Lecture 05
JDBC Database Access

Chapter 14: JDBC- Java Database Connectivity
(4 slots)

References:

- **Java-Tutorials/tutorial-2015/jdbc/index.html**
- **Java Documentation, the java.sql package**

Why should you study this lecture?

- In almost all large applications. Data are organized and stored in databases which are managed by database management systems (DBMS) such as MS Access, MS SQL Server, Oracle, My SQL,…
- Do you want to create Java applications which can connect to DBMSs?
- Database programming is a skill which can not be missed for programmers.

Objectives

- Introduction to databases
- Relational Database Overview
- JDBC and JDBC Drivers
- Steps to develop a JDBC application.
- Demonstrations.

1- Database and DBMS

- **Database** is a collection of related data which are stored in secondary mass storage and are used by some processes concurrently.
- Databases are organized in some ways in order to reduce redundancies.
- **DBMS**: Database management system is a software which manages some databases. It supports ways to users/processes for creating, updating, manipulating on databases and security mechanisms are supported also.
- DBMS libraries (C/C++ codes are usually used) support APIs for user programs to manipulate databases.

2- Relational Database Overview

- Common databases are designed and implemented based on relational algebra (set theory).
- Relational database is one that presents information in tables with rows and columns.
- A table is referred to as a relation in the sense that it is a collection of objects of the same type (rows).
- A Relational Database Management System (RDBMS)- such as MS Access, MS SQL Server, Oracle- handles the way data is stored, maintained, and retrieved.

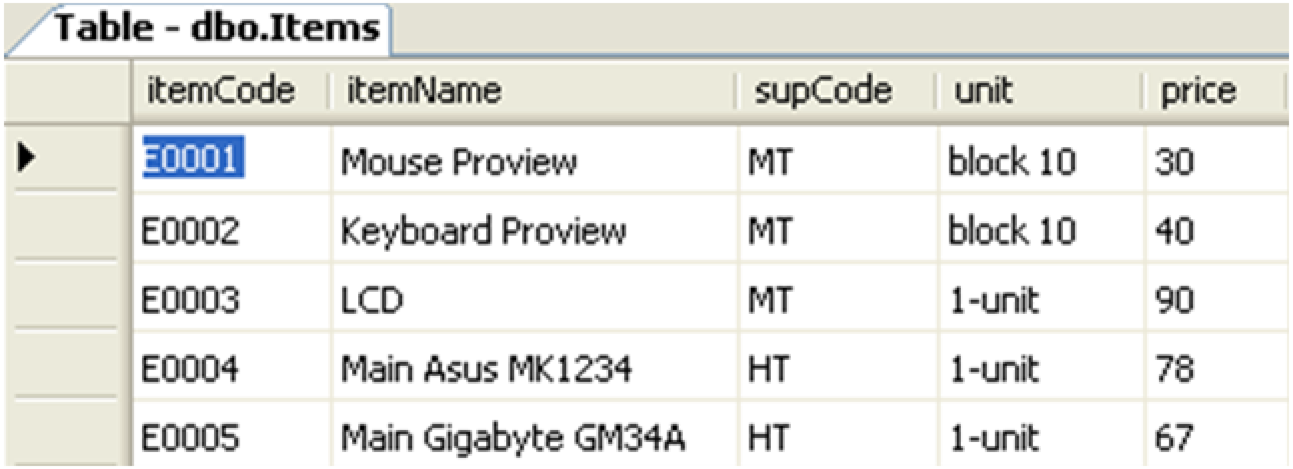


Table - dbo.Items

itemCode	itemName	supCode	unit	price
E0001	Mouse Proview	MT	block 10	30
E0002	Keyboard Proview	MT	block 10	40
E0003	LCD	MT	1-unit	90
E0004	Main Asus MK1234	HT	1-unit	78
E0005	Main Gigabyte GM34A	HT	1-unit	67

RDBMS: Structure Query Language (SQL)

Data Definition Language (DDL):
CREATE…/ ALTER…/ DROP…

3 languages:

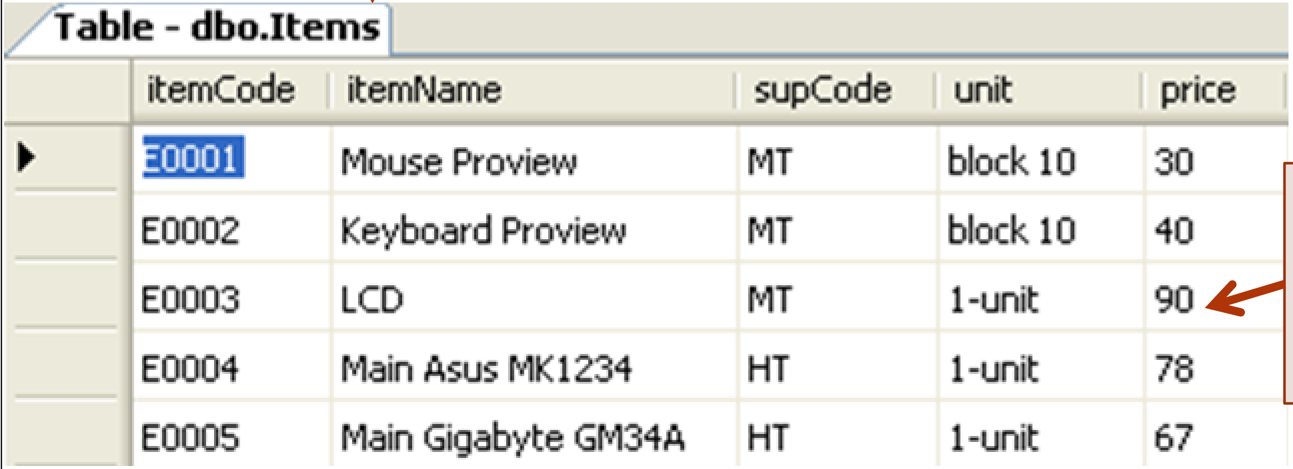
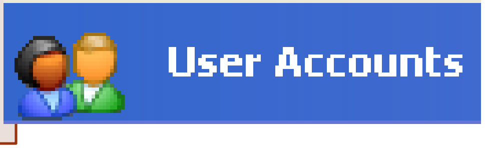


