

PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan 10

Royana Afwani

Teknik Informatika Universitas Mataram



Materi

- Java Database Connection
- Package



JDBC

Java Database Connection

Ruang Lingkup Pembahasan



- ❑ Sistem Database
- ❑ Pengantar teknologi JDBC
- ❑ Sejarah JDBC
- ❑ Desain JDBC
- ❑ JDBC Driver
- ❑ Arsitektur aplikasi JDBC
- ❑ Petunjuk langkah penggunaan JDBC
- ❑ Retrieve data dari ResultSet

Sistem Database



- Berbagai macam jenis sistem database :
 - ▣ High performance commercial databases – eg. Oracle, DB2, Informix, Microsoft SQL server
 - ▣ Open-source – eg. PostgreSQL, MySQL and Interbase
 - ▣ Lightweight Java databases – eg. Cloudscape, InstantDB and Pointbase.
 - ▣ Desktop databases – eg. Paradox and Access.

- Need to choose and install database first.

JDBC

- JDBC adalah Application Programming Interface (API) yang menyediakan fungsi-fungsi dasar untuk akses data.
- JDBC API terdiri atas sejumlah class dan interface yang dapat digunakan untuk menulis aplikasi database dengan menggunakan Java.
- Class dan Interface JDBC terdapat pada package `java.sql`
- Contoh standard API JDBC:
 - ▣ Membuat koneksi ke database
 - ▣ Mengakses data dengan query
 - ▣ Membuat stored (parameterized) query
 - ▣ Mendapatkan struktur data dari result query (tabel) :
 - Menyatakan jumlah kolom
 - Mendapatkan nama kolom, dll
 - ▣ dll

JDBC Driver



- Masing-masing database server memiliki arsitektur dan sistem yang berbeda → cara komunikasi berbeda.
- Sehingga tiap database server memiliki driver sendiri.
- Untuk aplikasi Java, driver database disebut dengan JDBC Driver.
- **JDBC Driver** adalah software library yang diperlukan agar program JDBC dapat berkomunikasi dengan database tertentu

JDBC Driver



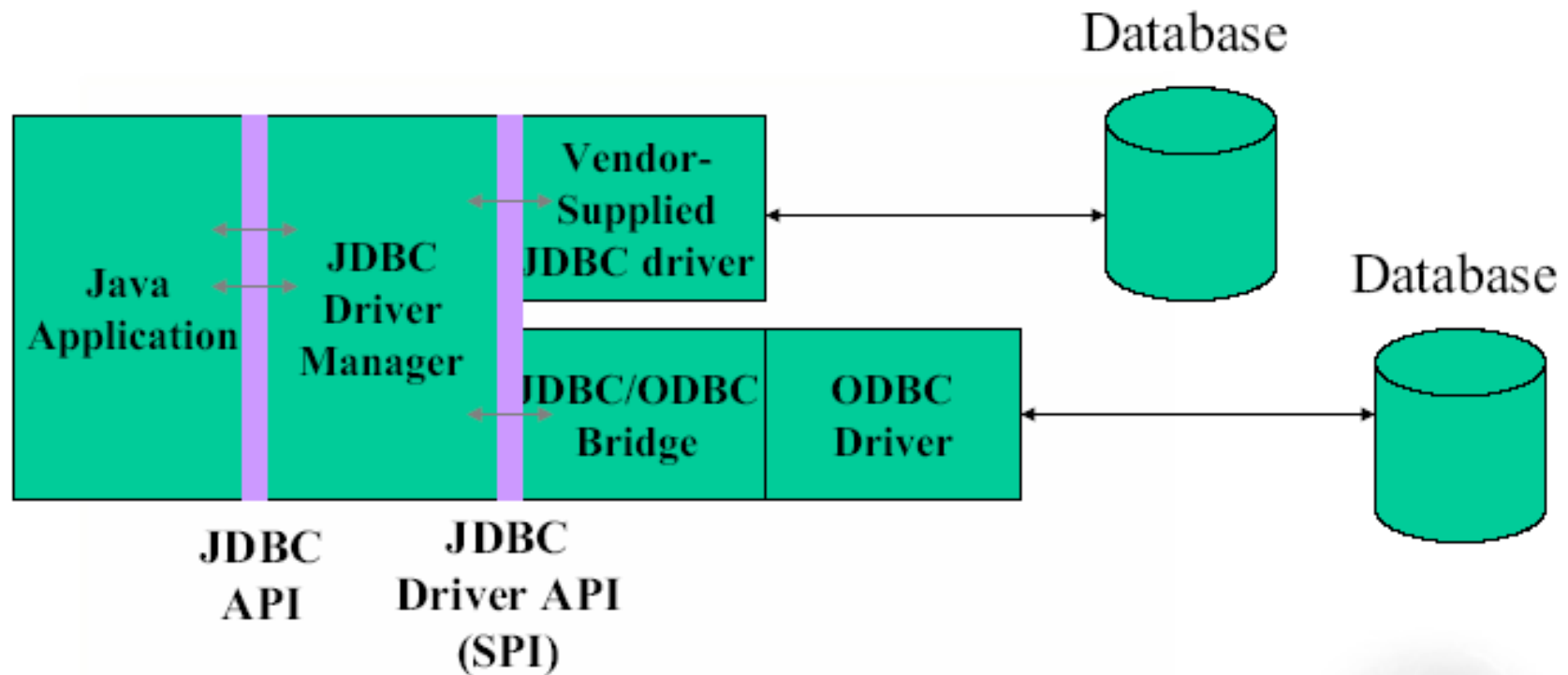
- Masing-masing driver memiliki implementasi dari spesifikasi JDBC secara berbeda.
- Perbedaan:
 - ▣ Kecepatan
 - ▣ Kestabilan
 - ▣ Fasilitas

Registrasi JDBC Driver



- Sebelum menggunakan JDBC Driver, dilakukan ***registrasi*** driver.
 - ▣ `Class.forName(String namaDriver).newInstance()`
 - ▣ `DriverManager.registerDriver(Driver namaDriver)`
 - ▣ Menggunakan properti `jdbc.drivers`

Database Communication



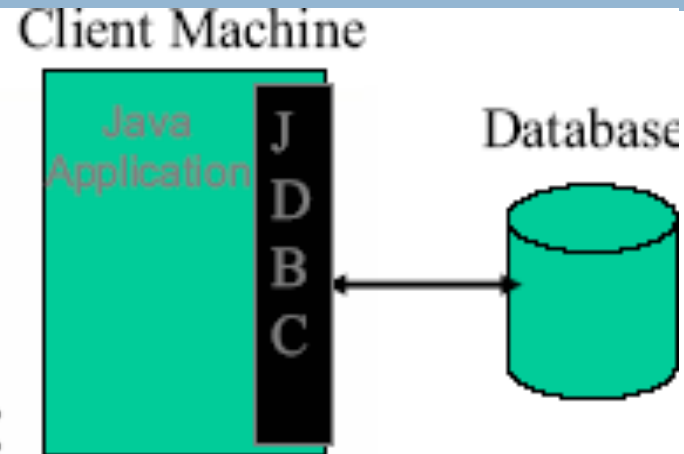
- Two Layers of JDBC API involved in communication.

