package com.didi.model;  
  
import java.sql.\*;  
import java.util.ArrayList;  
  
public class Datasource {  
  
 public static final String *DB\_NAME* = "Buyers.db";  
 public static final String *CONNECTION\_STRING* = "jdbc:postgresql://localhost:5432/Buyers";  
  
 public static final String *TABLE\_BUYER* = "buyers";  
 public static final String *COLUMN\_BUYER\_ID* = "id";  
 public static final String *COLUMN\_BUYER\_NAME* = "buyerName";  
 public static final String *COLUMN\_BUYER\_VALUE* = "value";  
 public static final int *INDEX\_BUYER\_ID* = 1;  
 public static final int *INDEX\_BUYER\_NAME* = 2;  
 public static final int *INDEX\_BUYER\_VALUE* = 3;  
  
 public static final String *TABLE\_MAPTRANSACTIONS* = "maptransactions";  
 public static final String *COLUMN\_MAPTRANSACTIONS\_ID* = "id";  
 public static final String *COLUMN\_MAPTRANSACTIONS\_BIND* = "bind";  
 public static final int *INDEX\_MAPTRANSACTIONS\_ID* = 1;  
 public static final int *INDEX\_MAPTRANSACTIONS\_BIND* = 2;  
  
  
 public static final String *TABLE\_INDIVIDUALBUYER* = "individualbuyers";  
 public static final String *COLUMN\_INDIVIDUALBUYER\_DATEREGISTERED* = "dateRegistered";  
 public static final String *COLUMN\_INDIVIDUALBUYER\_PID* = "buyerPersonalId";  
 public static final String *COLUMN\_INDIVIDUALBUYER\_TRANSACTIONS* = "transactions";  
 public static final int *INDEX\_INDIVIDUALBUYER\_DATEREGISTERED* = 1;  
 public static final int *INDEX\_INDIVIDUALBUYER\_PID* = 2;  
 public static final int *INDEX\_INDIVIDUALBUYER\_TRANSACTIONS* = 3;  
  
 public static final String *TABLE\_CORPORATEBUYER* = "corporatebuyers";  
 public static final String *COLUMN\_CORPORATEBUYER\_ADDRESS* = "address";  
 public static final String *COLUMN\_CORPORATEBUYER\_CID* = "companyId";  
 public static final String *COLUMN\_CORPORATEBUYER\_TRANSACTIONS* = "transactions";  
 public static final int *INDEX\_CORPORATEBUYER\_ADDRESS* = 1;  
 public static final int *INDEX\_CORPORATEBUYER\_CID* = 2;  
 public static final int *INDEX\_CORPORATEBUYER\_TRANSACTIONS* = 3;  
  
 public static final String *TABLE\_TRANSACTIONS* = "transactions";  
 public static final String *COLUMN\_TRANSACTION\_ID* = "id";  
 public static final String *COLUMN\_TRANSACTION\_NUMBER* = "transactionNumber";  
 public static final String *COLUMN\_TRANSACTION\_VALUE* = "value";  
 public static final String *COLUMN\_TRANSACTION\_DESCRIPTION* = "description";  
 public static final int *INDEX\_TRANSACTION\_ID* = 1;  
 public static final int *INDEX\_TRANSACTION\_NUMBER* = 2;  
 public static final int *INDEX\_TRANSACTION\_VALUE* = 3;  
 public static final int *INDEX\_TRANSACTION\_DESCRIPTION* = 4;  
  
 public static final int *ORDER\_BY\_NONE* = 1;  
 public static final int *ORDER\_BY\_ASC* = 2;  
 public static final int *ORDER\_BY\_DESC* = 3;  
  
 public static final String *INSERT\_INDIVIDUALBUYER* = "INSERT INTO public." + *TABLE\_INDIVIDUALBUYER* +  
 '(' + *COLUMN\_BUYER\_ID* + ", \"" + *COLUMN\_BUYER\_NAME* + "\"," +  
 *COLUMN\_BUYER\_VALUE* + ", \"" + *COLUMN\_INDIVIDUALBUYER\_DATEREGISTERED* + "\", \"" +  
 *COLUMN\_INDIVIDUALBUYER\_PID* + "\", " + *COLUMN\_INDIVIDUALBUYER\_TRANSACTIONS* +  
 ") VALUES(uuid\_generate\_v4(), ?, ?, ?, ?, ?)";  
  