Table - dbo.Items

	itemCode	itemName	supCode	unit	price
▶	E0001	Mouse Proview	MT	block 10	30
	E0002	Keyboard Proview	MT	block 10	40
	E0003	LCD	MT	1-unit	90
	E0004	Main Asus MK1234	HT	1-unit	78
	E0005	Main Gigabyte GM34A	HT	1-unit	67

Data Manipulating Language (DML):
SELECT…/ INSERT INTO … / UPDATE … / DELETE

Data Control Language (DCL):
GRANT…/ REVOKE … / DENY…



RDBMS: SQL…

- ***Common DML queries***:
  - SELECT columns FROM tables WHERE condition
  - UPDATE table SET column=value,… Where condition
  - DELETE FROM table WHERE condition
  - INSERT INTO table Values (val1, val2,…)
  - INSERT INTO table (col1, col2,…) Values (val1, val2,…)

3-JDBC and JDBC Driver

- The JDBC™ API was designed to keep simple things simple. This means that the JDBC makes everyday database tasks easy. This trail walks you through examples of using JDBC to execute common SQL statements, and perform other objectives common to database applications.
- The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.

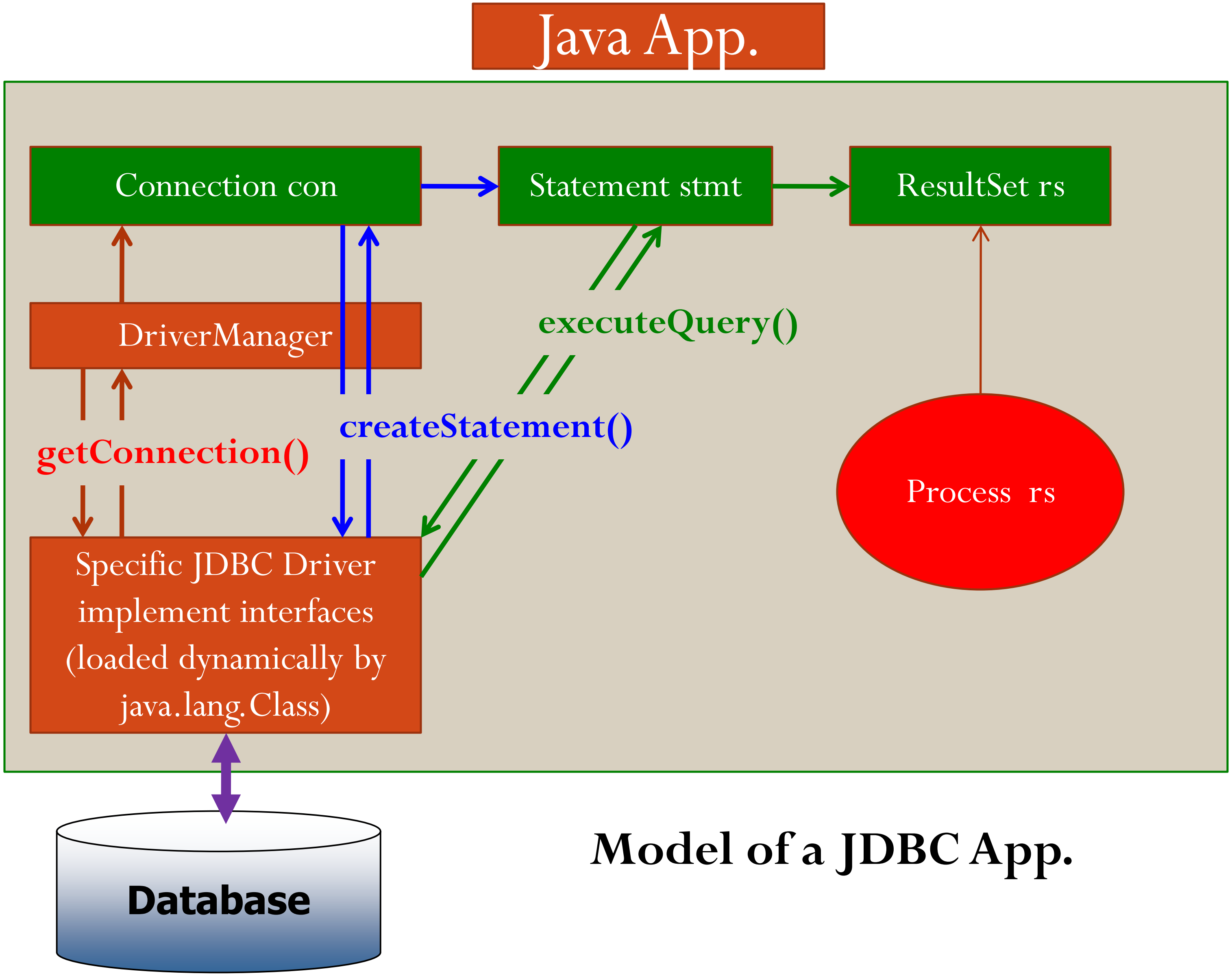
JDBC and JDBC Driver…

- JDBC APIs has 02 parts in the **java.sql** package.

Part	Details	Purposes
JDBC Driver	**DriverManager** class	Java.lang.Class.forName(DriverClass) will dynamically load the concrete driver class, provided by a **specific provider for a specific database**. This class implemented methods declared in JDBC interfaces. The class DriverManager will get a connection to database based on the specific driver class loaded.
JDBC API	Interfaces: **Connection,** **Statement** **ResultSet** **DatabaseMetadata** **ResultSetMetadata** Classes **SQLException**	 For creating a connection to a DBMS For executing SQL statements For storing result data set and achieving columns For getting database metadata For getting resultset metadata

Refer to the java.sql package for more details in Java documentation

JDBC and JDBC Driver…

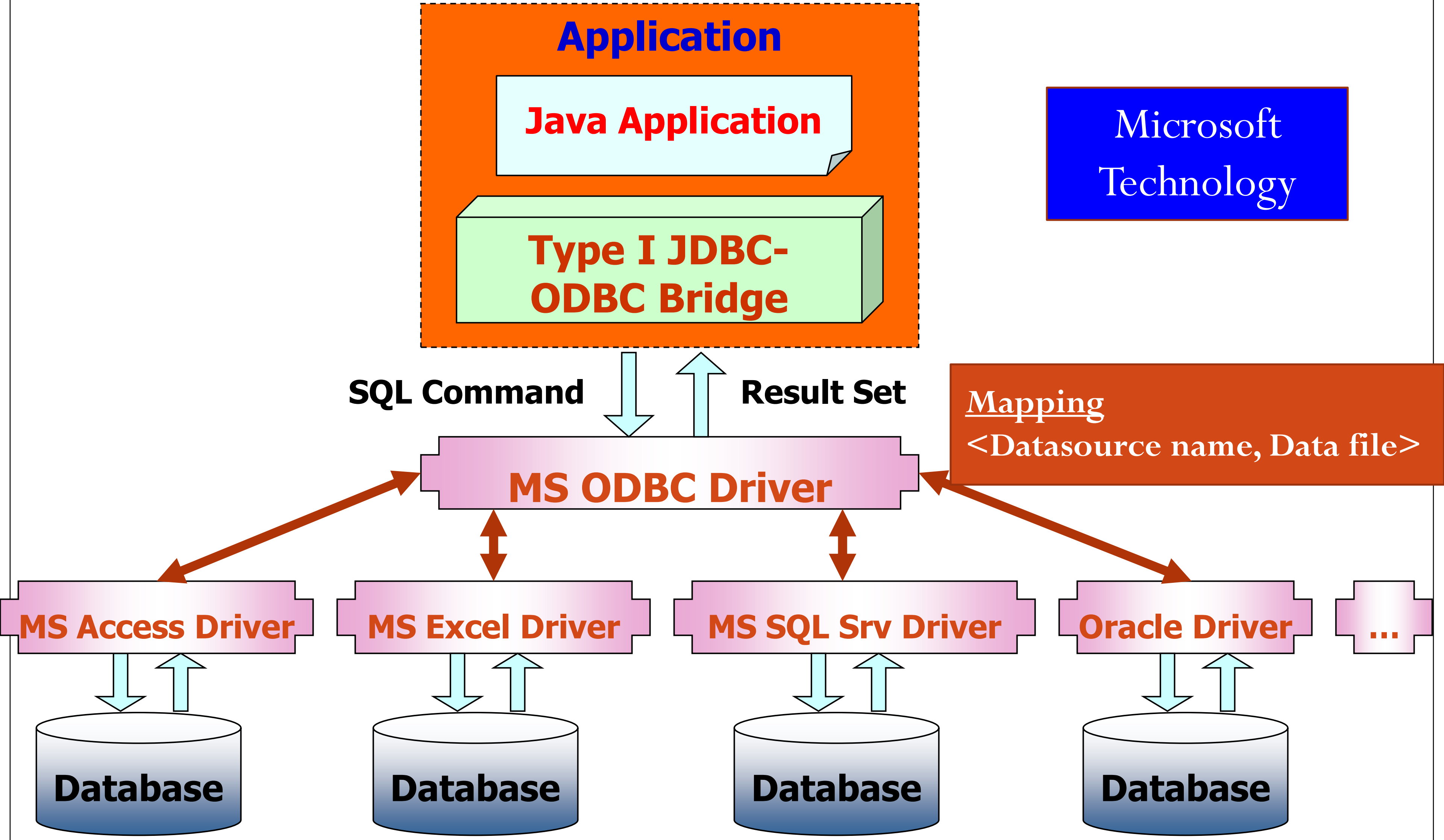


Model of a JDBC App.

