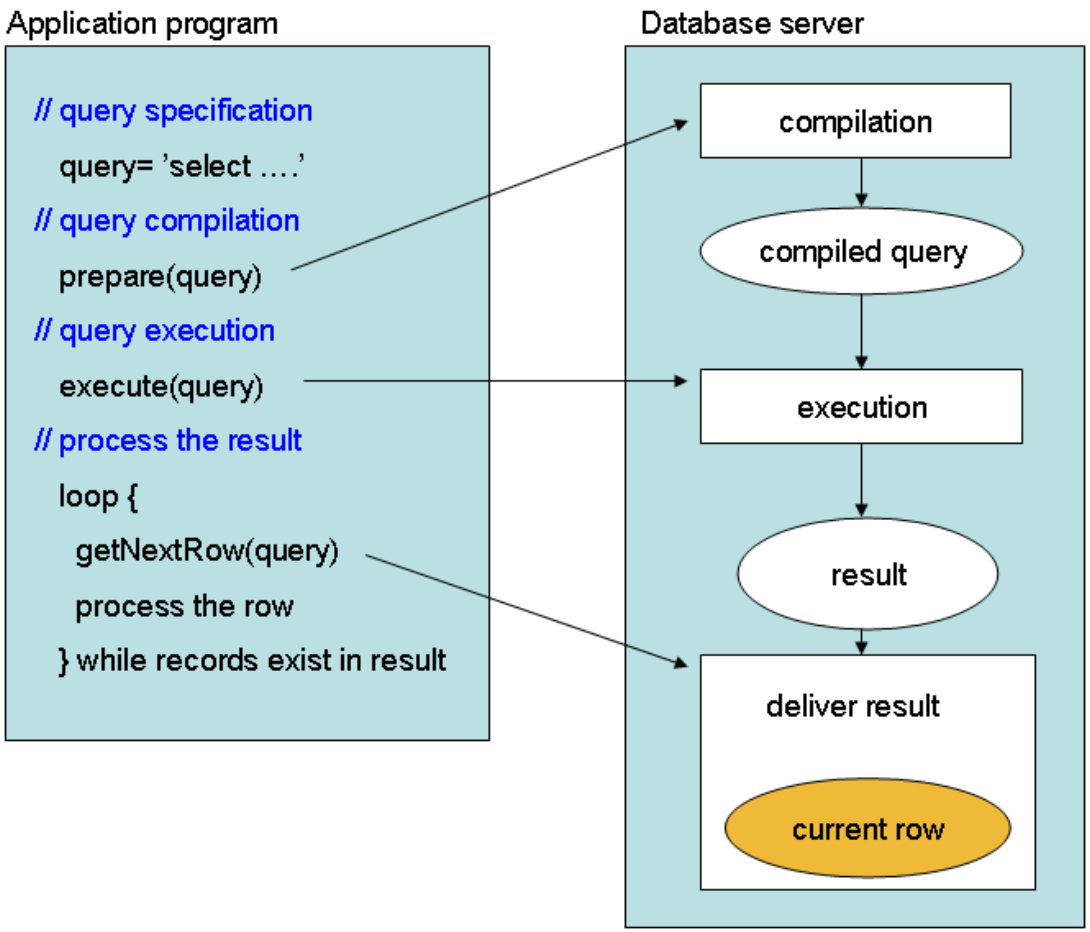
האקדמית עמק יזרעאל
האקדמיה של הצפון!

מבוא למסדי נתונים

שילוב SQL בשפת התכנות

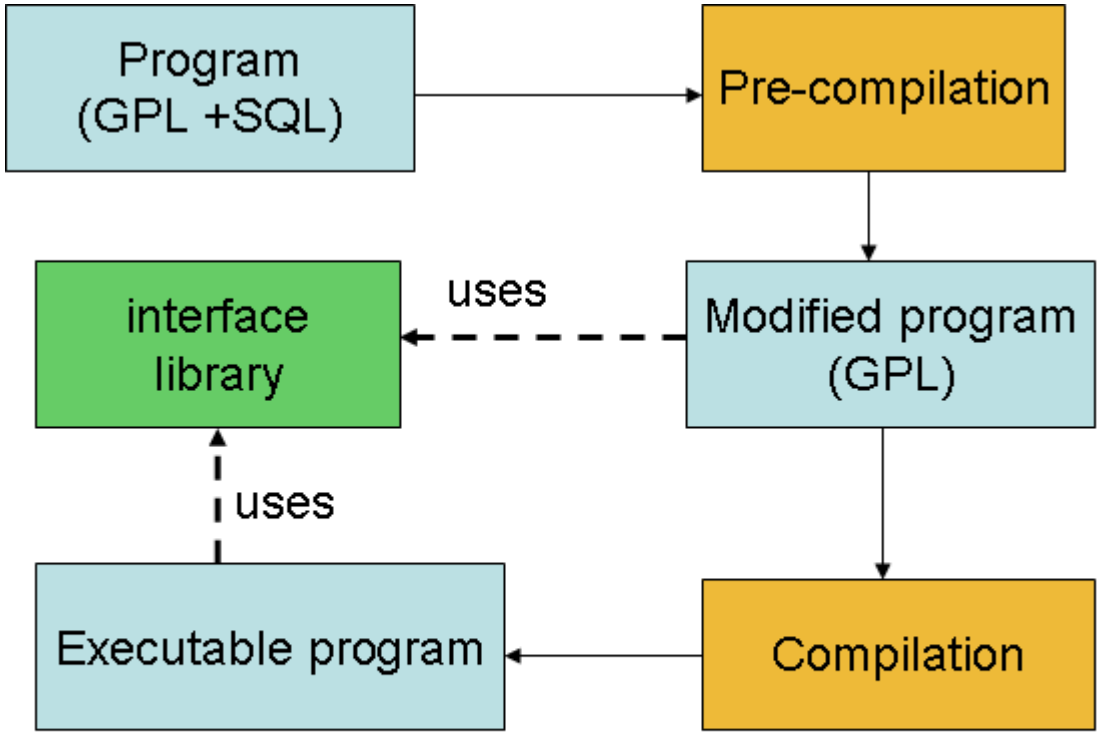
ד"ר רמי רשקוביץ

שילוב SQL בשפת תכנות



- שיטות שונות לשילוב SQL
  - Embedded SQL – application programming interfaces (API)

שילוב SQL בשפת תכנות



GPL = general purpose programming language

- Embedded SQL
  - שיבוץ פקודות SQL בתוך שפת התכנות
    - נפוץ ב-C וב-COBOL

שילוב SQL בשפת תכנות

- Embedded SQL
  - דוגמה

```
int main() {
   EXEC SQL INCLUDE SQLCA;
   EXEC SQL BEGIN DECLARE SECTION;
      int OrderID;          /* Employee ID (from user)        */
      int CustID;             /* Retrieved customer ID        */
      char SalesPerson[10]    /* Retrieved salesperson name      */
      char Status[6]          /* Retrieved order status       */
   EXEC SQL END DECLARE SECTION;

   /* Set up error processing */
   EXEC SQL WHENEVER SQLERROR GOTO query_error;
   EXEC SQL WHENEVER NOT FOUND GOTO bad_number;

   /* Prompt the user for order number */
   printf ("Enter order number: ");
   scanf_s("%d", &OrderID);

   /* Execute the SQL query */
   EXEC SQL SELECT CustID, SalesPerson, Status
      FROM Orders
      WHERE OrderID = :OrderID
      INTO :CustID, :SalesPerson, :Status;

   /* Display the results */
   printf ("Customer number:  %d\n", CustID);
   printf ("Salesperson: %s\n", SalesPerson);
   printf ("Status: %s\n", Status);
   exit();

query_error:
   printf ("SQL error: %ld\n", sqlca->sqlcode);
   exit();

bad_number:
   printf ("Invalid order number.\n");
   exit();
}
```