Secara garis besar, teknologi JDBC melakukan:

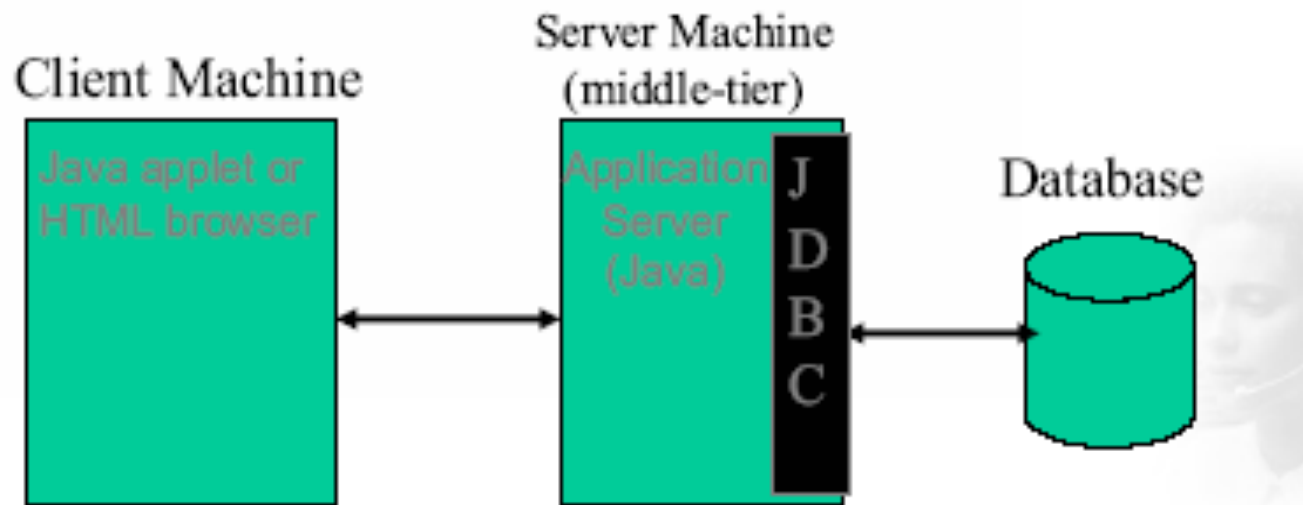
- Membangun sebuah koneksi ke sumber data (data source).
- Mengirim statement ke sumber data.
- Memproses hasil dari statement tersebut.

Typical JDBC Use

- Two-tier architecture:



- Three-tier architecture:

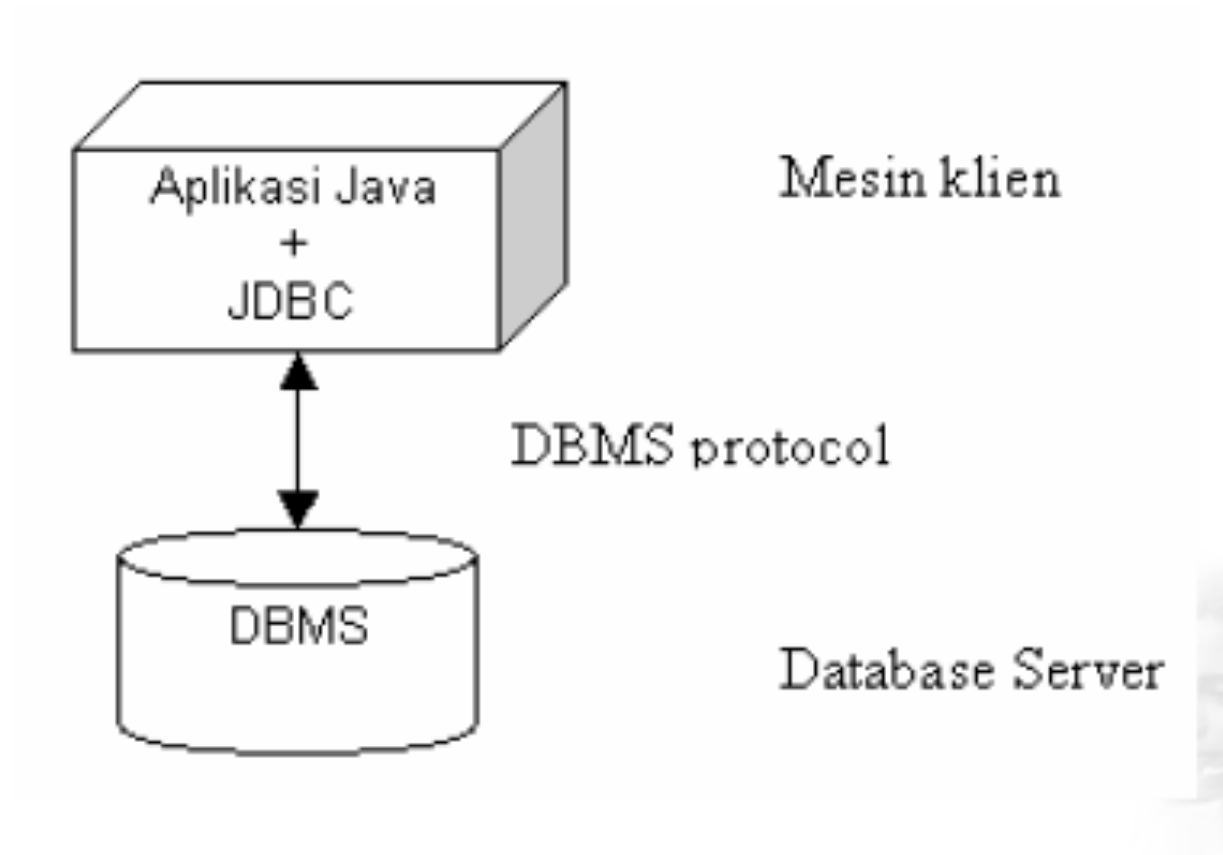


Model 2-tier



- Sebuah applet atau aplikasi java berbicara langsung ke database.
- Sebuah perintah atau statement dari user dikirim ke database dan hasil dari statement dikirim balik ke user.
- Database dapat terletak pada mesin yang sama atau berbeda dengan klien.
- Jika letak database berbeda dengan mesin klien maka disebut dengan client/server. Mesin user → client dan mesin dimana database berada → server.