JDBC and JDBC Driver…

- DBMS provider/developer will supply a package in which specific classes implementing standard JDBC driver (free).
- Based on characteristics of DBMSs, four types of JDBC drivers are:
  - Type 1: JDBC ODBC
  - Type 2: Native API
  - Type 3: Network Protocol
  - Type 4: Native Protocol
- Type 1 and Type 4 are populated.

Type 1-Driver : JDBC-ODBC Bridge



Microsoft Technology

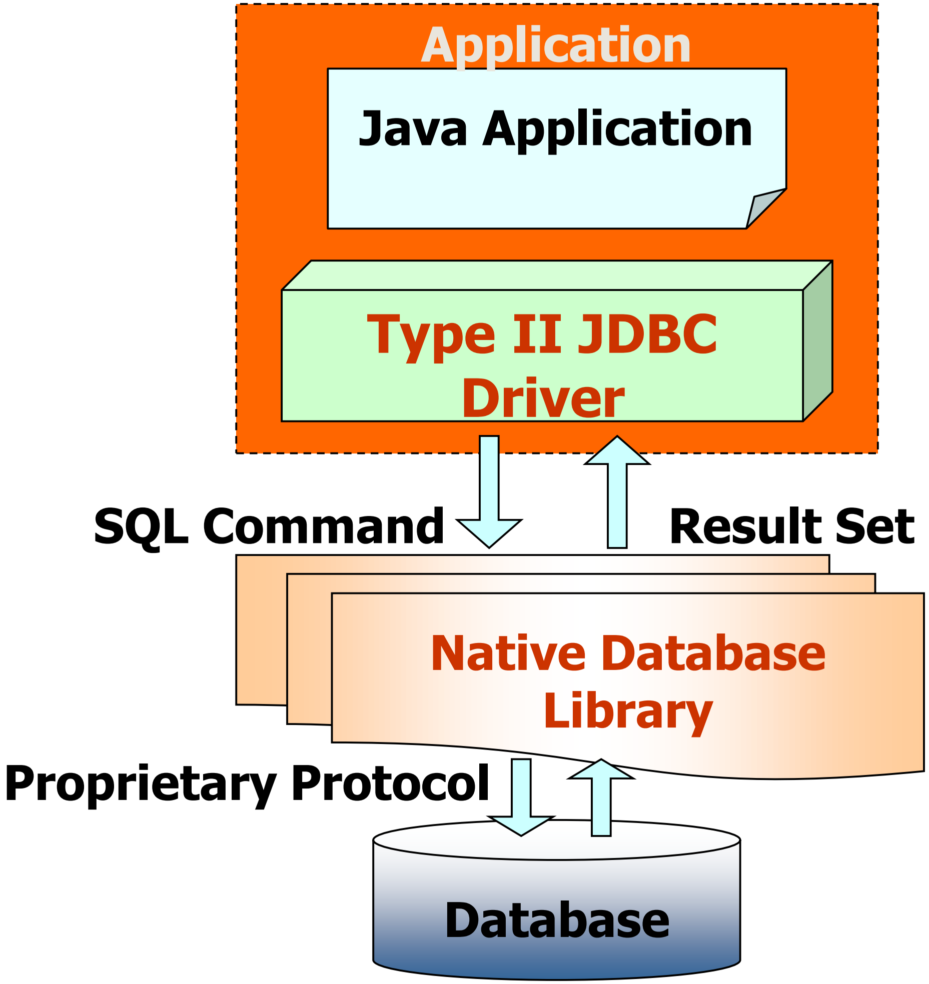
Mapping
<Datasource name, Data file>

Type 1-Driver : JDBC-ODBC…

- This package is in the JDK as default.
- Translates JDBC APIs to ODBC APIs
- Enables the Java applications to interact with any database supported by Microsoft.
- Provides platform dependence, as JDBC ODBC bridge driver uses ODBC
- **JDBC-ODBC bridge is useful when Java driver is not available for a database but it is supported by Microsoft.**
- Disadvantages
  - Platform dependence (Microsoft)
  - The performance is comparatively slower than other drivers
  - Require the ODBC driver and the client DB to be on the server.
- Usage: DSN is registered to use connecting DB (a data source is declared in Control Panel/ODBC Data sources)