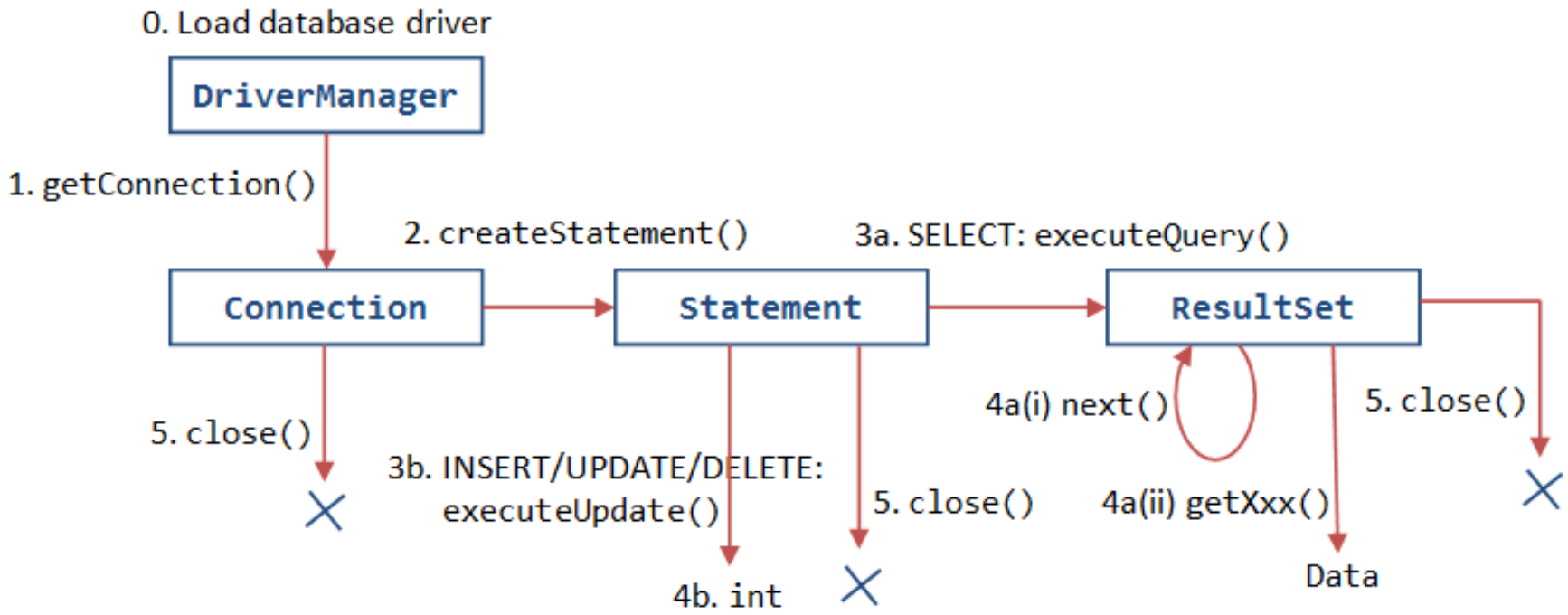
שילוב SQL בשפת תכנות

- Application Programming Interface (API)
- לכל תוכנה של מסד נתונים (Oracle, SQL Server, ...) יש ספרית פונקציות שניתן להפעיל אותן לצורך ביצוע פעולות שונות ב-Database
  - למעשה, אלו הפונקציות שה-pre-compiler שפועל בסביבת ה-embedded sql "שותל" בתוך קוד
  - ספריות אלו נקראות Native API
- בנוסף לספריות ה-Native קיימות גם ספריות שאינן תלויות בתוכנת מסד נתונים כזה או אחר. ספריות לדוגמה:

ODBCODBC (Microsoft Open Database Connection), JDBC (for Java)

  - היתרון של ספריות אלו ע"פ ספריות ה-Native הוא בכך שהוא מספק למתכנת ממשק סטנדרטי לעבודה מול מסד הנתונים ללא קשר למסד הנתונים המסוים בה הוא עושה שימוש. לשם כך עליו לטעון Driver מהסוג המתאים
  - החיסרון של ספריות אלו עשוי לבוא לידי ביטוי בביצועים

עבודה מול מסד נתונים באמצעות JDBC



סדר פעולות במסד נתונים

- פתיחת מסד הנתונים – Connection
  - Connection String
    - Provider – מנוע מסד הנתונים
    - שם הקובץ ומיקומו
    - שם משתמש
    - סיסמא
- ביצוע פעולות על מסד הנתונים
  - Command/Statement
- סגירת מסד הנתונים
  - סגירת ה-Command/Statement
  - סגירת ה-Connection

עבודה מול מסד נתונים באמצעות JDBC

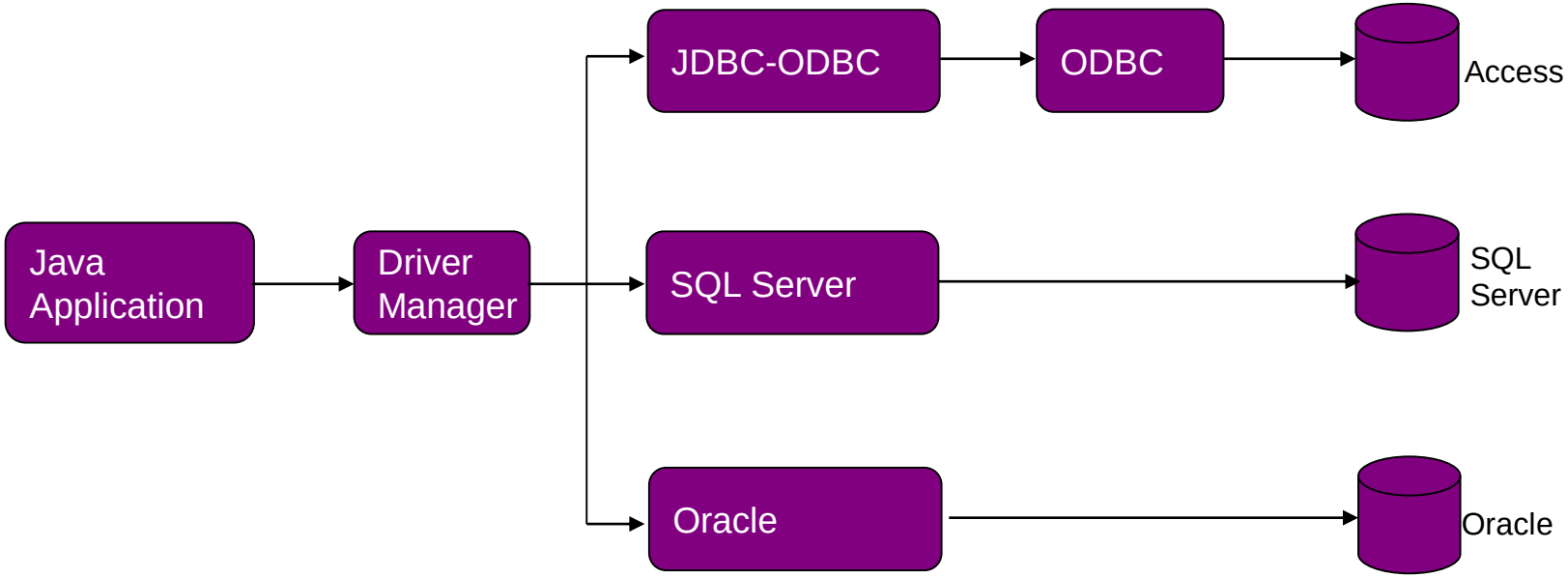
```
import java.sql.*;
```

- Java Data Base Connectivity – JDBC
  - אוסף ממשקים – package java.sql.*;
  - ממשק בין תכנית java למסד הנתונים