 public static final String *INSERT\_CORPORATEBUYER* = "INSERT INTO public." + *TABLE\_CORPORATEBUYER* +  
 "(\"" + *COLUMN\_BUYER\_ID* + "\", " + *COLUMN\_BUYER\_NAME* + ", " +  
 *COLUMN\_BUYER\_VALUE* + ", \"" + *COLUMN\_CORPORATEBUYER\_ADDRESS* + "\", \"" +  
 *COLUMN\_CORPORATEBUYER\_CID* + "\", " + *COLUMN\_CORPORATEBUYER\_TRANSACTIONS* +  
 ") VALUES(uuid\_generate\_v4(), ?, ? ,? ,?, ?)";  
  
 public static final String *INSERT\_MAPTRANSACTION* = "INSERT INTO public." + *TABLE\_MAPTRANSACTIONS* +  
 '(' + *COLUMN\_MAPTRANSACTIONS\_ID*+ ", " + *COLUMN\_MAPTRANSACTIONS\_BIND* +  
 ") VALUES(?, uuid\_generate\_v4())";  
  
 public static final String *INSERT\_TRANSACTION* = "INSERT INTO public." + *TABLE\_TRANSACTIONS* +  
 '(' + *COLUMN\_TRANSACTION\_ID* + ", \"" + *COLUMN\_TRANSACTION\_NUMBER* + "\", " +  
 *COLUMN\_TRANSACTION\_VALUE* + ", " + *COLUMN\_TRANSACTION\_DESCRIPTION* +  
 ") VALUES(uuid\_generate\_v4(), ?, ?, ?)";  
  
 public static final String *QUERY\_INDIVIDUALBUYERS* = "SELECT " + *COLUMN\_BUYER\_NAME* + " FROM public." +  
 *TABLE\_INDIVIDUALBUYER*;  
  
 public static final String *QUERY\_CORPORATEBUYERS* = "SELECT " + *COLUMN\_BUYER\_NAME* + " FROM public." +  
 *TABLE\_CORPORATEBUYER*;  
  
 public static final String *QUERY\_TRANSACTIONS* = "SELECT " + *COLUMN\_TRANSACTION\_VALUE* + " FROM public." +  
 *TABLE\_TRANSACTIONS* + " WHERE " + *COLUMN\_TRANSACTION\_ID* + " = \"";  
  
 public static final String *QUERY\_MAPTRANSACTIONS* = "SELECT " + *COLUMN\_MAPTRANSACTIONS\_BIND* + " FROM public." +  
 *TABLE\_MAPTRANSACTIONS* + " WHERE " + *COLUMN\_MAPTRANSACTIONS\_ID* + " = \"";  
  
  
  
 private Connection conn;  
  
 private PreparedStatement insertIntoIndividualBuyers;  
 private PreparedStatement insertIntoCorporateBuyers;  
 private PreparedStatement insertIntoTransactions;  
 private PreparedStatement insertIntoMapTransactions;  
  
 private PreparedStatement queryIndividualBuyers;  
 private PreparedStatement queryCorporateBuyers;  
 private PreparedStatement queryMapTransactions;  
  
  
 public boolean open() {  
 try {  
 // java.sql - class DriveManager  
 // public static Connection getConnection(String url, Properties info)  
 // throws SQLException  
 // ttempts to establish a connection to the given database URL. The DriverManager  
 // attempts to select an appropriate driver from the set of registered JDBC drivers.  
 // https://docs.oracle.com/javase/8/docs/api/java/sql/DriverManager.html#getConnection-java.  
 // lang.String-java.util.Properties-  
 conn = DriverManager.*getConnection*(*CONNECTION\_STRING*, "postgres", "sarchizian");  
  