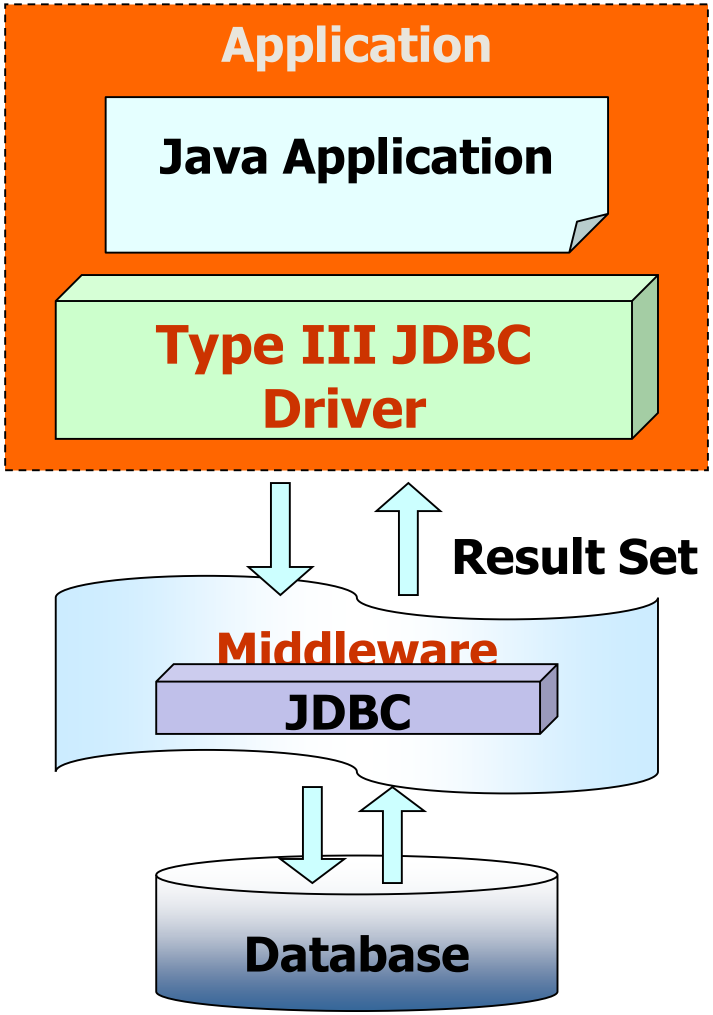
Type 2-Driver: Native API

- Provides access to the database through C/C++ codes.
- Developed using native code libraries
- Native code libraries provide access to the database, and improve the performance
- Java application sends a request for database connectivity as a normal JDBC call to the Native API driver
- Establishes the call, and translates the call to the particular database protocol that is forwarded to the database



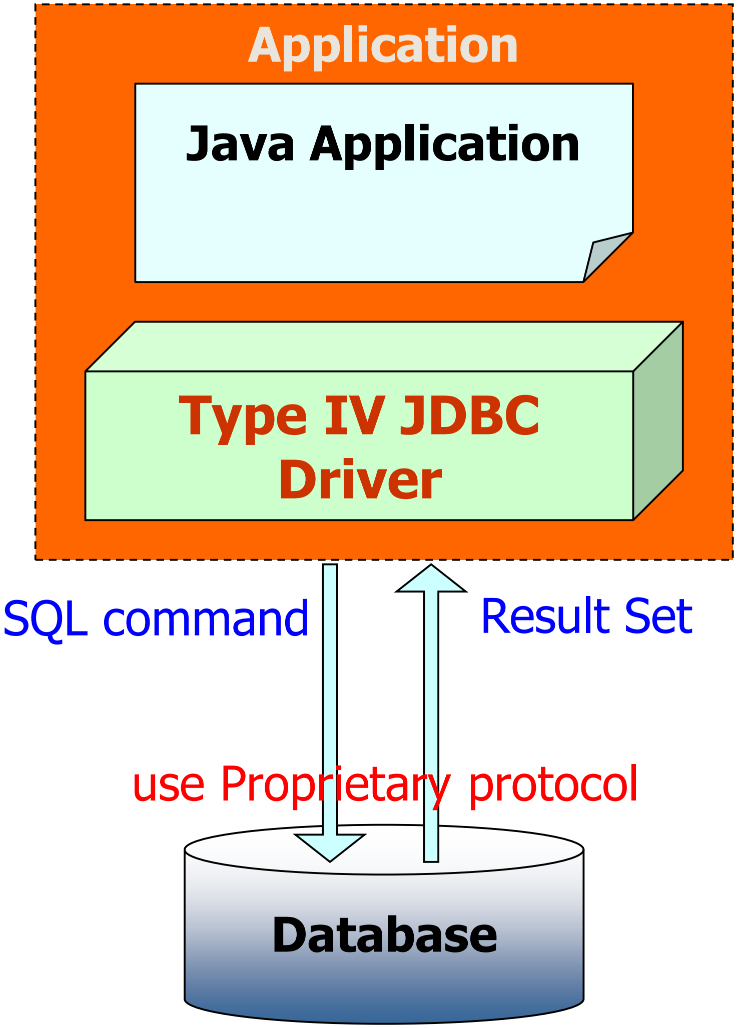
Type 3-Driver: Network Protocol

- Use a pure Java client and communicate with a middleware server using a database-independent protocol.
- The middleware server then communicates the client's requests to the data source
- Manages multiple Java applications connecting to different databases