- Driver - תוכנת גישה למסד נתונים
  - מקשר בין תכנית Java למסד הנתונים
  - מאפשר כתיבת קוד אחיד לטיפול בכל מסדי הנתונים

המשך - JDBC – Driver

- לכל טיפוס מסד נתונים קיים Driver שלו המסופק ע"י יצרן מסד הנתונים



יצירת Connection

- טעינת Driver

Class.forName("driverName")

- throws ClassNotFoundException (checked)

– התקנת Driver

– https://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

יצירת Connection

- יצירת אובייקט Connection

Connection conn = DrivareManager.*getConnection*(connectionString)

```
jdbc:sqlserver://[serverName[\instanceName][:portNumber]][;property=value[;property=value]]
```

jdbc:sqlserver:// (Required)
serverName is the address of the server to connect to. This could be a DNS or IP address, or it could be localhost or 127.0.0.1 for the local computer.
instanceName (Optional) is the instance to connect to on serverName. If not specified, a connection to the default instance is made.
portNumber (Optional) is the port to connect to on serverName. The default is 1433.
property (Optional) is one or more option connection properties. For more information, see Setting the Connection Properties.

```
"jdbc:sqlserver://localhost;databaseName=Dance;integratedSecurity=true;" ;

"jdbc:sqlserver://localhost;user=MyUserName;password=*****";

"jdbc:sqlserver://localhost;databaseName=Dance;integratedSecurity=true;"
```

הפקת שאילתה

- יצירת Statement

```
Statement statement = conn.createStatment()
```

- הפקת השאילתא

```
ResutSet resultSet = statement.executeQuery("SQL_Statement")
```

or

```
RowSet resultSet = statement.executeQuery("SQL_Statement")
```

- כל הפעולות מול מסד הנתונים "זורקות" SQLException

ביצוע פעולות וסגירה

- עיבוד הנתונים

```
while (resultSet.next()) {
    // read fields of current data
    resultSet.getXXX(fieldNumber/"fieldName");
```

- getXXX = getInt(), getDouble(), getString(), …
- fieldNumber = 1…N

```
    // do something with the data
}
```

מיפוי טיפוסי נתונים של SQL Server

SQL type	Java Type
CHAR, VARCHAR, LONGVARCHAR	String
NUMERIC, DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT, DOUBLE	double
BINARY, VARBINARY, LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

ביצוע פעולות וסגירה

- סגירת ה-Connection
  - Stmt.close();
  - resultSet.close()
  - con.close()

- אי סגירה של האובייקטים לאחר תום השימוש תגרום לבזבוז משאבים ולבעיות אבטחת מידע פוטנציאליות

Connections

- SQLServer
  - Driver : **"com.microsoft.sqlserver.jdbc.SQLServerDriver"**
  - URL : **"jdbc:sqlserver://localhost:1433; databaseName=DanceDB;user=UserName;password=*****"**

במקום localhost אפשר לספק כתובת IP של השרת

- MySQL
  - Driver : **"org.gjt.mm.mysql.Driver"**
  - URL : **"jdbc:mysql://localhost/test?user=minty&password=greatsqldb""**

- Oracle
  - Driver : **"oracle.jdbc.driver.OracleDriver"**
  - URL : **"jdbc:oracle:thin:@myhost:1521:orcl", "scott", "tiger"**

עבודה עם Eclipse מול SQL Server

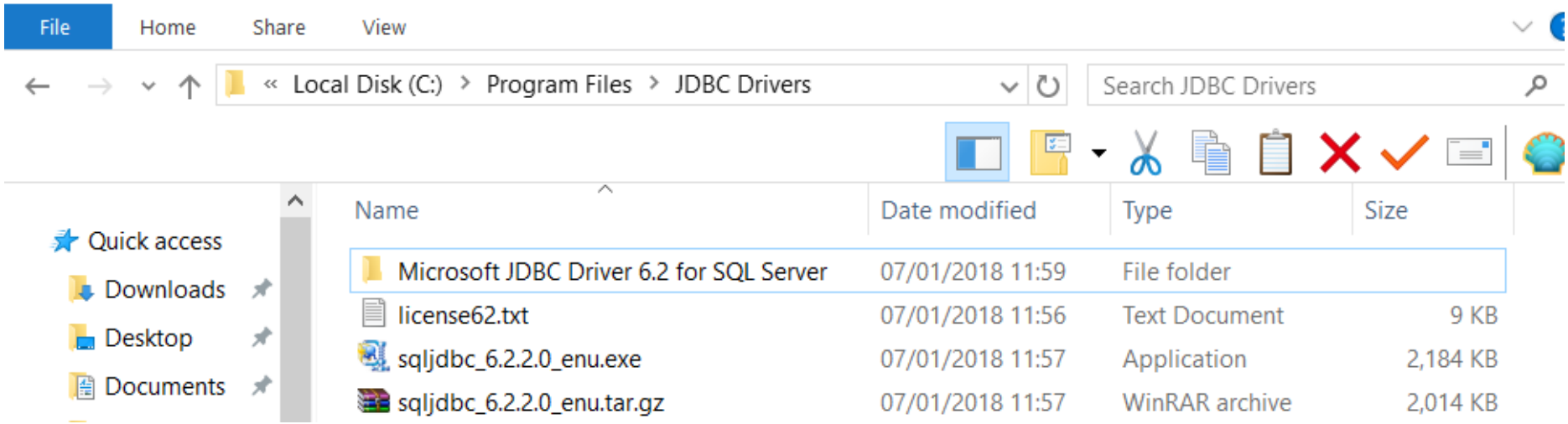
- תחילה יש להוריד JDBC Drivers עבור SQL Server (2017)
  - https://www.microsoft.com/en-us/download/details.aspx?id=55539

Choose the download you want

File Name	Size
license62.txt	9 KB
sqljdbc_6.2.2.0_enu.exe	2.1 MB
sqljdbc_6.2.2.0_enu.tar.gz	2.0 MB

עבודה עם Eclipse מול SQL Server

- לשמור את הקבצים הפרוסים (unzip) בתיקייה כלשהי
  - נניח C:\Program Files\JDBCDrivers



