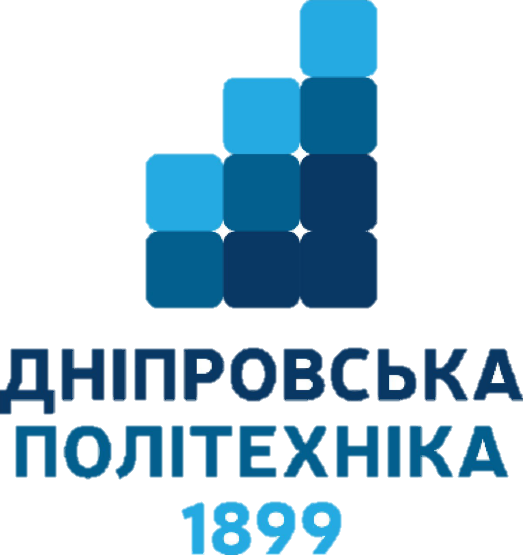
Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»



Факультет інформаційних технологій

Кафедра системного аналізу і управління

*Звіт*

З дисципліни:

“ОПСJ”

**Виконав**

студент групи 124-18ск-1

Владимиров Я.Д

**GitHub**

https://github.com/gotoindex/javalabs.git

**Перевірив**

Мінєєв О.С

м. Дніпро

2020

Лабораторна робота №6

Хід роботи

Постановка задачі:

Створити базу даних в будь  якому сервері баз даних. Створити таблицю з переліком студентів вказати їх прізвище, ім'я, по батькові, день народження номер залікової книжки та ID.

Створити програму  що буде дозволяти виводити на екран  інформацію про студентів  які народилися в тому чи іншому місяці року.  Програма повинна завдяки системі jdbc під'єднатися до вашої бази даних та робити до неї запроси.  Вимог до розробки бази даних немає.  Програма ж має бути написана за усіма стандартами ООП.

Лістинг програми:

*ConnectionManager.java:*

**package** lr6;  
  
**import** java.sql.Connection;  
**import** java.sql.DriverManager;  
**import** java.sql.SQLException;  
  
*/\*\*  
 \* This class contains a basic toolkit for connecting to a single  
 \* pre-defined database called <i>lr6</i>.  
 \*  
 \* <p>Please make sure you are using the correct version of Idea,  
 \* if you're not sure which version you are using please contact  
 \* the administrator. The version intended for usage is maven-14.  
 \*  
 \* <p>The documentation for the methods contained in this class includes  
 \* brief descriptions of the <i>implementations</i>. Such descriptions should  
 \* be regarded as <i>implementation notes</i>, rather than parts of the  
 \* <i>specification</i>. Implementors should feel free to substitute other  
 \* algorithms, so long as the specification itself is adhered to. (For  
 \* example, the algorithm used by {****@code*** *sort(Object[])} does not have to be  
 \* a MergeSort, but it does have to be <i>stable</i>.)  
 \*  
 \* <p>This class is a member of the  
 \* <a href="https://github.com/gotoindex/javalabs">  
 \* Java Learning Course</a>.  
 \*  
 \** ***@author*** *Akim Vladimirov  
 \*/***public class** ConnectionManager {  
  
 */\*  
 \* These are pre-defined credentials of the database that was used for testing.  
 \*  
 \* If you want to test the connection you should replace these credentials  
 \* with yours.  
 \*/* **static final** String ***URL*** = **"jdbc:postgresql://localhost:5432/lr6"**;  
 **static final** String ***USER*** = **"postgres"**;  
 **static final** String ***PASSWORD*** = **"postgres"**;  
  
 *// This attribute can be accessed outside of the class via the getConnection() method.* **private** Connection **conn**;  
  
 */\*\*  
 \* Establishes the connection using provided credentials.  
 \*  
 \** ***@implNote*** *In case of failed connection, further actions will throw  
 \* {****@code*** *java.sql.SQLException} error.  
 \*/* **public void** connect() {  
 **try** {  
 **this**.**conn** = DriverManager.*getConnection*(***URL***, ***USER***, ***PASSWORD***);  
 System.***out***.println(**"Connected to the database."**);  
 }  
 **catch** (SQLException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 */\*\*  
 \* The use of the connection instance outside of the class is  
 \* purely for the query-making.  
 \*/* **public** Connection getConnection() {  
 **return this**.**conn**;  
 }  
  
 */\*\*  
 \* Closes the connection if active.  
 \*/* **public void** closeConnection() {  
 **try** {  
 **this**.**conn**.close();  
 System.***out***.println(**"Closed the connection."**);  
 }  
 **catch** (SQLException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
}