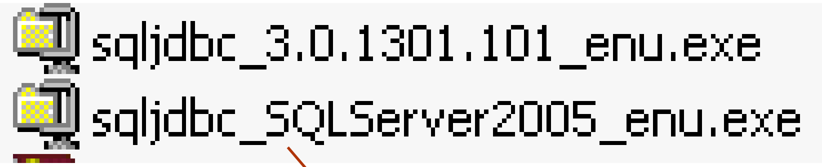
Type 4-Driver: Native Protocol

- Communicates directly with the database using Java sockets
- Improves the performance as translation is not required
- Converts JDBC queries into native calls used by the particular RDBMS
- The driver library is required when it is used and attached with the deployed application (**sqlserver 2000**: mssqlserver.jar, msutil.jar, msbase.jar; **sqlserver 2005**: sqljdbc.jar; **jtds**: jtds.jar …)
- Independent platform



Download Type 4 SQL Server JDBC

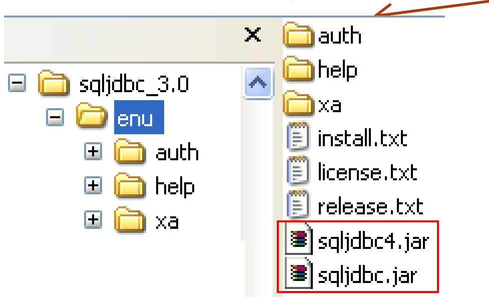
Google : Microsoft SQL Server JDBC Driver



sqljdbc_3.0.1301.101_enu.exe
sqljdbc_SQLServer2005_enu.exe

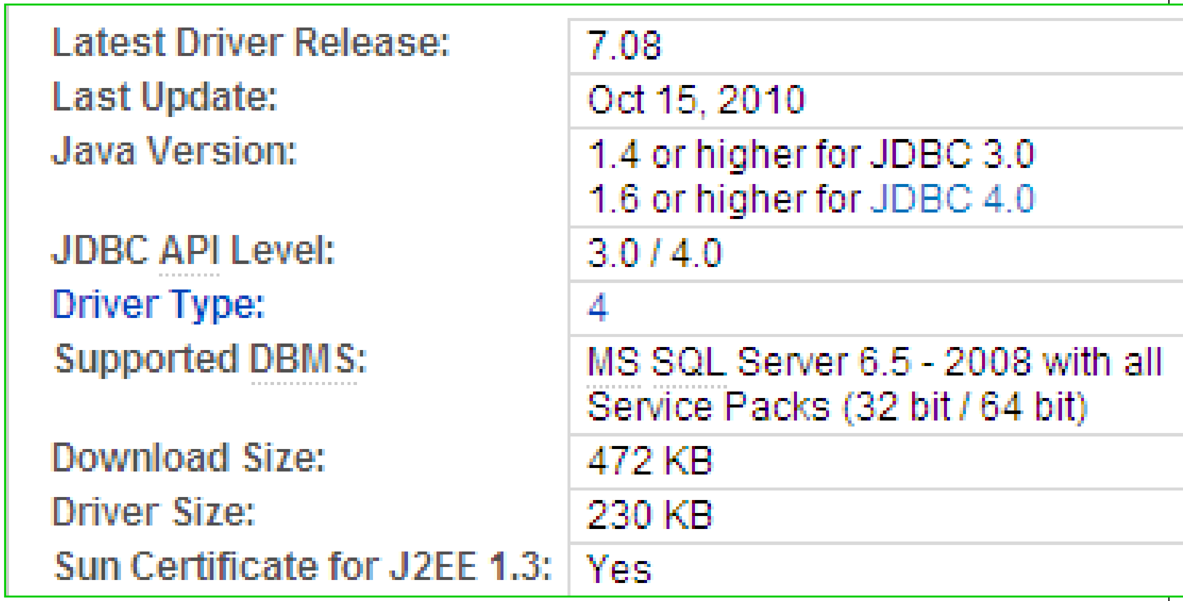
MS SQL Server 2008
MS SQL Server 2005

Setup



```
sqljdbc_3.0
  enu
    auth
    help
    xa
```

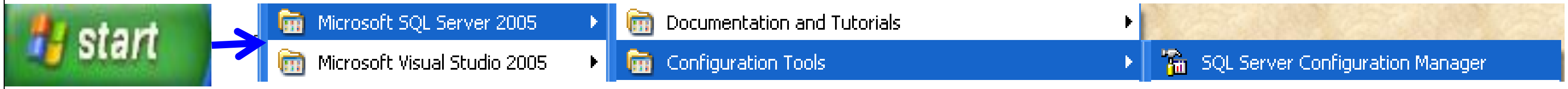
```
auth
help
xa
install.txt
license.txt
release.txt
sqljdbc4.jar
sqljdbc.jar
```