עבודה עם Eclipse מול SQL Server

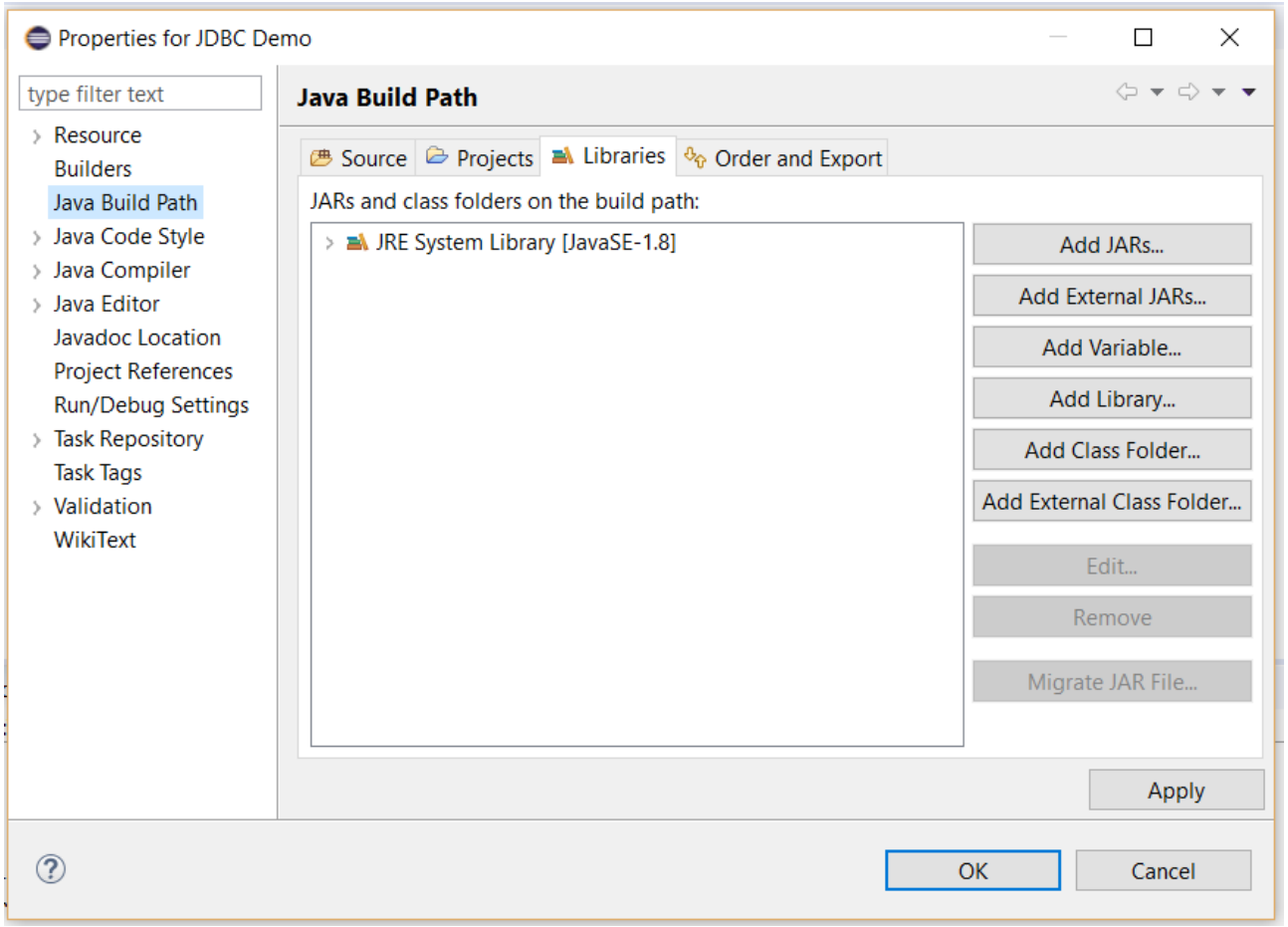
- התקנת גרסת JRE עדכנית
  - נכון להיום version 8.151

http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

- בעת פתיחת פרויקט חדש ב-Eclipse יש לוודא בחירה ב-JRE המעודכן ביותר

עבודה עם Eclipse מול SQL Server

- בפרויקט שנפתח יש ללכת ל-Project Properties
- יש לבחור ב- Java Build Path ובלשונית Libraries

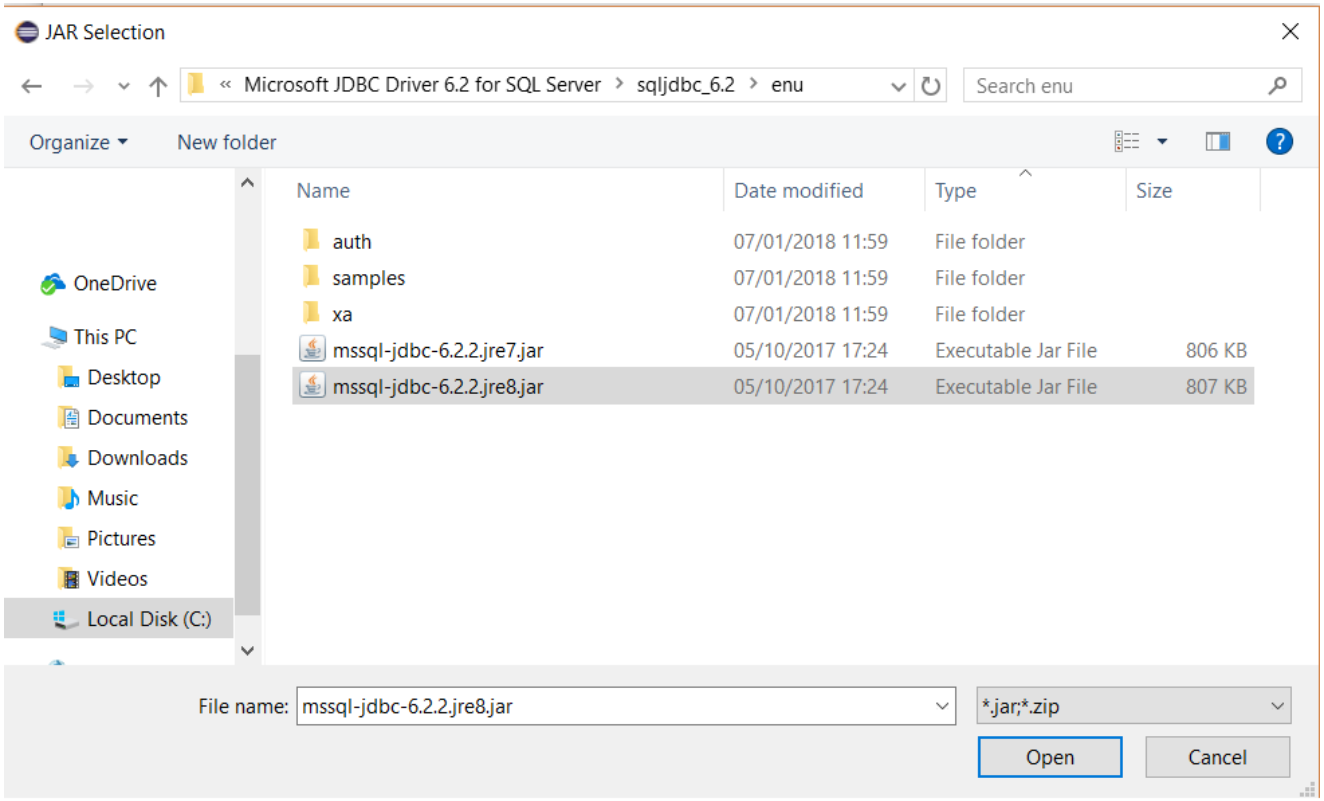


עבודה עם Eclipse מול SQL Server

- יש לבחור ב-Add External Jar ובחלון שנפתח לנווט לתיקייה בה הותקנו ה-drivers לתיקייה הפנימית

C:\Program Files\JDBC Drivers\Microsoft JDBC Driver 6.0 for SQL Server\sqljdbc_6.0\enu

- ושם לבחור בקובץ אשר תואם לגרסת JRE 8



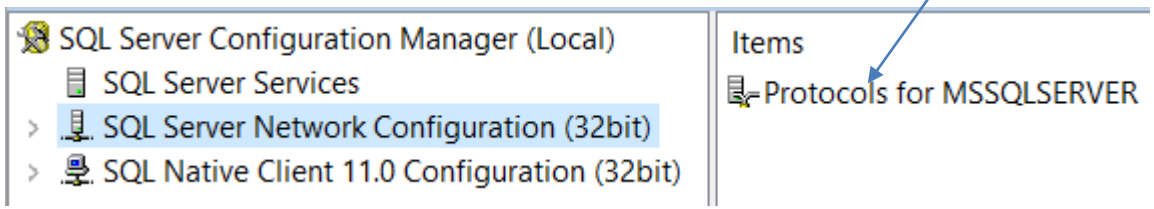
עדכון SQL Server

- יש לפתוח את SQL Server Configuration Management
  - לצורך הגדרת תצורת שרת SQL Server
  - אם אינכם מוצאים את האפשרות הנ"ל תחת תפריט התחל תוכלו לאתר את קובץ ההפעלה של התוכנית הנ"ל (בהתאם לגרסה) תחת תיקיית C:\Windows\SysWOW64\
    - לחיצה כפולה על הקובץ תפתח את ה-Configuration Manager