Arsitektur 2-tier

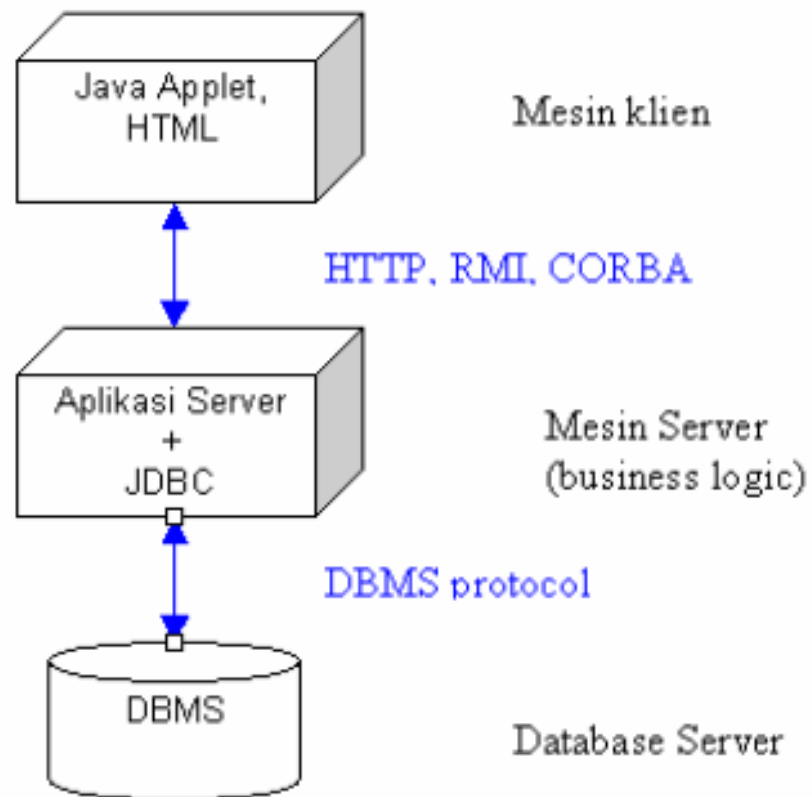


Model 3-tier



- ❑ User mengirimkan perintah ke sebuah middle tier.
- ❑ Selanjutnya middle tier mengirimkan perintah tersebut ke database.
- ❑ Database memproses perintah tersebut dan mengirim balik hasilnya ke middle tier.
- ❑ Kemudian middle tier mengirimkannya ke user.
- ❑ Keuntungan: mempermudah aplikasi untuk dideploy dan meningkatkan performansi.

Arsitektur 3-tier



JDBC Data Types

JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT DOUBLE	double
BINARY VARBINARY LONGVARBINARY	byte[]
CHAR VARCHAR LONGVARCHAR	String

JDBC Type	Java Type
NUMERIC DECIMAL	BigDecimal
DATE	java.sql.Date
TIME TIMESTAMP	java.sql.Timestamp
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