*DataBaseManager.java:*

**package** lr6;  
  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.ResultSetMetaData;  
**import** java.sql.SQLException;  
  
*/\*\*  
 \* This class contains a set of basic queries to the <i>lr6</i> database.  
 \*  
 \* <p>Please make sure you are using the correct version of Idea,  
 \* if you're not sure which version you are using please contact  
 \* the administrator. The version intended for usage is maven-14.  
 \*  
 \* <p>The documentation for the methods contained in this class includes  
 \* brief descriptions of the <i>implementations</i>. Such descriptions should  
 \* be regarded as <i>implementation notes</i>, rather than parts of the  
 \* <i>specification</i>. Implementors should feel free to substitute other  
 \* algorithms, so long as the specification itself is adhered to. (For  
 \* example, the algorithm used by {****@code*** *sort(Object[])} does not have to be  
 \* a MergeSort, but it does have to be <i>stable</i>.)  
 \*  
 \* <p>This class is a member of the  
 \* <a href="https://github.com/gotoindex/javalabs">  
 \* Java Learning Course</a>.  
 \*  
 \** ***@author*** *Akim Vladimirov  
 \*/***public class** DataBaseManager {  
  
 *// This object contains the connection object used to access the database.* **private final** ConnectionManager **database** = **new** ConnectionManager();  
  
 */\*\*  
 \* Basic constructor that tries to connect to the database upon execution.  
 \*  
 \** ***@implNote*** *Note that if the connection fails, any further actions may lead to  
 \* a {****@code*** *java.sql.SQLException} throw.  
 \*/* **public** DataBaseManager() {  
 System.***out***.println(**"Trying to connect to the database..."**);  
 **database**.connect();  
 }  
  
 */\*\*  
 \* Closes the current connection.  
 \*  
 \** ***@implNote*** *You should always use this method at the end of a session  
 \* and only in the end of a session.  
 \*/* **public void** exit() {  
 System.***out***.println(**"Exiting and closing the connection..."**);  
 **database**.closeConnection();  
 }  
  
 */\*\*  
 \* Adds a new student to the database.  
 \*  
 \** ***@param name*** *the first name of the student.  
 \** ***@param surname*** *the last name of the student.  
 \** ***@param patronymic*** *the patronymic of the student.  
 \** ***@param birthday*** *the birth date of the student.  
 \*/* **public void** create(String name, String surname, String patronymic, String birthday) {  
 String sql = **"INSERT INTO students (name, surname, patronymic, birthday) VALUES (?, ?, ?, ?)"**;  
 **try** {  
 executeInsertQuery(sql, name, surname, patronymic, birthday);  
 System.***out***.println(**"Successfully created a student!"**);  
 } **catch** (SQLException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 */\*\*  
 \* Finds and displays all student with the given birthday.  
 \*  
 \** ***@implNote*** *The algorithm also returns the amount of entries found.  
 \*  
 \** ***@param birthday*** *the date that should be used in the search.  
 \*/* **public int** searchByBirthday(String birthday) {  
 String sql = **"SELECT \* FROM students WHERE birthday = ?"**;  
 **int** resultCount = 0;  
 **try** {  
 resultCount = executeSelectQuery(sql, birthday);  
 System.***out***.println(**"Successfully performed a search!"**);  
 } **catch** (SQLException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 **return** resultCount;  
 }  
  
 */\*\*  
 \* Clears all students from the database.  
 \*  
 \** ***@implNote*** *The algorithm performs a full DELETE query to clear all entries.  
 \*/* **public void** clearTable() {  
 String sql = **"DELETE FROM students"**;  
 **try** {  
 PreparedStatement query = **database**.getConnection().prepareStatement(sql);  
 query.executeUpdate();  
 query.close();  
 System.***out***.println(**"Successfully cleared the database!"**);  
 } **catch** (SQLException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 */\*\*  
 \* Executes INSERT query to add a student.  
 \*  
 \** ***@param sql*** *the sql code that will be executed.  
 \** ***@param name*** *the first name of the student.  
 \** ***@param surname*** *the last name of the student.  
 \** ***@param patronymic*** *the patronymic of the student.  
 \** ***@param birthday*** *the birth date of the student.  
 \*  
 \** ***@throws*** *SQLException if the connection or the sql string is invalid.  
 \*/* **private void** executeInsertQuery(String sql, String name, String surname, String patronymic,  
 String birthday) **throws** SQLException {  
 PreparedStatement query = **database**.getConnection().prepareStatement(sql);  
 query.setString(1, name);  
 query.setString(2, surname);  
 query.setString(3, patronymic);  
 query.setString(4, birthday);  
 query.executeUpdate();  
 query.close();  
 }  
  