SQL Server 2017	C:\Windows\SysWOW64\SQLServerManager14.msc
SQL Server 2016	C:\Windows\SysWOW64\SQLServerManager13.msc
SQL Server 2014	C:\Windows\SysWOW64\SQLServerManager12.msc
SQL Server 2012	C:\Windows\SysWOW64\SQLServerManager11.msc

עדכון SQL Server

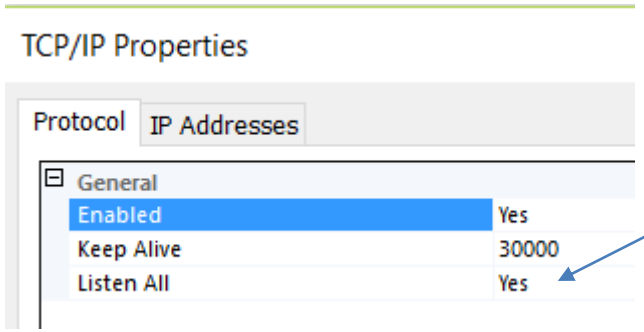
- בחלון שנפתח יש לבחור ב SQL Server Network Configuration, ולחיצה כפולה על Protocols



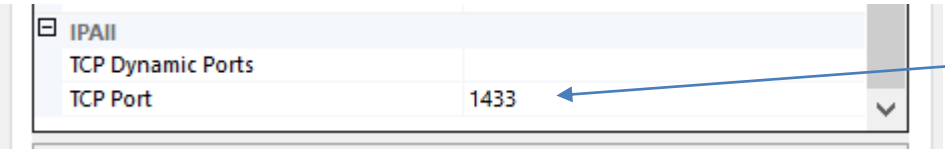
- יש לבחור ב- TCP/IP ולשנות באמצעות תפריט ההקשר (לחצן ימני) את מצבו ל-Enable

עדכון SQL Server

- לאחר מכן יש ללחוץ לחיצה כפולה על TCP/IP, ולוודא שבשדה Listen All מופיע yes



- לאחר מכן יש ללחוץ לחיצה כפולה על TCP/IP, ולוודא שבשדה Listen All מופיע yes
- לאחר מכן לבחור בלשונית IP Addresses
- לגלול למטה ולוודא שבשדה TCP PORT תחת IPALL מופיע הערך 1433



– מידע נוסף ניתן למצוא ב-

- https://support.ca.com/us/knowledge-base-articles.TEC1176711.html

עבודה עם Eclipse מול SQL Server

- כעת ניתן להתחבר לשרת עם ConnectionString

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String connectionUrl = "jdbc:sqlserver://localhost:1433;" +
 "databaseName=MyDb;user=MyUserName;password=*****;";
Connection con = DriverManager.getConnection(connectionUrl);
```

- אם עובדים עם Windows Authentication

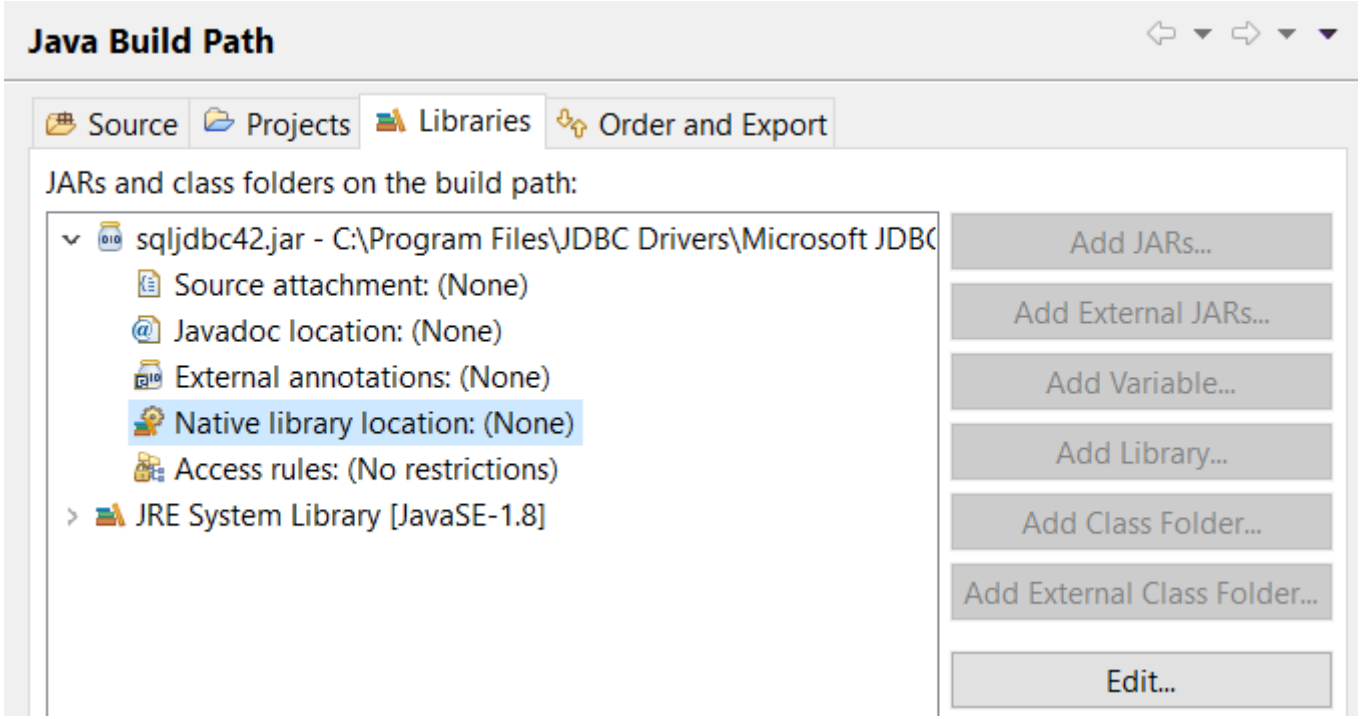
```
String connectionUrl = "jdbc:sqlserver://localhost:1433;"
        + "databaseName=MyDB;integratedSecurity=true";
```

– אם מקבלים שגיאת התחברות מסוג:

com.microsoft.sqlserver.jdbc.AuthenticationJNI <clinit>
WARNING: Failed to load the sqljdbc_auth.dll cause : no sqljdbc_auth in java.library.path

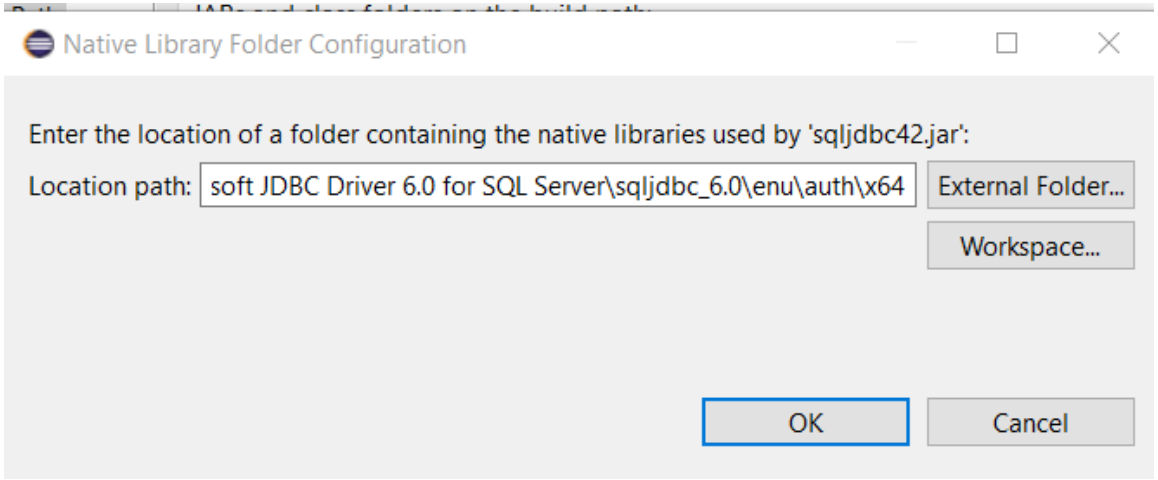
עבודה עם Eclipse מול SQL Server

- במקרה של שגיאה יש לפעול עפ"י ההנחיות הבאות:
- נפתח שוב את Project Properties ונבחר בלשונית Libraries שם נפתח את קובץ ה-Jar שהוספנו קודם לכן, נסמן את Native Library Location ונבחר ב-Edit



עבודה עם Eclipse מול SQL Server

- בחלון שנפתח נבחר ב-External Folder



- וננווט לתיקייה בה הותקן קובץ ה-JAR לתיקייה /auth/x64 או לתיקיית /auth/x32 בהתאם לגרסת ה-JRE שהורדתם

C:\Program Files\JDBC Drivers\Microsoft JDBC Driver 6.0 for SQL Server\sqljdbc_6.0\enu\auth\x64

- נאשר וכעת נוכל להריץ

דוגמת קוד

- **אתחול**