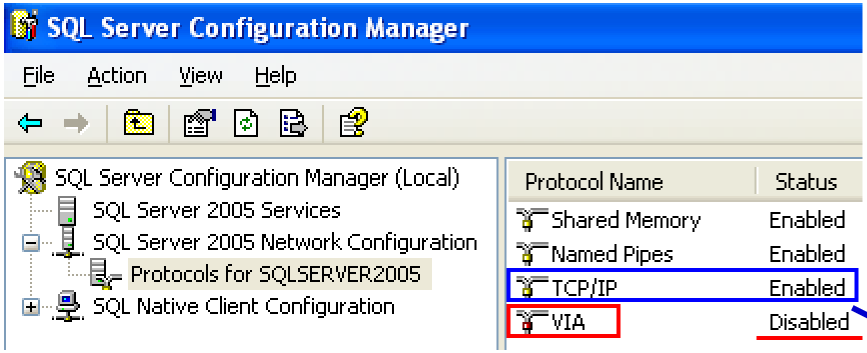
Latest Driver Release:	7.08
Last Update:	Oct 15, 2010
Java Version:	1.4 or higher for JDBC 3.0 1.6 or higher for JDBC 4.0
JDBC API Level:	3.0 / 4.0
Driver Type:	4
Supported DBMS:	MS SQL Server 6.5 - 2008 with all Service Packs (32 bit / 64 bit)
Download Size:	472 KB
Driver Size:	230 KB
Sun Certificate for J2EE 1.3:	Yes

Demonstrations

Configure Ports, Protocols for SQL Server


start
Microsoft SQL Server 2005
Microsoft Visual Studio 2005
Documentation and Tutorials
Configuration Tools
SQL Server Configuration Manager

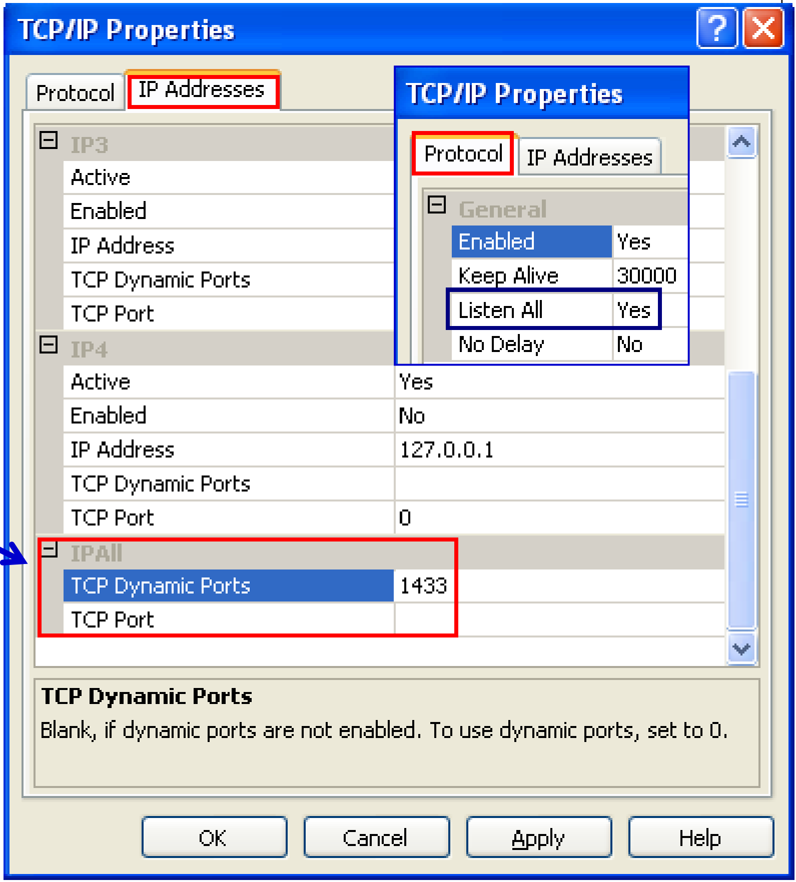
Enable Server protocols and port


SQL Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server 2005 Services
SQL Server 2005 Network Configuration
Protocols for SQLSERVER2005
SQL Native Client Configuration

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Enabled
TCP/IP	Enabled
VIA	Disabled

Attention: Disable VIA

Right click

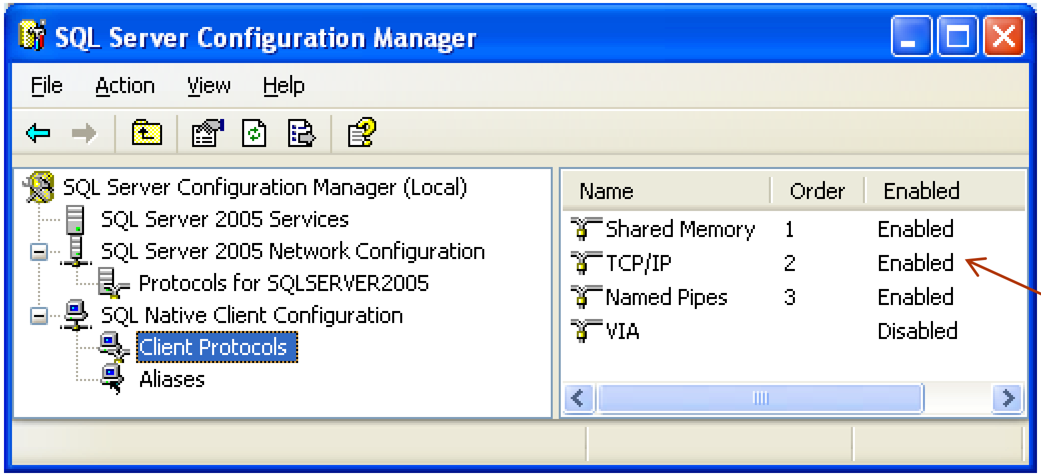


General	
Enabled	Yes
Keep Alive	30000
Listen All	Yes
No Delay	No

IP3	
Active	
Enabled	
IP Address	
TCP Dynamic Ports	
TCP Port	
IP4	
Active	Yes
Enabled	No
IP Address	127.0.0.1
TCP Dynamic Ports	
TCP Port	0
IPAll	
TCP Dynamic Ports	1433
TCP Port	