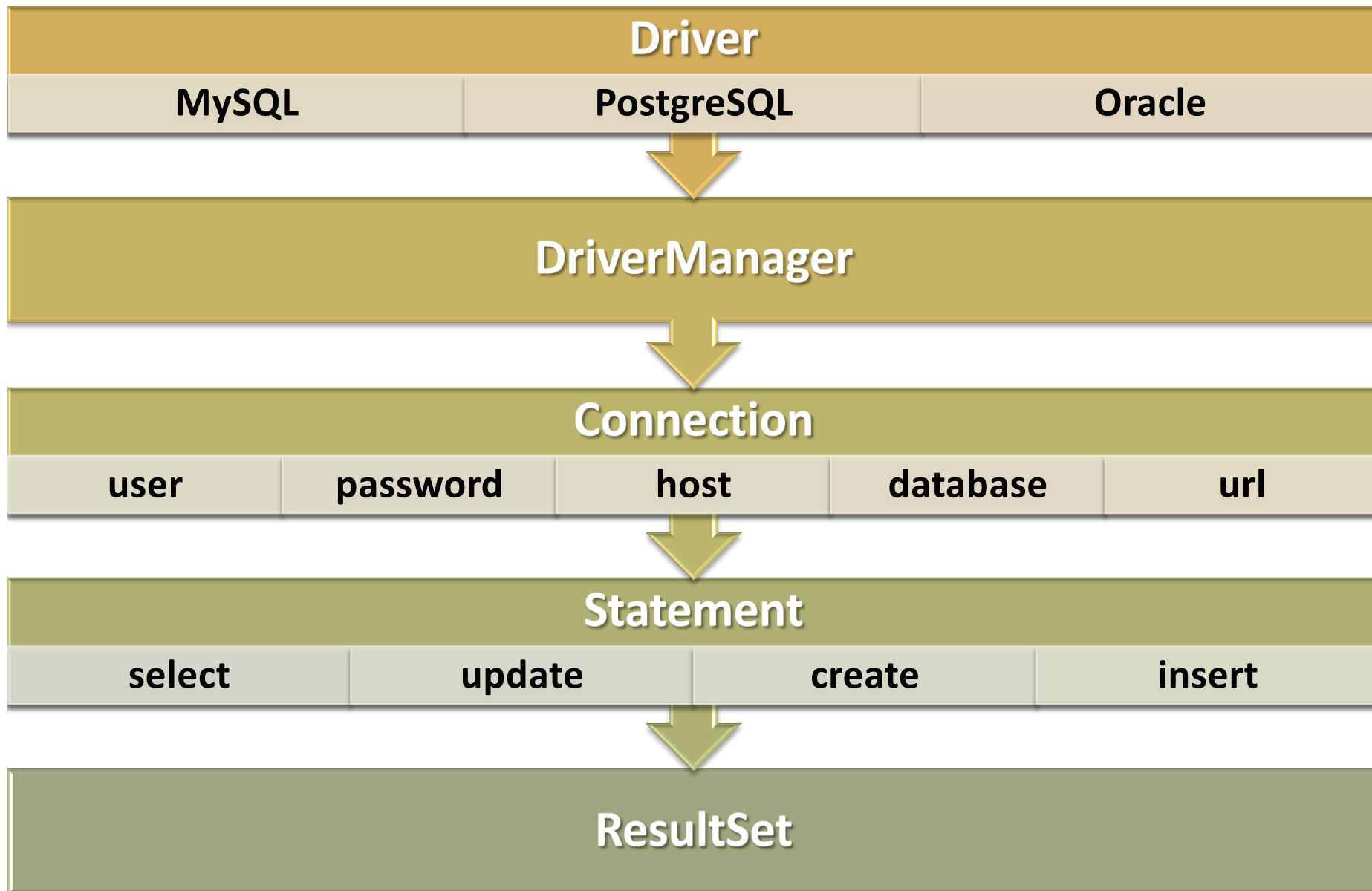
*SQL3 data type supported in JDBC 2.0

Langkah Penggunaan JDBC

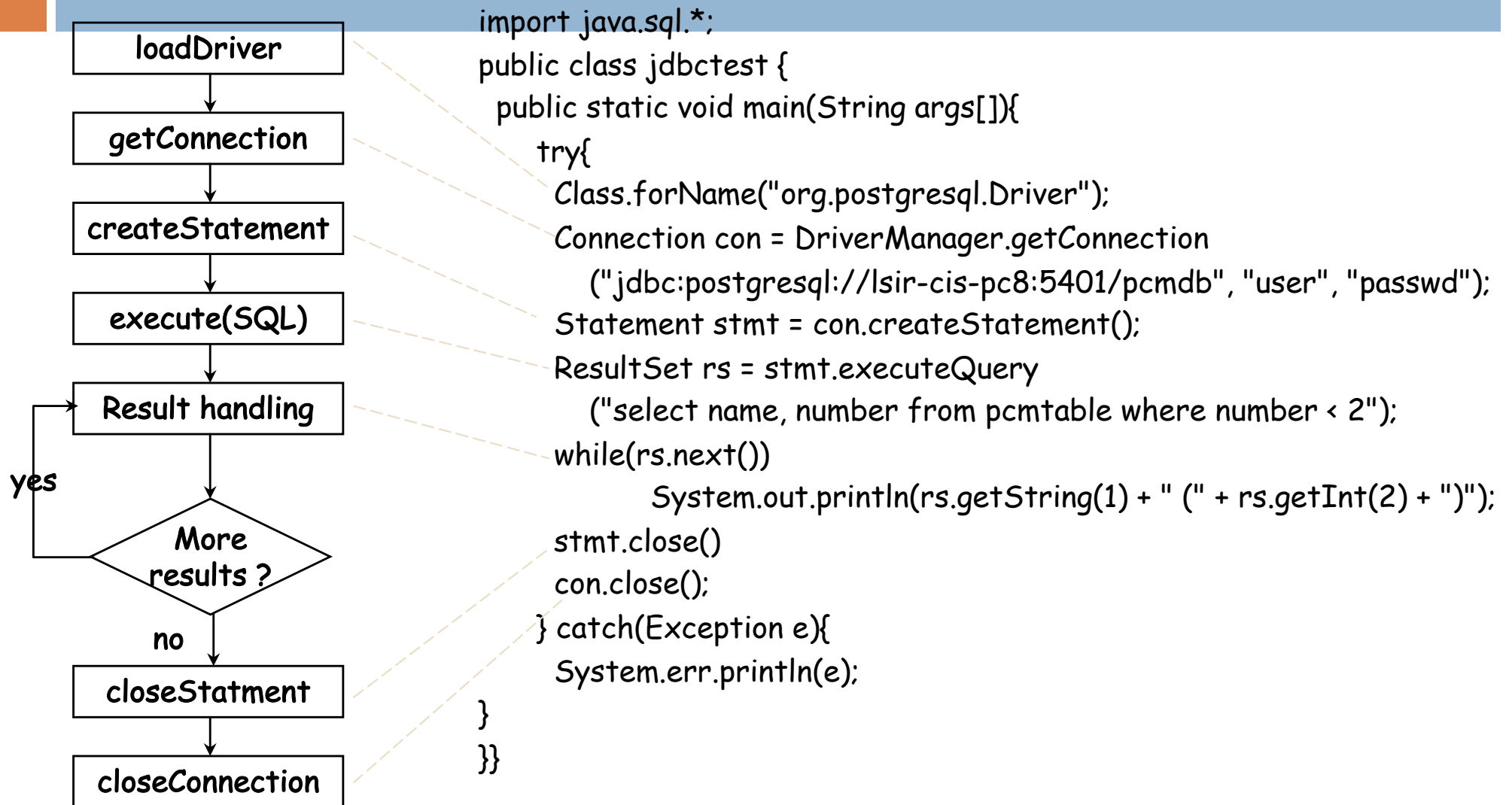


1. Load driver JDBC
2. Definisikan URL database
3. Membuat dan melakukan koneksi
4. Membuat obyek statement
5. Mengeksekusi query
6. Memproses result
7. Menutup koneksi

Tahapan Akses Database dengan JDBC




A Simple JDBC application



1. JDBC : Load Driver



- Driver :
 - ▣ library yang digunakan untuk berkomunikasi dengan database server
 - ▣ Dengan menggunakan driver, program Java yang menggunakan API JDBC dapat berinteraksi dan dapat dimengerti oleh database server.
- Untuk database yang berbeda dibutuhkan driver yang berbeda.



```
try {  
    Class.forName("sun.jdbc.odbc.JdbcDriver");  
}  
catch (ClassNotFoundException ex) {  
    System.err.println("Driver Error");  
    ex.printStackTrace();  
    System.exit(1);  
}
```

- Contoh di atas jika yang kita gunakan adalah JDBC-ODBC driver.
- Dokumentasi driver anda akan memberikan nama class yang digunakan.



Nama Driver database:

□ JDBC-ODBC :

sun.jdbc.odbc.JdbcOdbcDriver

□ Oracle :

oracle.jdbc.driver.OracleDriver

□ Sybase :

com.sybase.jdbc.SybDriver

□ MySQL:

com.mysql.jdbc.Driver

□ PostgreSQL:

org.postgresql.Driver

□ Microsoft SQLServer 2000 :

com.microsoft.jdbc.sqlserver.SQLServerDriver

2. JDBC : Definiskan koneksi URL

- Menspesifikasikan lokasi database server
- Gunakan dokumentasi driver
- Untuk penggunaan JDBC di applet:
 - ▣ database server harus berada pada node yang sama dengan letak applet.
 - ▣ Menggunakan proxy server yang me “reroute” request database ke actual server.
- Contoh:

```
String host = "dbhost.yourcompany.com";
String dbName = "someName";
int port = 1234;
String oracleURL = "jdbc:oracle:thin:@" + host +
    ":" + port + ":" + dbName;
String sybaseURL = "jdbc:sybase:Tds:" + host +
    ":" + port + ":" +
    "?SERVICENAME=" + dbName;
```




Nama URL database:

□ JDBC-ODBC :

jdbc:odbc:nama_database

□ Oracle :

jdbc:oracle:thin:@nama_host:1521:namaDB

□ MySQL:

jdbc:mysql://nama_host:3306/namaDB

□ PostgreSQL:

jdbc:postgresql://nama_host:5432/namaDB

□ Microsoft SQLServer 2000 :

jdbc:microsoft:sqlserver://nama_host:1433;DatabaseName=namaDB

3. JDBC : Membuat Koneksi

- Cara : memanggil method getConnection dari class DriverManager dengan melewati URL (hasil langkah dua) sebagai argumen.
- getConnection akan melempar SQLException
- Contoh:

```
String username = "jay_debesees";  
String password = "secret";  
Connection connection =  
    DriverManager.getConnection(oracleURL,  
                                username,  
                                password);
```

4. JDBC : Membuat Obyek Statement

- ❑ Object Statement digunakan untuk mengirim query dan perintah ke database.
- ❑ Object Statement dibuat dengan cara bekerjasama dengan class Connection.
- ❑ Cara: memanggil method `createStatement()` dari obyek Connection.
- ❑ Contoh:

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

5. JDBC : Mengeksekusi Query

- Memanfaatkan object Statement untuk memproses query.
- Cara: memanggil method `executeQuery()` dari object Statement. → memberikan return value bertipe `ResultSet`
- Returns: `ResultSet`.

```
String sql="select col1, col2, col3 from sometable";  
ResultSet rs=statement.executeQuery(sql);
```

- Note : Untuk memodifikasi database, gunakan `statement.executeUpdate(sql);` yang mendukung string sql UPDATE, INSERT INTO, DELETE

ResultSet

- ResultSet memberikan bermacam2 method getXxx dengan parameter indek kolom atau nama kolom dan mengembalikan data.
- Method lain object ResultSet:
 - findColumn()
mendapatkan index (integer value) berdasarkan nama kolom.
Kolom pertama mempunyai index 1 bukan 0.
 - getMetaData()
retrieve informasi mengenai ResultSet, returns object ResultSetMetaData.
 - wasNull()
Mengetahui apakah getXxx() menghasilkan SQL null.