```
// Create a variable for the connection string.
String connectionUrl ="jdbc:sqlserver://localhost:1433;"
      + "databaseName=DanceDB;user=UserName;password=*****";
```

אפשר לשרשר את שדות user, password
לאחר ביצוע login של המשתמש

```
// Declare the JDBC objects.
Connection con = null;
Statement stmt = null;
ResultSet rs = null;
```

- **ניסיון התחברות**

```
try {
    // Establish the connection.
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
    con = DriverManager.getConnection(connectionUrl);
catch (ClassNotFoundException e) {
      e.printStackTrace();
}
```

דוגמת קוד

```
try {
    // Create and execute an SQL statement that returns some data.
    String SQL = "SELECT * FROM DanceDb.Dancer";
    stmt = con.createStatement();
    rs = stmt.executeQuery(SQL);

   // Iterate through the data in the result set and display it.
    while (rs.next()) {
        int id = rs.getString("DancerId");
        String name = rs.getString("FirstName") +
                                rs.getString("LastName");
        // Do something with current row
        ...
    }
 }
catch (SQLException e) {
        e.printStackTrace();
}
```

דוגמת קוד

```
finally {
   if (rs != null) try { rs.close(); } catch(Exception e) {}
   if (stmt != null) try {stmt.close(); } catch(Exception e) {}
   if (con != null) try {con.close(); } catch(Exception e) {}
 }
```

דוגמת קוד

```
ResultSetMetaData metaData = rs.getMetaData();

  for(int i = 1;  i <= metaData.getColumnCount();  i++){
        System.out.println(
              "Column Name:"+metaData.getColumnName(i)+"-"+
              "Type Name:"+metaData.getColumnTypeName(i)+"-"+
              "Type Num:"+metaData.getColumnType(i));
  }
```

- **Interface ResultSetMetaData**
  - מספק מידע על מבנה השאילתה: שמות העמודות והסוג שלהן

אובייקט ה-Statement

- **execute()** method
  - משמשת להרצת פקודות שלא מחזירות ערך כגון create table / alter table וכד'
- **executeUpdate()** method
  - משמשת לביצוע פקודות הוספה, עדכון ומחיקה
  - המתודה מחזירה את מספר הרשומות שהושפעו מהרצת הפקודה
- **executeQuery()** method
  - משמשת לביצוע פקודות המחזירות ResultSet כגון פקודות Select

ביצוע שינויים ב-DataBase

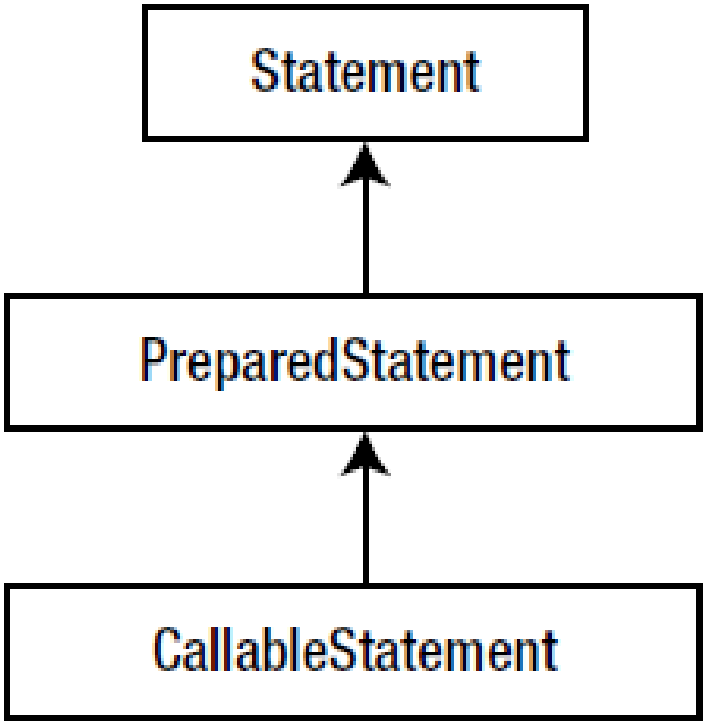
- המתודה executeUpdate
  - מופעלת באמצעות אובייקט מטיפוס Statement.
  - מאפשרת ביצוע כל סוגי השינויים ב-DataBase:
    - insert
    - update
    - delete
  - מחזירה int המציין את מספר הרשומות שהושפעו מהפעולה.

- דוגמא

```
stmt = connection.createStatement();
stmt.executeUpdate("Insert into StylesPerTeacher values (1, 1234)");
```

אובייקט ה-Statement

- סוגי Statement נוספים



Prepared Statement

- Interface PreparedStatement
  - מאפשר ליצור שאילתות עם פרמטרים
  - השאילתה נשמרת מקומפלת
  - נוצרת באמצעות המתודה prepareStatement("SQL") של האובייקט connection
  - מכיל מתודות setXXX(paramNo, value) לכל טיפוס משתנה

Prepared Statement

```
PreparedStatement prepared =
  con.prepareStatement("Select * From Teacher Where MainStyleId = ?");

prepared.setInt(1, 1);
rs = prepared.executeQuery();
if(rs != null)
    while(rs.next())
        System.out.println(rs.getInt (1) + " - " + rs.getString(2) + " - " + rs.getString(3));

prepared.setDouble(1, 2);
rs = prepared.executeQuery();
if(rs != null)
    while(rs.next())
        System.out.println(rs.getInt (1) + " - " + rs.getString(2) + " - " + rs.getString(3));
```

Prepared Statement

- אפשר עם מספר פרמטרים ועם פעולת עדכון

```
PreparedStatement prepared = con.prepareStatement( "UPDATE Act SET
TeacherInCharge = ? WHERE ShowNum = ? AND ActNum = ?");
```

```
prepared.setInt(1, 1234);
prepared.setInt(2, 16);
prepared.setInt(3, 3);

prepared.executeUpdate();
```

– יש גם פקודות setString(), setDouble() וכד'

Callable Statement

- בשבוע שעבר כתבנו פרוצדורה שמקבלת שני מזהי מורים ומספר קורס, ומחליפה ביניהם את המפגשים. נעדכן את הפרוצדורה כך שאם מזהי המורים אינם תקינים נחזיר ערך שגיאה: 101 מורה ראשון לא קיים 102 מורה שני לא קיים. כמו כן אם אחד המורים לא שובץ כלל לקורס נחזיר קוד שגיאה 103. לאחר מכן נפעיל את הפרוצדורה מתוך Java

```
DROP PROCEDURE Exchange_Teacher;

Create Proc Exchange_Teacher
@TeacherId1 int = null,
@TeacherId2 int = null,
@courseNo int
AS Begin

