Application Program Development

APD545

Instructor: Maryam Sepehrinour
Email: Maryam.Sepehrinour@SenecaPolytechnic.ca

Outcomes

Understanding of why Database Programming?

Understanding architecture of JDBC.

Understanding of JDBC Interface

Understanding the Technology of JDBC

- Connectivity
- Read operations
- Write operations

Why Java for Database Programming?

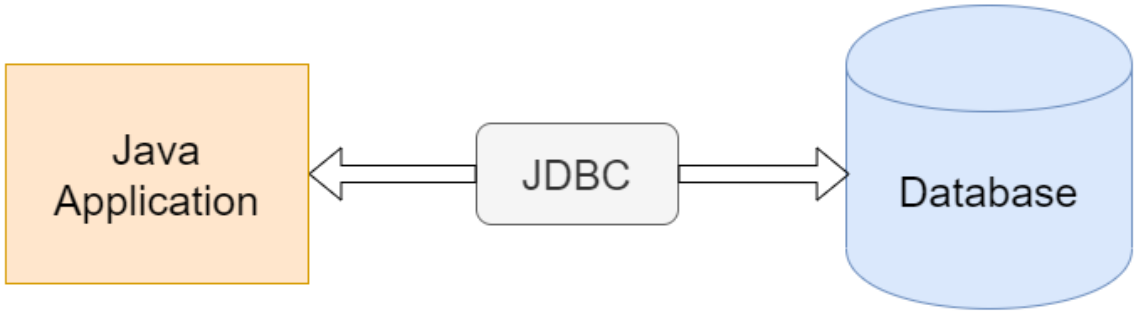
- Platform independent
- Support for accessing database systems from Java is built into Java API

What is JDBC?

Java Database Connectivity or JDBC API provides industry-standard and database-independent connectivity between the Java applications and relational database servers (relational databases, spreadsheets, and flat files).

To keep it simple, JDBC allows a Java application to connect to a relational database. The major databases are supported such as Oracle, Microsoft SQL Server, DB2 and many others.

JDBC allows a Java application to connect to a relational database



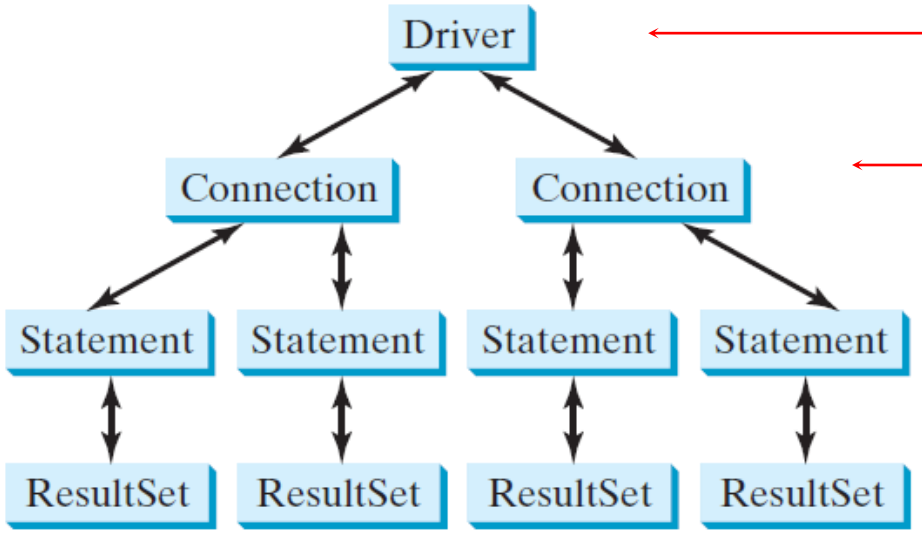
JDBC

- JDBC helps you to write Java applications that manage these three programming activities:
  - Connect to a data source, like a database
  - Send queries and update statements to the database
  - Retrieve and process the results received from the database in answer to your query

JDBC API

- JDBC API consists of two packages
  - java.sql
    - We use java.sql package API for accessing and processing data stored in a data source (usually a relational database) using the Java programming language.
  - javax.sql
    - This is the JDBC driver(there are four different types of JDBC drivers) A JDBC driver is a set of Java classes that implement the JDBC interfaces, targeting a specific database.
    - The JDBC interfaces come with standard Java, but the implementation of these interfaces is specific to the database you need to connect to. Such an implementation is called a *JDBC driver*.

One Path of Core JDBC Interfaces



Loading driver (java *library* containing classes that implement the JDBC API.)