6. JDBC : Memproses result

- Dengan menggunakan method `next()` pada object `ResultSet` → mendapatkan results per satu baris.

- Contoh:

```
String nrp;
```

```
String nama;
```

```
while (rs.next()){  
    nrp=rs.getString(1);  
    nama=rs.getString(2);  
    System.out.println("NRP : " +nrp);  
    System.out.println("NAMA : " +nama);  
    System.out.println("-----");  
}
```

- Kolom pertama mempunyai index 1 bukan 0.
- Object `ResultSet` otomatis akan ditutup bila ada object `ResultSet` baru.

7. JDBC : Menutup Koneksi

- Harus didefinisikan secara eksplisit.

connection.close();

- Karena membuka koneksi adalah mahal, maka penundaan langkah terakhir ini hanya jika masih ada operasi database yang dilakukan.

Contoh

```
import java.sql.*;

public class TestDB {
    public static void main(String[] args) {

        // Use driver from Connect SW.
        String driver = "connect.microsoft.MicrosoftDriver";
        try {
            Class.forName(driver);
            String url = "jdbc:ff-microsoft://" +      // FastForward
                        "dbtest.apl.jhu.edu:1433/" + // Host:port
                        "pubs";                       // Database name
            String user = "sa", password="";

            Connection connection =
                DriverManager.getConnection(url, user, password);
            Statement statement = connection.createStatement();
            String query =
                "SELECT col1, col2, col3 FROM testDB";

            // Execute query and save results.
            ResultSet results = statement.executeQuery(query);
```

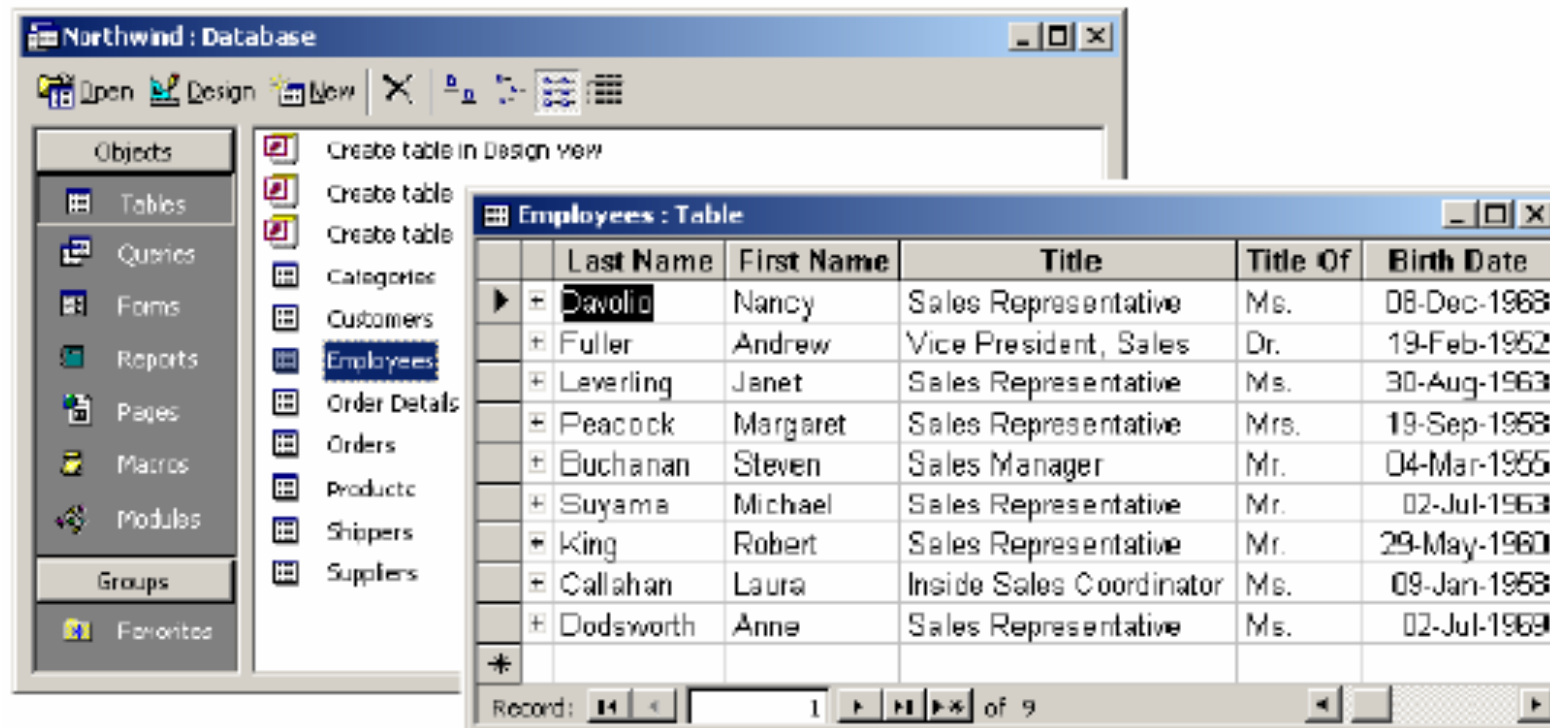


```
// Print column names.
String divider = "-----+-----+-----";
System.out.println("Col1 | Col2 | Col3\n" + divider);

// Print results
while(results.next()) {
    System.out.println
        (pad(results.getString(1), 4) + " | " +
         pad(results.getString(2), 4) + " | " +
         results.getString(3) + "\n" + divider);
}
connection.close();
} catch(ClassNotFoundException cnfe) {
    System.out.println("No such class: " + driver);
} catch(SQLException se) {
    System.out.println("SQLException: " + se);
}
}
```

Menggunakan Microsoft Access via ODBC(1)