     SET NOCOUNT on;

     IF @TeacherId1 IS NULL OR not exists (
              select teacherId
              from Teacher
              where teacherId = @TeacherId1
          )
          return 101
```

Callable Statement

```
IF @TeacherId2 IS NULL OR not exists (
        select teacherId
        from Teacher
        where teacherId = @TeacherId2
    )
    return 102

if not exists (
        select *
        from Meeting
        where teacherId = @TeacherId1
            and courseId = @CourseNo
    )
    return 103

if not exists (
        select *
        from Meeting
        where teacherId = @TeacherId2
            and courseId = @CourseNo
    )
    return 103
```

Callable Statement

```
DECLARE @TeacherNum as INT;
DECLARE @Count as INT;

SET @Count = 0;

DECLARE @MeetingCursor as CURSOR;

SET @MeetingCursor =  CURSOR FOR
     SELECT TeacherID
     FROM Meeting
     WHERE CourseId = @courseNo
     AND TeacherId IN (@TeacherId1,@TeacherId2)

OPEN @MeetingCursor

FETCH NEXT FROM @MeetingCursor INTO @TeacherNum;
```

Callable Statement

```
    WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @Count = @Count + 1
        IF @TeacherNum = @TeacherId1
            UPDATE Meeting SET TeacherId = @TeacherId2
            WHERE CURRENT OF @MeetingCursor;
        ELSE
            UPDATE Meeting SET TeacherId = @TeacherId1
            WHERE CURRENT OF @MeetingCursor;

        FETCH NEXT FROM @MeetingCursor INTO @TeacherNum;
    END
    RETURN @Count
END
```

Callable Statement

```
System.out.println("Call Exchange_Teacher Procedure");
CallableStatement  cstmt =
con.prepareCall("{? =call Exchange_Teacher(?,?,?)}");
cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
cstmt.setInt(2, 11111);
cstmt.setInt(3, 22222);
cstmt.setInt(4, 1);

int result = 0;
cstmt.execute();
result = cstmt.getInt(1);
// Handle any errors that may have occurred.
if (result==101)
        System.out.println("First teacher does not exist");
else if (result==102)
        System.out.println("Second teacher does not exist");
else if (result==103)
        System.out.println("First or Second teacher (or both) "
                + "do not teach in the given course");
else
        System.out.println(result + " meetings were changed!");
```

Callable Statement

- כאשר הפרוצדורה משתמשת בפקודה raiserror() ניתן לתפוס אותה בתכנית כחריגה מסוג SQLException

```
catch (SQLException e) {
    System.out.println("sqlerror state:"+e.getSQLState()
        +" error code: "      + e.getErrorCode()
        +" message: "         + e.getMessage());
    e.printStackTrace();
{
```

– מומלץ להגדיר בפרוצדורה SET NOCOUNT ON

Transactions

- מימוש ב-JDBC
  - המימוש מתבצע באמצעות מתודות של הממשק Connection
  - מצב default של ה-connection – autoCommit
  - בחינת מצב ה-autoCommit – conn.getAutoCommit()
- סדר הפעולות
  - קביעת autoCommit ל-false - conn.setAutoCommit(false)
  - ביצוע סידרת פקודות SQL
  - בחינת ההצלחה לאחר הפעולה האחרונה
    - הצלחה – conn.commit()
    - כישלון – conn.rollBack()

דוגמת קוד – Transactions

```
try {
        // Disable auto commit
        con.setAutoCommit(false);

        // Do SQL()

        // Commit updates
        con.commit();
}
catch(SQLException e){
        // Rollback update
        con.rollback();
}
```

אובייקט ResultSet

- אובייקט ResultSet הוא למעשה טבלה שאליה מוכנסות הרשומות שחזרו מהשאילתה, והוא מחזיק "מצביע" לרשומה הנוכחית
  - המצביע מתקדם באמצעות פקודת next()
- יש מתודות שונות המאפשרות לשלוף את נתוני הרשומה הנוכחית
  - getInt(), getString(), getDouble(), getDate() ועוד מתודות רבות (https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSet.html)
  - ניתן לשלוף את הנתון עפ"י מיקומו בטבלה
    - String firstName=rs.getString(2);
  - ניתן לשלוף את הנתון עפ"י שם העמודה
    - String firstName=rs.getString("FirstName");

אובייקט ResultSet

```
Statement stmt = connection.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY)
```

- TYPE_FORWARD_ONLY
  - מאפשר תנועה ב-ResultSet קדימה בלבד
  - מהרשומה הראשונה ועד לאחרונה
- TYPE_SCROLL_INSENSITIVE
  - מאפשר תנועה דו כיוונית ב-ResultSet
  - שינויים אפשריים אך אינם באים לידי ביטוי אלא לאחר ריענון
- TYPE_SCROLL_SENSITIVE
  - מאפשר תנועה דו כיוונית ב-ResultSet
  - שינויים אפשריים ובאים לידי ביטוי מיד עם השינוי
- CONCUR_READ_ONLY
  - שינויים ב-ResultSet לא יבואו לידי ביטוי ב-DataBase
- CONCUR_UPDATEABLE
  - שינויים ב-ResultSet יבואו לידי ביטוי ב-DataBase באמצאות המתודה ()update של ה-ResultSet

אובייקט ResultSet

- **void beforeFirst()**
  - Sets the cursor just before the first row in the ResultSet.
- **void afterLast()**
  - Sets the cursor just after the last row of the ResultSet.
- **boolean absolute(int rowNumber)**
  - Sets the cursor to the requested row number absolutely.
- **boolean relative(int rowNumber)**
  - Sets the cursor to the requested row number relatively.
- **boolean next()**
  - Sets the cursor to the next row of the ResultSet.
- **boolean previous()**
  - Sets the cursor to the previous row of the ResultSet

Updatable ResultSets

- עדכון ה-DB באמצעות ה-ResultSet
  - ה-DB צריך לאפשר זאת
  - יש ליצור Statement עם אפשרות לתנועה דו-כיוונית ועדכון
  - פעולות אפשריות:
    - הוספה
    - שינוי
    - ביטול

Updatable ResultSets

- סדר פעולות
  - יצירת Statement המאפשר תנועה דו-כיוונית ועדכון.
    - Prepared Statement, Callable
  - יצירת ResultSet – executeQuery("SQL")
  - פעולות על ה-ResultSet:
    - הוספה
    - שינוי
    - ביטול
  - עדכון ה-DB
  - ניתן לבצע פעולות נוספות

Updatable ResultSets - המשך

- הוספת רשומה
  - התמקמות על "insert row" - moveToInsertRow()
  - עדכון השדות - updateXXX(col_no, value)
  - הוספת הרשומה ל-DB – insertRow()

```
– Statement stmt = connection.createStatement(
  ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
– ResultSet resultSet = stmt.executeQuery("SELECT * FROM my_table");
– resultSet.moveToInsertRow();
– resultSet.updateString("col_string", "new data");
– resultSet.insertRow();
– resultSet.moveToCurrentRow();          // תקף רק בהוספת רשומה
```

Updatable ResultSets

- עדכון רשומה
  - התמקמות על רשומה – move(), moveAbsolute()....
  - עדכון השדות - updateXXX(col_no, value)
  - עדכון הרשומה ב-DB – updateRow()

```
– Statement stmt = connection.createStatement(
  ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
– ResultSet resultSet = stmt.executeQuery("SELECT * FROM my_table");
– resultSet.moveAbsolute(5);
– resultSet.updateString("col_string", "new data");
– resultSet.updateRow();
```

Updatable ResultSets

- ביטול רשומה
  - התמקמות על רשומה – first(), last(), next(), previous()....
  - ביטול הרשומה ב-DB – delateRow()