  
 // interface Connection  
 // PrepareStatement prepareStatement(String sql, String[] columnNames)  
 // Creates a default PreparedStatement object capable of returning the auto-generated  
 // keys designated by the given array.  
 // Parameters:  
 // sql - an SQL statement that may contain one or more '?' IN parameter placeholders  
 // autoGeneratedKeys - a flag indicating whether auto-generated keys should be returned;  
 // one of Statement.RETURN\_GENERATED\_KEYS or Statement.NO\_GENERATED\_KEYS  
 // https://docs.oracle.com/javase/8/docs/api/java/sql/Connection.html  
  
 insertIntoIndividualBuyers = conn.prepareStatement(*INSERT\_INDIVIDUALBUYER*);  
 //, Statement.RETURN\_GENERATED\_KEYS);  
 insertIntoCorporateBuyers = conn.prepareStatement(*INSERT\_CORPORATEBUYER*);  
 //, Statement.RETURN\_GENERATED\_KEYS);  
 insertIntoTransactions = conn.prepareStatement(*INSERT\_TRANSACTION*);  
 //, Statement.RETURN\_GENERATED\_KEYS);  
 insertIntoMapTransactions = conn.prepareStatement(*INSERT\_MAPTRANSACTION*);  
 //, Statement.RETURN\_GENERATED\_KEYS);  
  
  
 queryIndividualBuyers = conn.prepareStatement(*QUERY\_INDIVIDUALBUYERS*);  
 queryCorporateBuyers = conn.prepareStatement(*QUERY\_CORPORATEBUYERS*);  
 queryMapTransactions = conn.prepareStatement(*QUERY\_MAPTRANSACTIONS*);  
  
  
 return true;  
 } catch (SQLException e) {  
 System.*out*.println("Couldn't connect to database: " + e.getMessage());  
 return false;  
 }  
 }  
  
 public void close() {  
 try {  
  
 if (insertIntoIndividualBuyers != null) {  
 insertIntoIndividualBuyers.close();  
 }  
  
 if (insertIntoCorporateBuyers != null) {  
 insertIntoCorporateBuyers.close();  
 }  
  
 if (insertIntoTransactions != null) {  
 insertIntoTransactions.close();  
 }  
  
 if (insertIntoMapTransactions != null) {  
 insertIntoMapTransactions.close();  
 }  
  
 if (queryIndividualBuyers != null) {  
 queryIndividualBuyers.close();  
 }  
  
 if (queryCorporateBuyers != null) {  
 queryCorporateBuyers.close();  
 }  
  
 if (queryMapTransactions != null) {  
 queryMapTransactions.close();  
 }  
  
 if (conn != null) {  
 conn.close();  
 }  
 } catch (SQLException e) {  
 System.*out*.println("Couldn't close connection: " + e.getMessage());  
 }  
 }  
  
 public void queryTransactionsMetadata() {  
 String sql = "SELECT \* FROM " + *TABLE\_TRANSACTIONS*;  
  
 try (Statement statement = conn.createStatement();  
 ResultSet results = statement.executeQuery(sql)) {  
  
 ResultSetMetaData meta = results.getMetaData();  
 int numColumns = meta.getColumnCount();  
 for (int i = 1; i <= numColumns; i++) {  
 System.*out*.format("Column %d in the transactions table is named %s\n",  
 i, meta.getColumnName(i));  
 }  
 } catch (SQLException e) {  
 System.*out*.println("Query failed: " + e.getMessage());  
 }  
 }  
  
 public long insertMapTransaction (long id) throws SQLException {  
 queryMapTransactions.setLong(1, id);  
 ResultSet results = queryMapTransactions.executeQuery();  
 if (results.next()) {  
 return results.getLong(1);  
 } else {  
 insertIntoMapTransactions.setLong(1, id);  
 }  
  
 int affectedRows = insertIntoIndividualBuyers.executeUpdate();  
 if (affectedRows != 1) {  
 throw new SQLException("Couldn't insert MapTransactions!");  
 }  
 ResultSet generatedKeys = insertIntoIndividualBuyers.getGeneratedKeys();  
 // the id from the new generated IB  
  
 if (generatedKeys.next()) {  
 return generatedKeys.getLong(1);  
 } else {  
 throw new SQLException("Couldn't get id for IndividualBuyer");  
 }  
 }  
  
  
 public String insertIndividualBuyer(String buyerName, long value,  
 Date dateRegistered, String buyerPersonalId,  
 long transactions) throws SQLException {  
  