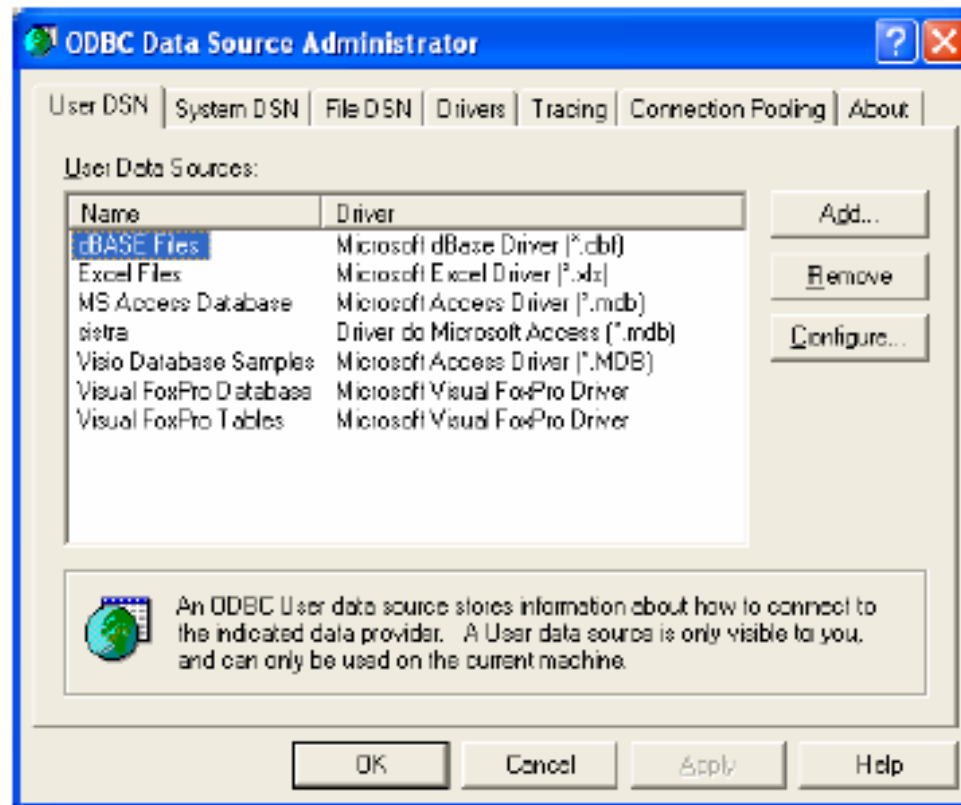
Northwind sample database



- Northwind.mdb located in C:\Program Files\Microsoft Office\Office\Samples
- <http://office.microsoft.com/downloads/2000/Nwind2k.aspx>

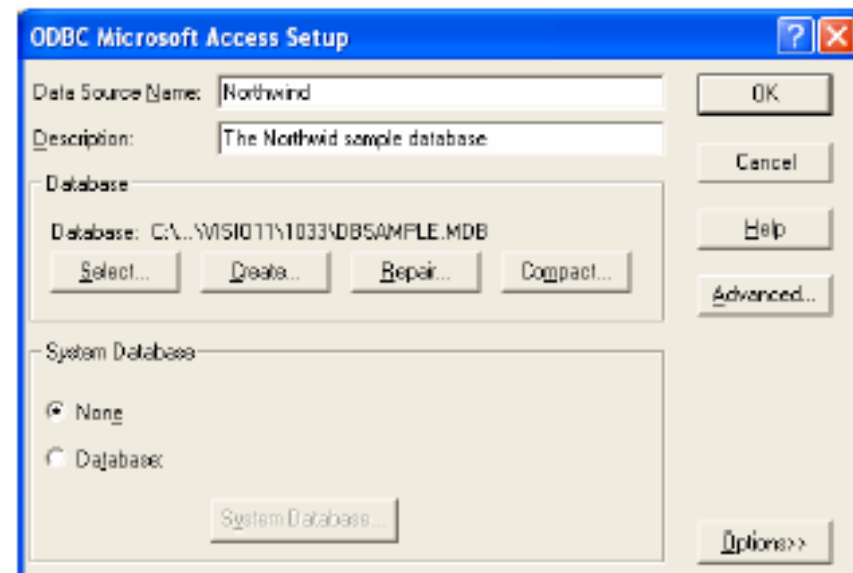
Menggunakan Microsoft Access via ODBC (2)

- ❑ Click Start, Settings, Control Panel, Administrative Tools, Data Sources(ODBC), System DSN, dan pilih Add



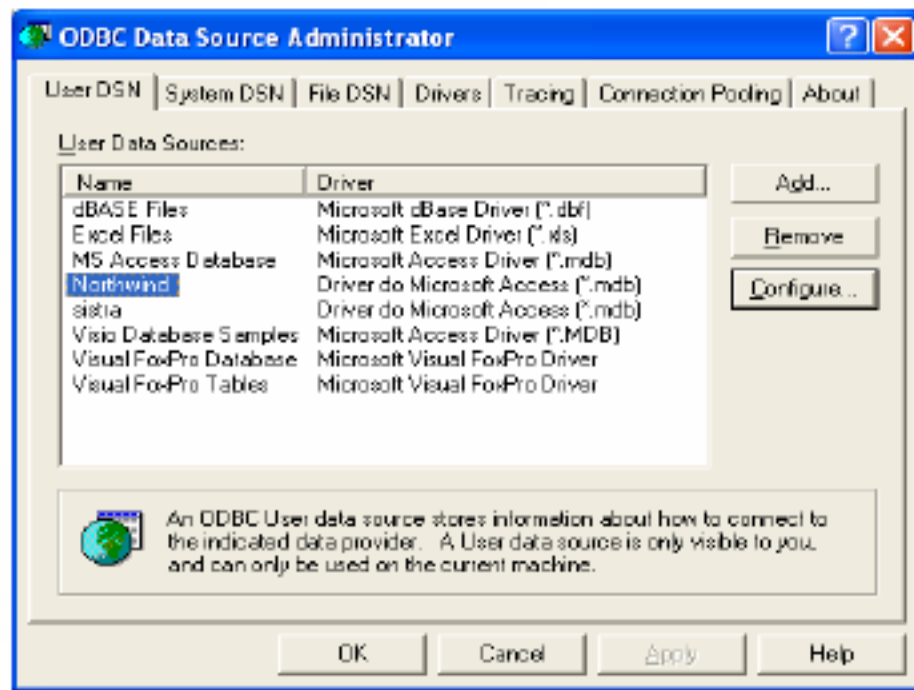
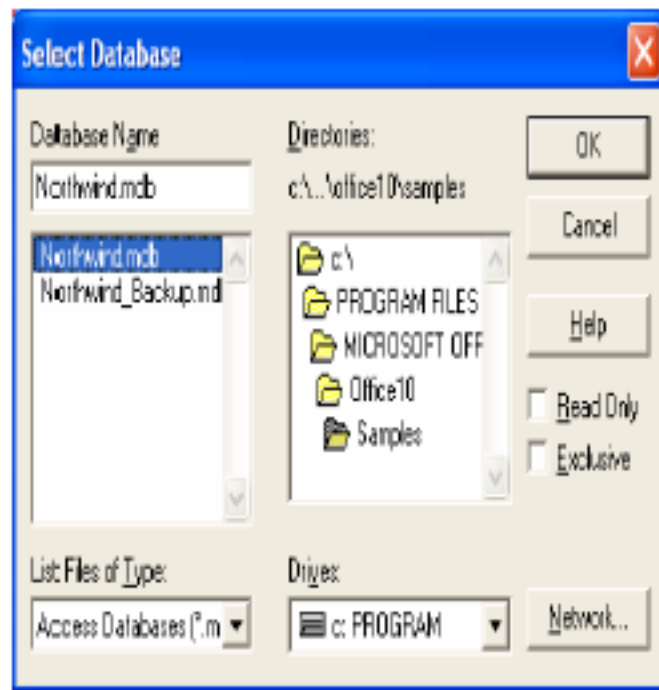
Menggunakan Microsoft Access via ODBC (3)