```
– Statement stmt = connection.createStatement(
  ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
– ResultSet resultSet = stmt.executeQuery("SELECT * FROM my_table");
– resultSet.first();
– resultSet.deleteRow();
```

תרגיל כיתה

- כתוב תכנית Java (Console) שמתחברת למסד הנתונים ומדפיסה למסך את רשימת המורים והרקדנים (ת"ז ושם מלא) ממוינת לפי א"ב למסך
- כתוב תכנית Java (Console) שמקבלת מהמשתמש פרטי סגנון חדש, ומוסיפה את הסגנון החדש לטבלת הסגנונות
- כתוב תכנית Java (Console) שמקבלת ת"ז רקדן וכתובת חדשה (עיר, רחוב, מספר בית) ומעדכנת את כתובתו החדשה
- כתוב תכנית Java (Console) שמקבלת מהמשתמש ת"ז רקדן ומוחקת באמצעות הפרוצדורה delete_teacher את המורה

תרגיל מסכם

- כתוב view שמציג לכל קורס את מספר הנרשמים, מספר המפגשים שנערכו עד כה, מספר המורים שלימדו בקורס עד היום, ואחוז ההשתתפות הממוצע במפגשים
  - אפשר להשתמש ב-views נוספים
- כתוב פרוצדורה המקבלת מספר קורס ומחזירה את הסטטיסטיקה הנ"ל עבור הקורס יחד עם סגנון הקורס, דרגת הקושי שלו, ותאריך הפתיחה. אם לא קיים קורס כזה יוחזר קוד שגיאה 101
- כתוב תכנית Java (Console) אשר מקבלת כפרמטר מספר קורס, ומציגה את סטטיסטיקת הקורס המתוארת לעיל

קריאת רשות

ROWSET

עבודה עם RowSet

- בניגוד ל-ResultSet שהוא תמיד במצב Connected למסד הנתונים, ל-RowSet אפשרות לעבוד גם במצב Disconnected (CachedRowSet וכל המחלקות היורשות)
- יצירת RowSet:
  - "עוטף" של ResultSet במצב Connected

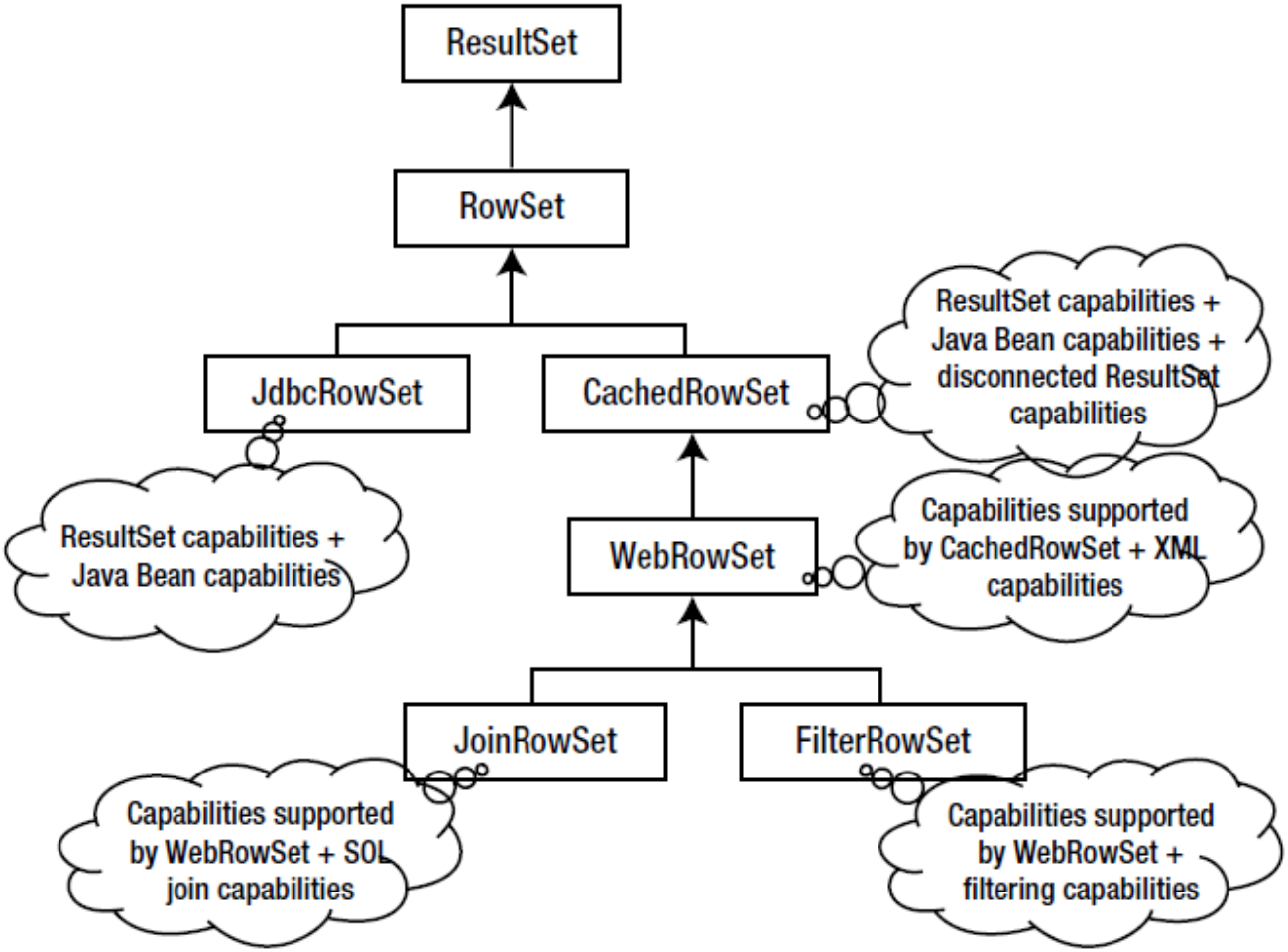
```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(select * from Teacher);
JdbcRowSet jdbcRs = new JdbcRowSetImpl(rs);
```

  - או

```
JdbcRowSet rowSet =
      RowSetProvider.newFactory().createJdbcRowSet();
rowSet.setUrl("jdbc:jtds:sqlserver://localhost:1433;”
+"databaseName=DanceDB;integratedSecurity=true;");
```

עבודה עם RowSet

- ישנם סוגים שונים של RowSet



למה להשתמש ב-RowSet

- ממשק פשוט יותר, מאפשר לעבוד מבלי לעבור על שלבי יצירת ה-connection וה-statement
- מעבר לממשק המוכר של ResultSet אפשר לבצע פעולות נוספות
  - יצירת connection למקור הנתונים
  - קריאת הנתונים לתוך ה-RowSet וניתוק הקשר
  - ביצוע שינויים ועדכונים במצב offline
  - חיבור מחדש לצורך כתיבת השינויים
  - איתור קונפליקטים מול נתוני המקור ופתרון שלהם

RowSet

Some of the most commonly used methods of the `JdbcRowSetImpl` class are as follows:

- `absolute(int row)`
- `afterLast()`
- `beforeFirst()`
- `first()`
- `deleteRow()`
- `insertRow()`
- `last()`
- `boolean isLast()`
- `boolean isAfterLast()`
- `isBeforeFirst()`
- `next()`
- `previous()`
- `moveToCurrentRow()`
- `moveToInsertRow()`
- `updateDate(int column, java.sql.Date date)`
- `updateInt(int column, int i)`
- `updateString(int column, String str)`
- `updateRow()`

JTDS

- לצורך עבודה עם RowSet ייתכן ותצטרכו להוריד את JTDS Driver
  - http://jtds.sourceforge.net/
  - https://sourceforge.net/projects/jtds/files/
- לצרף אותו כ-external jar (כמו שעשינו קודם ל-sqljdbc42.jar)
- ולצרף את תיקיית ה-SSO שלו ל- Native Library Location