance = f\_balance\_entered AND f\_balance >0 AND operation.r\_payee\_id = payee.id "

+ "AND operation.d\_date=suboperation.d\_date AND suboperation.r\_category\_id = payee.r\_category\_id DESC limit 5";

try

{

conn.Selecting(qy);

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

/\*try

{

conn.close();

}

catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}\*/

}

}

package views;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.util.HashMap;

import java.util.Map;

import java.util.Vector;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

public class CashView

{

public void runView(ResultSet rs)

{

// this vector makes columns for both the table as well

//as a possible serialization process

Vector<Vector<Object>> data = new Vector<Vector<Object>>();

Vector<String> column = new Vector<String>();

try {

ResultSetMetaData metaData = rs.getMetaData();

int columns = metaData.getColumnCount();

// get column names from table!

String cols = "";

for (int i = 1; i <= columns; i++) {

cols = metaData.getColumnName(i);

column.add(cols);

}

while (rs.next())

{

Vector<Object> row = new Vector<Object>(columns);

for (int i = 1; i <= columns; i++)

row.addElement(rs.getObject(i));

data.addElement(row);

}

DefaultTableModel model = new DefaultTableModel(data, column);

JTable table = new JTable(model);

JFrame frame = new JFrame("Cash Flow");

frame.setSize(700, 200);

frame.add(new JScrollPane(table));

frame.setDefaultCloseOperation(0);

frame.pack();

frame.setVisible(true);

rs.close(); //close ResultSet instance

}

catch (SQLException e)

{

e.printStackTrace();

}

}

}

**package** controllers;

**import** java.sql.SQLException;

**import** java.sql.ResultSet;

**import** application.Expense;

**import** application.Income;

**import** javafx.application.Application;

**import** javafx.scene.Scene;

**import** javafx.scene.layout.BorderPane;

**import** javafx.stage.Stage;

**import** models.DaoModel;

**import** models.DaoModelLite;

**import** views.CashView;

**public** **class** Main //extends Application

{

/\* @Override

public void start(Stage primaryStage)

{

try

{

BorderPane root = new BorderPane();

Scene scene = new Scene(root,400,400);

scene.getStylesheets().add(getClass().

getResource("application.css").toExternalForm());

primaryStage.setScene(scene);

primaryStage.show();

}

catch(Exception e)

{

e.printStackTrace();

}

} \*/

**public** **static** **void** main(String[] args) **throws** SQLException

{

Income zz = **new** Income();

Expense tty = **new** Expense();

DaoModel dao = **new** DaoModel();

dao.createTable();

ResultSet rs;

**try**

{

rs = DaoModel.*retrieveRecords*();

**new** CashView().runView(rs);

}

**catch**(SQLException e)

{

e.printStackTrace();

}

//launch(args);

}

}

package models;

import java.sql.\*;

import application.Expense;

public class DBConnect

{

// Papdemas code database URL

static final String DB\_URL = "jdbc:mysql://www.papademas.net:3306/fp510?autoReconnect=true&useSSL=false";

// Papdemas Database credentials

static final String USER = "fpuser", PASS = "510";

public static Connection connect() throws SQLException

{

return DriverManager.getConnection(DB\_URL, USER, PASS);

}

}

package models;

import java.sql.Connection;

import java.sql.Date;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Time;

import java.sql.Timestamp;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

public class DaoModelLite

{

static Connection c = null;

static float payment;

static String payee;

static float payments;

static String payees;

static String df;

//static DaoModel D = new DaoModel();

public static void connect() throws SQLException

{

// Connection c = null;

String url = "jdbc:sqlite:skrooge\_sqlite.db";

try

{

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection(url);

c.setAutoCommit(false);

}

catch ( Exception e )

{

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

System.out.println("Opened database successfully1");

}

public void Selecting(String qy) throws SQLException

{

Statement stmt = null;

try

{

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery(qy);

while ( rs.next() )

{

payment = rs.getFloat("f\_balance");

payee = rs.getString("t\_name");

df = rs.getString("d\_date");

System.out.println( "AMOUNT = " + payment);

System.out.println( "payee = " + payee );

System.out.println( "Date = " + df );

//DaoModel.insertRecords(payment, payee );

DaoModel.insertRecords(payment, payee, df);

}

rs.close();

stmt.close();

c.close();

}

catch ( Exception e )

{

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

System.exit(0);

}

System.out.println("Operation done successfully3");

}

}

package models;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.HashMap;

import java.util.Collection;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

//import com.sun.javafx.collections.MappingChange.Map;

public class DaoModel

{

//Declare DB objects

static DBConnect conn;

static Statement stmt;

//static ResultSet rs = null;

static int j= 1;

// constructor

final String FILE = "meta.ser";

// constructor

public DaoModel()

{

//create db object instance

conn = new DBConnect();

}

// CREATE TABLE METHOD

public void createTable()

{

try

{

//conn = DBConnect.connect();

// Open a connection

System.out.println("Connecting to a selected database to create Table...");

System.out.println("Connected database successfully...");

// Execute create query

System.out.println("Creating table in given database...");

stmt = conn.connect().createStatement();

String sql = "CREATE TABLE cashflows " +

"(ID INTEGER AUTO\_INCREMENT, " +

"payee TEXT ," +

"payment FLOAT , " +

"Date TEXT , " +

"PRIMARY KEY ( ID ) )";

stmt.executeUpdate(sql);

System.out.println("Created table in given database...");

conn.connect().close(); //close db connection

}

catch (SQLException se)

{

// Handle errors for JDBC

se.printStackTrace();

}

}

// INSERT INTO METHOD

public static void insertRecords(float f, String p, String d)

{

try

{

//conn = DBConnect.connect();

// Execute a query

System.out.println("Inserting records into the table...");

stmt = conn.connect().createStatement();

// Include all object data to the database table

// finish string assignment to insert all object data

// (pid, id, income, pep) into your database table

//ask papdemas if the "cashflow should be deleted

String sql = "INSERT INTO cashflows (payee, payment, Date) "

+ "VALUES('"+p+"','"+f+

"','"+d+"')";

stmt.executeUpdate(sql);

conn.connect().close();

}

catch(SQLException se)

{

se.printStackTrace();

}

}

public static ResultSet retrieveRecords() throws SQLException

{

ResultSet rs = null;

stmt = conn.connect().createStatement();

String sql = "SELECT \* FROM cashflows; ";

rs = stmt.executeQuery(sql);

//conn.connect().close();

return rs;

}

}

package views;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.util.HashMap;

import java.util.Map;

import java.util.Vector;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

public class CashView

{

public void runView(ResultSet rs)

{

// this vector makes columns for both the table as well

//as a possible serialization process

Vector<Vector<Object>> data = new Vector<Vector<Object>>();

Vector<String> column = new Vector<String>();

try {

ResultSetMetaData metaData = rs.getMetaData();

int columns = metaData.getColumnCount();

// get column names from table!

String cols = "";

for (int i = 1; i <= columns; i++) {

cols = metaData.getColumnName(i);

column.add(cols);

}

while (rs.next())

{

Vector<Object> row = new Vector<Object>(columns);

for (int i = 1; i <= columns; i++)

row.addElement(rs.getObject(i));

data.addElement(row);

}

DefaultTableModel model = new DefaultTableModel(data, column);

JTable table = new JTable(model);

JFrame frame = new JFrame("Cash Flow");

frame.setSize(700, 200);

frame.add(new JScrollPane(table));

frame.setDefaultCloseOperation(0);

frame.pack();

frame.setVisible(true);

rs.close(); //close ResultSet instance

}

catch (SQLException e)

{

e.printStackTrace();

}

}

}