- ❑ Memilih driver Microsoft Access, Finish, ketikkan nama Data Source Name dan tekan Select untuk memilih nama dan lokasi database



Menggunakan Microsoft Access via ODBC (4)

- Navigasi pada directory Samples ms office, pilih Northwind.mdb, tekan OK dan lanjutkan tekan OK pada window II



Menggunakan Microsoft Access via ODBC (5)

- **Gunakan `sun.jdbc.JdbcOdbcDriver` sebagai nama class dari JDBC driver**
`Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`
- **Gunakan “`jdbc:odbc:Northwind`” sebagai alamat database, dan gunakan empty string pada username dan password**

```
Connection con=DriverManager.getConnection(jdbc:odbc:Northwind,"","");
```

Simple Northwind Test (1)

```
import java.sql.*;
public class DbTest {
    private Connection con;
    DbTest(){ con=null; }
    public ResultSet dbOpen() {
        String dbname="jdbc:odbc:Northwind";
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection(dbname,"","");
            if (con==null)
                System.err.println("Koneksi ke Database gagal");
            else
                System.err.println("Koneksi ke Database Berhasil");
        }
        catch(ClassNotFoundException ex) {
            System.err.println("Driver Error");
            ex.printStackTrace();
            System.exit(1);
        }
        catch(SQLException ex) {
            System.err.println("Tidak Berhasil Koneksi dengan Sistra");
            System.exit(1);
        }
    }
}
```



Simple Northwind Test (2)

```
ResultSet rs=null;
Statement st;
    try {
        System.out.println("Employees\n" +
                           "=====");
        st=con.createStatement();
        rs=st.executeQuery("SELECT * FROM employees");
    }
    catch(SQLException ex){
        ex.printStackTrace();
    }
    return(rs);
}

public void showEmployee (ResultSet rs) throws SQLException {
    while (rs.next()){
        System.out.print(rs.getString(3)+" ");
        System.out.println(rs.getString(2)+" ");
    }
}
```


Simple Northwind Test (3)

```
public void dbClose(){
    try {
        con.close();
    }
    catch(SQLException sqlex){
        System.err.println("Error :Koneksi
        Database tidak Bisa diputus");
    }
}

public static void main(String argv[]) throws Exception {
    DbTest app=new DbTest();
    ResultSet rs;
    rs=app.dbOpen();
    app.showEmployee(rs);
    app.dbClose();
}
}
```



Result Simple Northwind Test

```
C:\j2sdk1.4.1_01\bin\java.exe -classpath
"C:\j2sdk1.4.1_01\jre\lib\rt.jar;D:\DATA" DbTest
Employees
=====
Nancy Davolio
Andrew Fuller
Janet Leverling
Margaret Peacock
Steven Buchanan
Michael Suyama
Robert King
Laura Callahan
Anne Dodsworth
Koneksi ke Database Berhasil
Finished executing
```



Tambahan

-Package

PACKAGE

Pemrograman Orientasi Objek

Apa yang Disebut Package ?



Paket A



Paket B



Paket C



Paket D



You Package



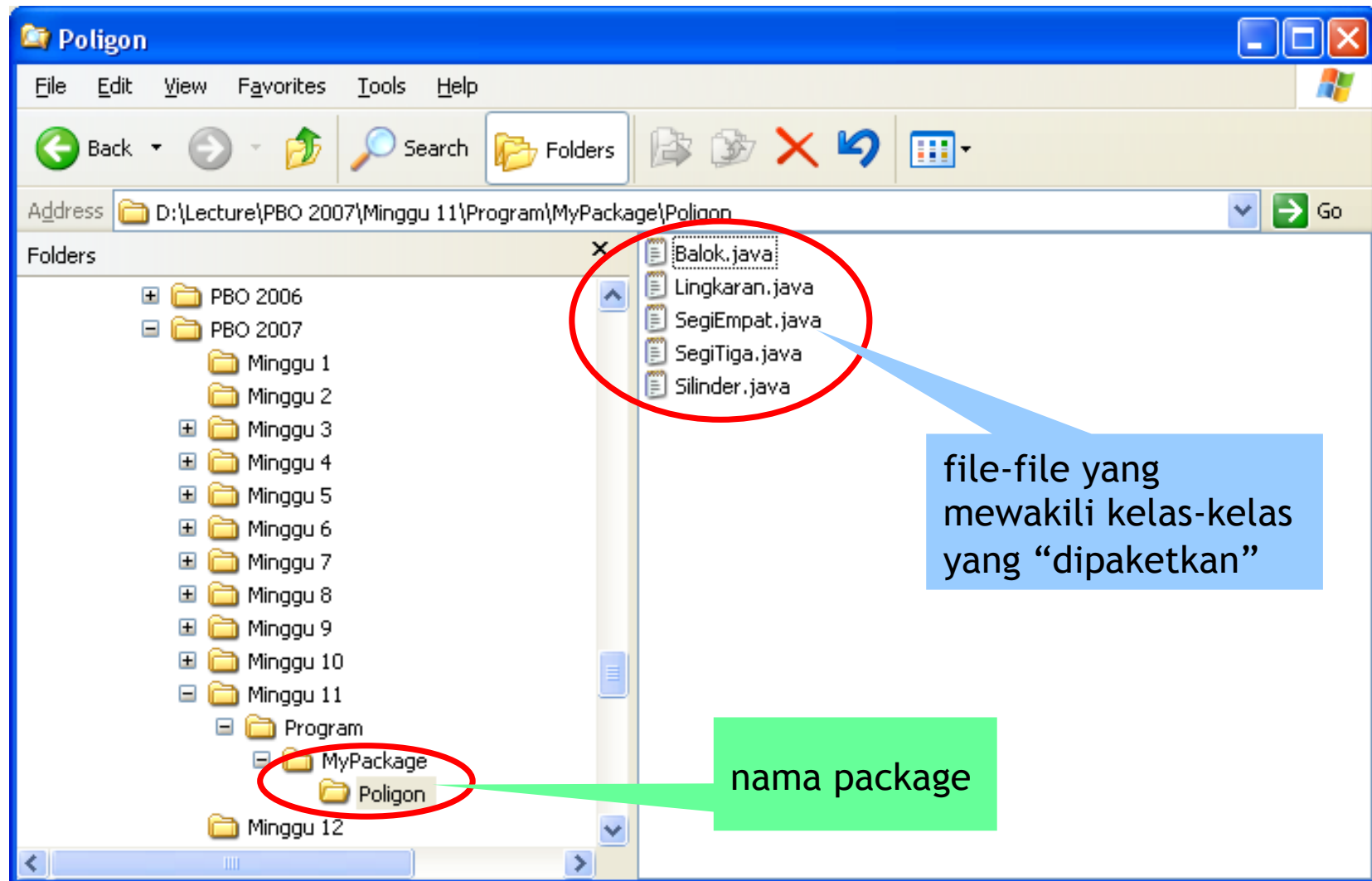
Pro Package

Apa yang Disebut Package ?