Establishing connections

Creating and executing statements

Processing ResultSet

Installing a JDBC driver generally consists of copying the driver to your computer, then add the location of it to your classpath.

JDBC Technology

Four steps required to design apps with JDBC

- 1. Connect to the database
- 2. Create a statement
- 3. Execute the query
- 4. Look at the result set

- Close connection **// not needed if you are using try-with-resources**

Basic JDBC Connection

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class StatementExample {
	public static void main(String[] args) {
		try (Connection connection = DriverManager.getConnection("jdbc:sqlite:C:\\Users\\
m_gur\\eclipse-workspace\\Winter2023\\APD545\\Practice\\JDBCConnectionExample
\\src\\testDB.db")) {
			System.out.println(connection);
		} catch (SQLException e) {
			printSQLException(e);
		}
	}
	public static void printSQLException(SQLException ex) {
		for (Throwable e : ex) {
			if (e instanceof SQLException) {
				e.printStackTrace(System.err);
				System.err.println("SQLState: " + ((SQLException)
e).getSQLState());
				System.err.println("Error Code: " + ((SQLException)
e).getErrorCode());
				System.err.println("Message: " + e.getMessage());
				Throwable t = ex.getCause();
				while (t != null) {
					System.out.println("Cause: " + t);
					t = t.getCause();
				}
			}
		}
	}
}
```

Create Statement Object and execute

```
… … …
import java.sql.Statement;
```

```
        Statement statement = connection.createStatement();

        statement.execute("Drop Table If Exists users");

        String query = "Create Table IF NOT EXISTS Users (id INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT, username varchar(20) not null, email
        varchar(20) not null, country varchar(15), password varchar(20))";

        boolean success = statement.execute(query);

        if(success)

                System.err.println("Table is not created as the resultSet is
being                                   returned");

        else

        System.err.println("Table is created");
```

```
… … …
```

Execute statement with Insert

```
. . .
int value=0;

query = "Insert into users (username, email, country, password) Values

         ('mali','m@ali.com','Canada','1234')";

value = statement.executeUpdate(query);

if(value!=0)

         System.err.println("1 row updated in the table");


query = "Insert into users (username, email, country, password) Values
                                          ('frank','frank@f.com','Canada','5678')";

success = statement.execute(query);

if(value!=0)

         System.err.println("1 row updated in the table");
```

Execute statement with ResultSet

```
. . .

query = "Select * from users";

ResultSet rs = statement.executeQuery(query);

while(rs.next())

{

        System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3)+"
        "+rs.getString(4)+" "+rs.getString(5));