 */\*\*  
 \* Finds students with matching birthday.  
 \*  
 \** ***@implNote*** *The algorithm also returns the amount of entries found.  
 \*  
 \** ***@param sql*** *the sql code that will be executed.  
 \** ***@param birthday*** *the date that should be used in the search.  
 \*  
 \** ***@throws*** *SQLException if the connection or the sql string is invalid.  
 \*/* **private int** executeSelectQuery(String sql, String birthday) **throws** SQLException {  
 PreparedStatement query = **database**.getConnection().prepareStatement(sql);  
 query.setString(1, birthday);  
 **int** resultCount = printQueryResults(query.executeQuery());  
 query.close();  
 **return** resultCount;  
 }  
  
 */\*\*  
 \* Prints the results of a SELECT query.  
 \*  
 \** ***@implNote*** *The algorithm cycles through the entries of the result of  
 \* the query and displays it to the user. It also counts how many  
 \* matches were found.  
 \*  
 \** ***@param result*** *the cleared up response from the query.  
 \*  
 \** ***@throws*** *SQLException if the result data is incorrect.  
 \*/* **private int** printQueryResults(ResultSet result) **throws** SQLException {  
 ResultSetMetaData meta = result.getMetaData();  
 **int** i = 1;  
 **while** (result.next()) {  
 System.***out***.print(**"Row "** + i++ + **": "**);  
 **for** (**int** j = 1; j < meta.getColumnCount() + 1; j++) {  
 System.***out***.print(meta.getColumnLabel(j) + **": "** + result.getObject(j) + **", "**);  
 }  
 System.***out***.println();  
 }  
 **return** i - 1;  
 }  
}

*DataTest.java:*

**package** lr6;  
  
**import** org.junit.jupiter.api.Assertions;  
**import** org.junit.jupiter.api.Test;  
  
**public class** DataTest {  
  
 @Test  
 **public void** insertCheck() {  
 DataBaseManager manager = **new** DataBaseManager();  
 **try** {  
 manager.create(**"Daniil"**, **"Mokhov"**, **"Dmytrovych"**, **"1999-11-01"**);  
 manager.create(**"Alexander"**, **"Prokopenko"**, **"Alexandrovych"**, **"1998-01-01"**);  
 manager.create(**"Alex"**, **"Nikitenko"**, **"Olegovych"**, **"1998-11-03"**);  
 manager.create(**"Marina"**, **"Prokopenko"**, **"Alexandrovna"**, **"1998-01-01"**);  
 Assertions.*assertEquals*(manager.searchByBirthday(**"1998-01-01"**), 2);  
 manager.clearTable();  
 } **finally** {  
 manager.exit();  
 }  
 }  
}

Результати виконання:

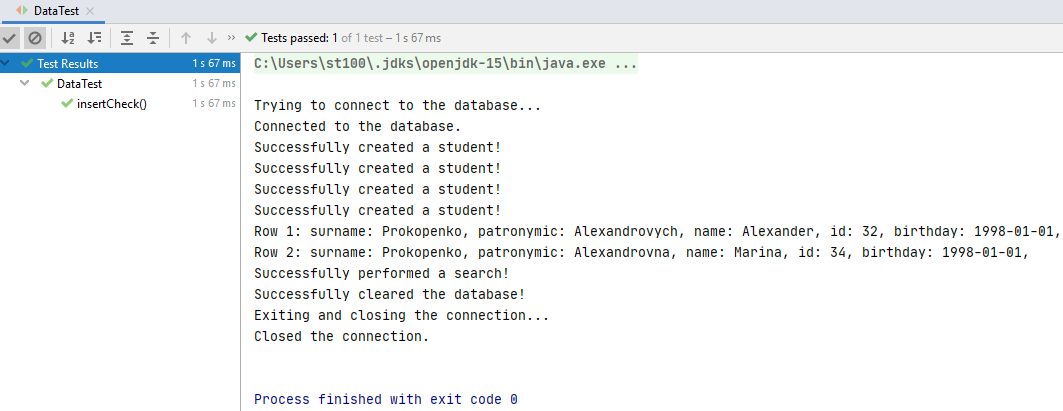


Рисунок 6.1 – Результат виконання програми

Висновок: Під час виконання лабораторної роботи №6 було створено програму, що буде дозволяти виводити на екран  інформацію про студентів, які народилися в тому чи іншому місяці та року.