- A package is a grouping of related classes and interfaces providing access protection and name space management.
- Packages are nothing more than the way we organize files into different directories according to their functionality, usability as well as category they should belong to.

Apa yang Disebut Package ? (lanjutan)



Membuat Package

- Buat directory yang merepresentasikan tempat package yang akan dibuat.



Membuat Package (lanjutan)

- Buat kelas (atau interface) yang akan menjadi isi package dengan susunan:

```
// Deklarasi package
package namapaket;

// Deklarasi kelas
public class namakelas {
    ...
}
```

Membuat Package (lanjutan)

Deklarasi Package

- Menggunakan kata kunci **package** yang ditulis di baris pertama pada file sumber (.java).

```
package namapaket;
```

Contoh:

```
package MyPackage;
```

- Hanya boleh ada satu pernyataan package pada setiap file sumber.

Membuat Package (lanjutan)



Deklarasi Kelas

- Dinyatakan secara **public** supaya bisa diakses oleh semua kelas yang berada didalam dan diluar package yang dibuat.
- Jika ada beberapa kelas pada file sumber, hanya boleh ada satu kelas yang dinyatakan secara public, yaitu kelas yang namanya sama dengan nama file sumber.

Membuat Package (lanjutan)

- Atur variabel lingkungan CLASSPATH sehingga menunjuk directory tempat dimana package disimpan:

- Melalui Control Panel
- Melalui perintah set path di command line

```
SET CLASSPATH = .; D:\Lecture\PBO 2007\Minggu 11\Program;
```

- Kompilasi kelas (atau interface) yang menjadi isi package.

Membuat Package (lanjutan)

- Gunakan package kelas (atau interface) yang sudah dikompilasi melalui:

- ▣ Pernyataan import

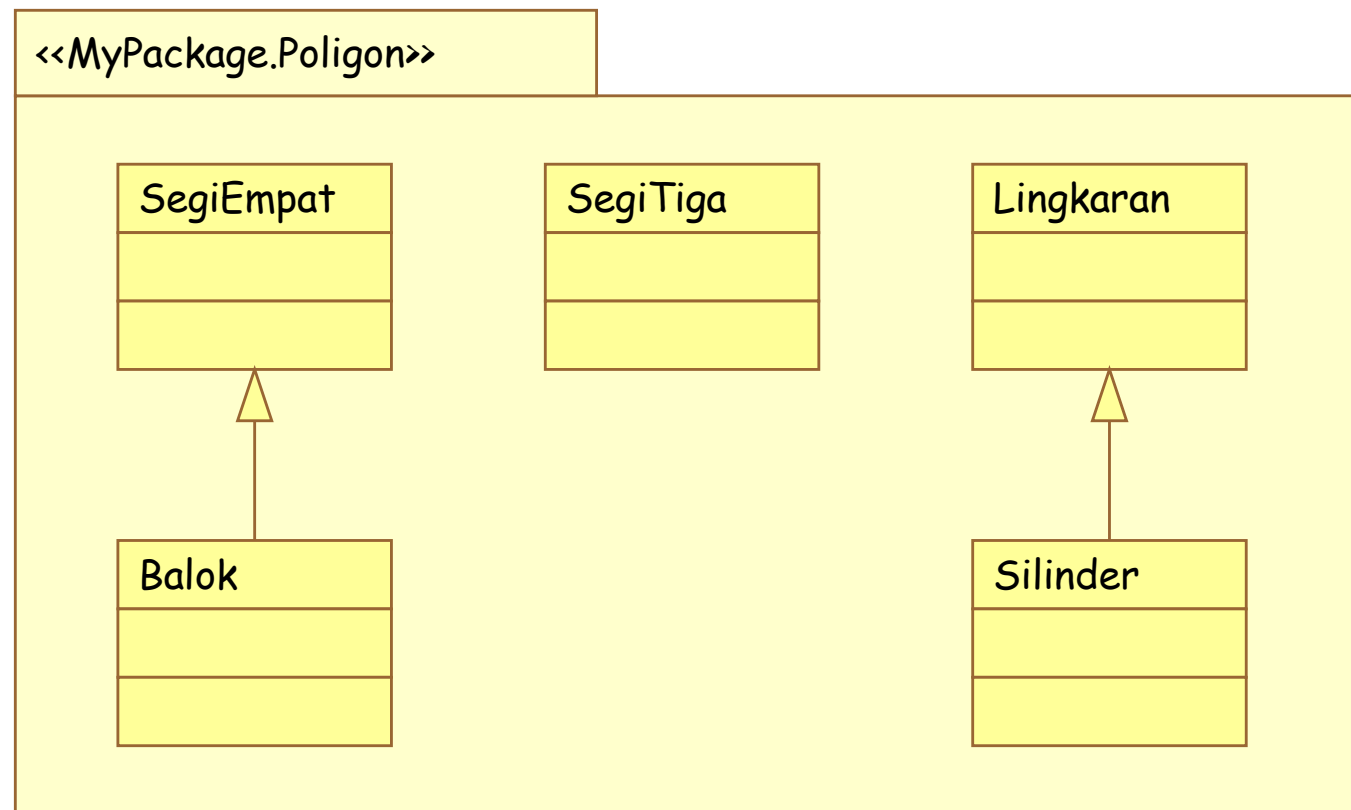
```
import MyPackage.Poligon.*;  
import MyPackage.Poligon.SegiEmpat;
```

- ▣ Nama *qualified* dari kelas (atau interface)

```
MyPackage.Poligon.SegiEmpat S;  
S = new MyPackage.Poligon.SegiEmpat(17, 8);
```

Contoh Penggunaan Package

□ Diagram Kelas:



Contoh Penggunaan Package (lanjutan)

- Nama package:
 - ▣ `MyPackage.Poligon`
- Kelas yang menjadi isi package:
 - ▣ `SegiEmpat.java`
 - ▣ `SegiTiga.java`
 - ▣ `Lingkaran.java`
 - ▣ `Balok.java`
 - ▣ `Silinder.java`
- Kelas yang menggunakan package:
 - ▣ `TestPackage.java`



TERIMA KASIH