}


… … …
```

Processing Statements

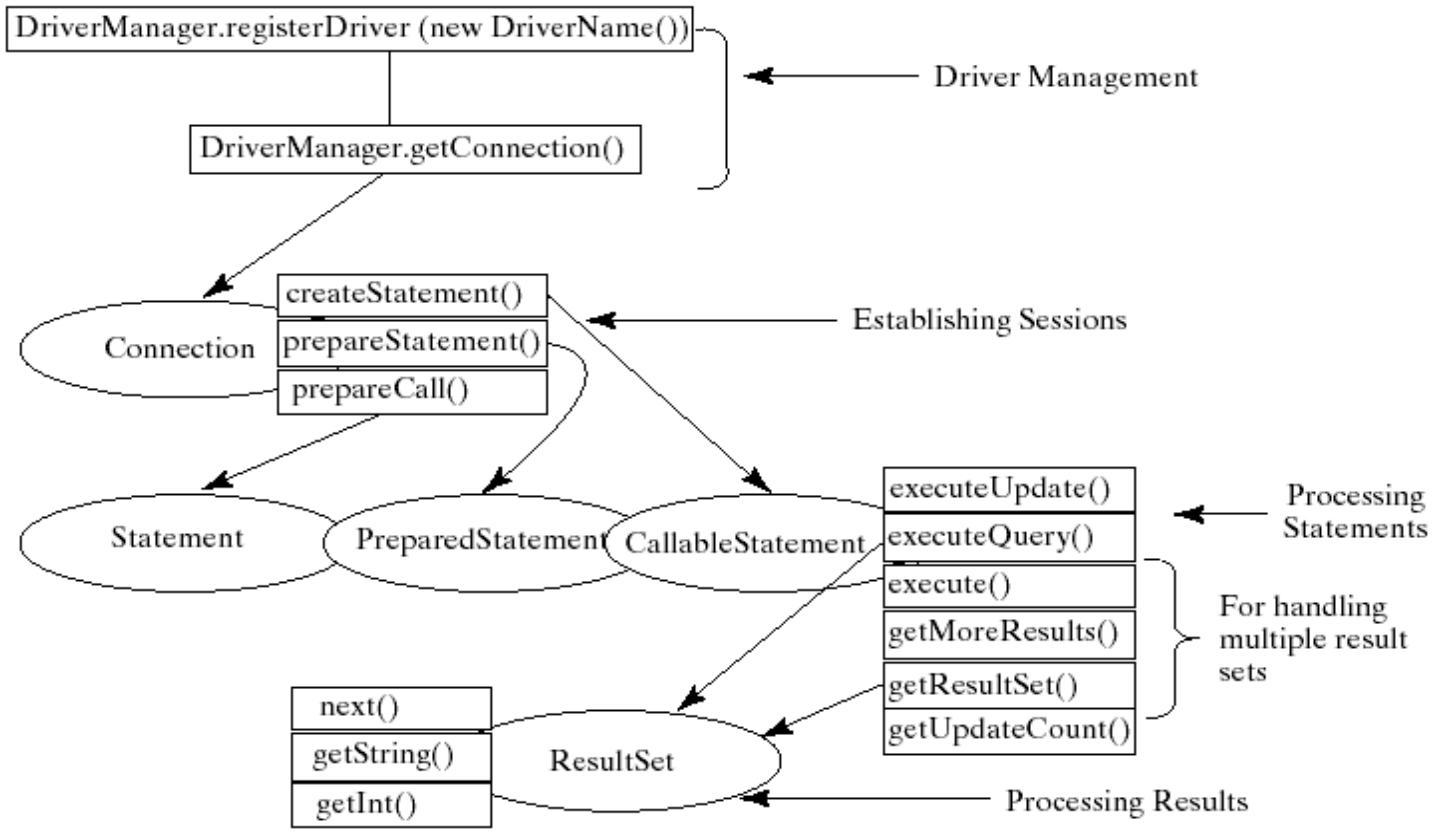
Once a connection to a particular database is established, it can be used to

- send SQL statements from your program to the database.

JDBC provides the Statement,

- PreparedStatement
- CallableStatement interfaces
- to facilitate sending statements to a database for execution and receiving execution results from the database more efficiently.

Processing Statements Diagram



Creating Table using PreparedStatement

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class PreparedStatementExample {
   public static void main(String[] args) {
      try (Connection connection = DriverManager.getConnection("jdbc:sqlite:C:\\Users\\m_gur\\eclipse-

                workspace\\Winter2023\\APD545\\Practice\\JDBCConnectionExample\\src\\testPreDB.db")) {
                System.out.println(connection);
                PreparedStatement ps = null;
                String query = "Create Table IF NOT EXISTS Books (id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, bname varchar(20)"
                                                                        + " not null, bcategory varchar(20) not null, bprice number(15), bisbn
varchar(20))";
                ps = connection.prepareStatement(query);
                boolean success = ps.execute();
                if(success)
                                System.err.println("Table is not created as the resultSet is being returned");
                else
                                System.err.println("Table is created");
                } catch (SQLException e) {
                                printSQLException(e);
       }
     }
   public static void printSQLException(SQLException ex) {
                for (Throwable e : ex) {
                                if (e instanceof SQLException) {
                                                e.printStackTrace(System.err);
                                                System.err.println("SQLState: " + ((SQLException) e).getSQLState());
                                                System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
                                                System.err.println("Message: " + e.getMessage());
                                                Throwable t = ex.getCause();
                                                while (t != null) {
                                                                System.out.println("Cause: " + t);
                                                                t = t.getCause();
                                                }
```

Insert Data using PreparedStatement

```
… … …
int value=0;
query = "Insert into books (bname, bcategory, bprice, bisbn) Values (?,?,?,?)";
ps = connection.prepareStatement(query);
ps.setString(1, "Harry Potter");
ps.setString(2, "Fantacy");
ps.setDouble(3,65.3);
ps.setString(4, "123456789");
value = ps.executeUpdate();

if(value!=0)
System.err.println("1 row updated in the table");
ps.setString(1, "Dog Man");
ps.setString(2, "Fantacy");
ps.setDouble(3,9.3);
ps.setString(4, "123456789");
value = ps.executeUpdate();
if(value!=0)
System.err.println("1 row updated in the table");
… … …
```

Select Data using PreparedStatement

… … …

```
query = "Select * from books where bname = ?";

ps = connection.prepareStatement(query);

ps.setString(1,"Harry Potter");

ResultSet rs = ps.executeQuery();

while(rs.next())

{

System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3)+"
"+rs.getString(4)+" "+rs.getString(5));

}
```

… … …

Thank you!