 // java.sql Interface Statement  
 // ResultSet executeQuery(String sql)  
 // Executes the given SQL statement, which returns a single ResultSet object.  
 // https://docs.oracle.com/javase/8/docs/api/java/sql/Statement.html  
  
 //first we query the table to see if the IB exists  
 queryIndividualBuyers.setString(1, buyerName);  
  
 ResultSet results = queryIndividualBuyers.executeQuery();  
  
 if (results.next()) { // checks to see whether an id is returned  
 return results.getString(1); // if the IB exists it returns the id of the existing IB  
 } else {  
 // Insert the IndividualBuyer  
 insertIntoIndividualBuyers.setString(1, buyerName);  
 insertIntoIndividualBuyers.setLong(2, value);  
 insertIntoIndividualBuyers.setDate(3, dateRegistered);  
 insertIntoIndividualBuyers.setString(4, buyerPersonalId);  
 insertIntoIndividualBuyers.setLong(5, transactions);  
  
 }  
  
 // we're not using execute since execute returns a boolean indicating  
 // what type of result the executing sql statement returns  
 // executeUpdate - returns the number of rows affected by the sql code that ran  
 int affectedRows = insertIntoIndividualBuyers.executeUpdate(); // saves the object and tells  
  
 //since we're inserting a single row we actually expect only one record to be returned  
 if (affectedRows != 1) {  
 throw new SQLException("Couldn't insert IndividualBuyer!");  
 }  
  
 // java.sql Interface Statement  
 // ResultSet getGeneratedKeys() throws SQLException  
 // Retrieves any auto-generated keys created as a result of executing  
 // this Statement object.  
 // https://docs.oracle.com/javase/8/docs/api/java/sql/Statement.html  
  
  
 // we use the getGeneratedKeys method to get a set that contains the generated key  
 // in this case it's just one key  
 ResultSet generatedKeys = insertIntoIndividualBuyers.getGeneratedKeys(); // we retrieve  
 // the id from the new generated IB  
  
 if(transactions!=0) {  
 insertIntoMapTransactions.setLong(1, transactions);  
  
 }  
 if (generatedKeys.next()) {  
 // key at position one since we only expect one key to be returned  
 return generatedKeys.getString(1);  
 } else {  
 throw new SQLException("Couldn't get id for IndividualBuyer");  
 }  
 }  
 }  
  
 public void insertTransaction( long transactionNumber, long value, String description) {  
  
 try {  
 // Interface Connection  
 // void setAutoCommit(boolean autoCommit)  
 // Sets this connection's auto-commit mode to the given state.  
 // https://docs.oracle.com/javase/8/docs/api/java/sql/Connection.html  
 conn.setAutoCommit(false);  
  
  
 insertIntoTransactions.setLong(1, transactionNumber);  
 insertIntoTransactions.setLong(2, value);  
 insertIntoTransactions.setString(3, description);  
  
 int affectedRows = insertIntoTransactions.executeUpdate();  
 if (affectedRows == 1) {  
 conn.commit();  
 } else {  
 throw new SQLException("The transaction insert failed");  
 }  
  
 } catch (Exception e) {  
 System.*out*.println("Insert transaction exception: " + e.getMessage());  
 try {  
 System.*out*.println("Performing rollback");  
 conn.rollback();  
 } catch (SQLException e2) {  
 System.*out*.println("Oh boy! Things are really bad! " + e2.getMessage());  
 }  
 } finally {  
 try {  
 System.*out*.println("Resetting default commit behavior");  
 conn.setAutoCommit(true);  
 } catch (SQLException e) {  
 System.*out*.println("Couldn't reset auto-commit! " + e.getMessage());  
 }  
  
 }  
 }  
  
  
}