Programowanie obiektowe

Laboratorium 4.

Zadanie polega na stworzeniu programu wykorzystującego JDBC do połączenia z bazą danych. Jako system bazodanowy wykorzystamy SQLite.

Do wykorzystania JDBC z konkretnym systemem bazodanowym konieczny jest sterownik. Sterownik dla SQLite pobieramy ze strony https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc (plik sqlite-jdbc-NUMER_WERSJI.jar)

Pobrany plik zapisujemy w lokalnym folderze.

Tworzymy projekt w IntelliJ przez File>New Project (zwykły projekt w Javie).

W głównym pliku programu umieszczamy następujący kod:

```
import java.sql.*;

public class Main {
   public static void main( String args[] ) {
      Connection c = null;

      try {
         Class.forName("org.sqlite.JDBC");
         c = DriverManager.getConnection("jdbc:sqlite:test.db");
      } catch ( Exception e ) {
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         System.exit(0);
      }
      System.out.println("Opened database successfully");
   }
}
```

Aby uruchomić program, należy dołączyć do konfiguracji projektu pobrany sterownik.

Robimy to wybierając **File>Project structure>Project Setting>Modules>Dependencies**

Klikamy **+ >Jar or Directory** i wybieramy plik sterownika **sqlite-jdbc-NUMER_WERSJI.jar**, który wcześniej pobraliśmy.

Po dodaniu sterownika do konfiguracji można uruchomić projekt. Jeśli wszystko przebiegło pomyślnie, to w konsoli pojawi się napis: „Opened database successfully”. Jednocześnie w głównym folderze projektu pojawi się plik test.db (program utworzy ten plik jeśli wcześniej tam nie istniał).

Tworzenie tablicy

W kolejnej wersji tego programu utworzymy tabelę w bazie danych. Metoda main powinna być następująca:

```
  public static void main( String args[] ) {
    Connection c = null;
    Statement stmt = null;
```

```
    try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
      System.out.println("Opened database successfully");

      stmt = c.createStatement();
      String sql = "CREATE TABLE COMPANY " +
                   "(ID INT PRIMARY KEY     NOT NULL," +
                   " NAME           TEXT    NOT NULL, " +
                   " AGE            INT     NOT NULL, " +
                   " ADDRESS        CHAR(50), " +
                   " SALARY         REAL)";
      stmt.executeUpdate(sql);
      stmt.close();
      c.close();
    } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);
    }
    System.out.println("Table created successfully");
  }
```

Program można uruchomić tylko raz. Przy kolejnym uruchomieniu wygeneruję błąd, gdyż tabela jest już utworzona.

Operacja INSERT

Kolejna wersja programu zademonstruje operacje dodania danych do tabeli. Metoda main powinna wyglądać następująco:

```
  public static void main( String args[] ) {
    Connection c = null;
    Statement stmt = null;

    try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
      c.setAutoCommit(false);
      System.out.println("Opened database successfully");

      stmt = c.createStatement();
      String sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
                   "VALUES (1, 'Paul', 32, 'California', 20000.00 );";
      stmt.executeUpdate(sql);

      sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
            "VALUES (2, 'Allen', 25, 'Texas', 15000.00 );";
      stmt.executeUpdate(sql);

      sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
            "VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );";
      stmt.executeUpdate(sql);
```

```
      sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
            "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );";
      stmt.executeUpdate(sql);

      stmt.close();
      c.commit();
      c.close();
    } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);
    }
    System.out.println("Records created successfully");
  }
```

Operacja SELECT

Kolejna wersja programu demonstruje odczyt danych

```
  public static void main( String args[] ) {

    Connection c = null;
    Statement stmt = null;
    try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
      c.setAutoCommit(false);
      System.out.println("Opened database successfully");

      stmt = c.createStatement();
      ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );

      while ( rs.next() ) {
        int id = rs.getInt("id");
        String  name = rs.getString("name");
        int age  = rs.getInt("age");
        String  address = rs.getString("address");
        float salary = rs.getFloat("salary");

        System.out.println( "ID = " + id );
        System.out.println( "NAME = " + name );
        System.out.println( "AGE = " + age );
        System.out.println( "ADDRESS = " + address );
        System.out.println( "SALARY = " + salary );
        System.out.println();
      }
      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 successfully");
 }
```

Operacja UPDATE

Kolejna wersja programu to aktualizacja danych:

```
 public static void main( String args[] ) {

  Connection c = null;
  Statement stmt = null;

  try {
    Class.forName("org.sqlite.JDBC");
    c = DriverManager.getConnection("jdbc:sqlite:test.db");
    c.setAutoCommit(false);
    System.out.println("Opened database successfully");

    stmt = c.createStatement();
    String sql = "UPDATE COMPANY set SALARY = 25000.00 where ID=1;";
    stmt.executeUpdate(sql);
    c.commit();

    ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );

    while ( rs.next() ) {
      int id = rs.getInt("id");
      String  name = rs.getString("name");
      int age  = rs.getInt("age");
      String  address = rs.getString("address");
      float salary = rs.getFloat("salary");

      System.out.println( "ID = " + id );
      System.out.println( "NAME = " + name );
      System.out.println( "AGE = " + age );
      System.out.println( "ADDRESS = " + address );
      System.out.println( "SALARY = " + salary );
      System.out.println();
    }
    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 successfully");
  }
```

Operacja DELETE

Wersja programu demonstrująca usunięcie wiersza z tablicy.

```
public static void main( String args[] ) {
  Connection c = null;
  Statement stmt = null;

  try {
    Class.forName("org.sqlite.JDBC");
    c = DriverManager.getConnection("jdbc:sqlite:test.db");
    c.setAutoCommit(false);
    System.out.println("Opened database successfully");

    stmt = c.createStatement();
    String sql = "DELETE from COMPANY where ID=2;";
    stmt.executeUpdate(sql);
    c.commit();

    ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );

    while ( rs.next() ) {
    int id = rs.getInt("id");
    String  name = rs.getString("name");
    int age  = rs.getInt("age");
    String  address = rs.getString("address");
    float salary = rs.getFloat("salary");

    System.out.println( "ID = " + id );
    System.out.println( "NAME = " + name );
    System.out.println( "AGE = " + age );
    System.out.println( "ADDRESS = " + address );
    System.out.println( "SALARY = " + salary );
    System.out.println();
  }
  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 successfully");
}
```

Zadanie

Na podstawie powyższych przykładów, należy stworzyć program z następującymi funkcjami:

1. Funkcja tworząca tabelę z listą obecności studentów na zajęciach on line. Tabela powinna obejmować:
    a. Prowadzącego
    b. Nazwę przedmiotu
    c. Imię i nazwisko studenta,

    d. Numer albumu
    e. Datę zajęć
    f. Czas zalogowania
    g. Czas wylogowania
    h. Długość czasu spędzonego na sesji.
2. Funkcję dodającą studenta do listy obecności. Podczas dodawania powinny być uzupełnione dane dotyczące obecności
3. Funkcję wyświetlającą obecności wszystkich studentów na zajęciach.