TCP Dynamic Ports

Blank, if dynamic ports are not enabled. To use dynamic ports, set to 0.

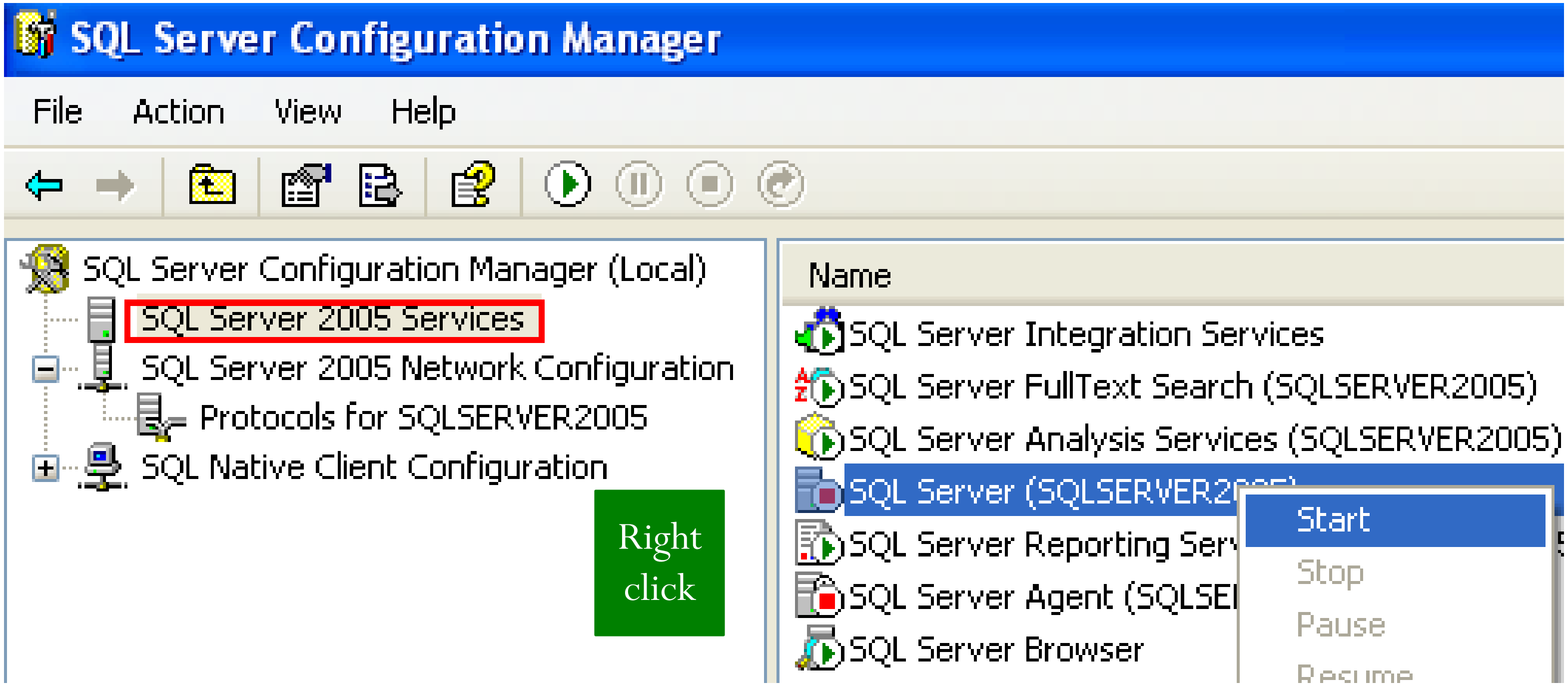

SQL Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server 2005 Services
SQL Server 2005 Network Configuration
Protocols for SQLSERVER2005
SQL Native Client Configuration
Client Protocols
Aliases

Name	Order	Enabled
Shared Memory	1	Enabled
TCP/IP	2	Enabled
Named Pipes	3	Enabled
VIA		Disabled

Enable client protocols and port

Configure Ports, Protocols for SQL Server…

Stop then restart SQL Server and SQL Server Agent for settings are affected.



SQL Server Configuration Manager
File
Action
View
Help
SQL Server Configuration Manager (Local)
SQL Server 2005 Services
SQL Server 2005 Network Configuration
Protocols for SQLSERVER2005
SQL Native Client Configuration
Name
SQL Server Integration Services
SQL Server FullText Search (SQLSERVER2005)
SQL Server Analysis Services (SQLSERVER2005)
SQL Server (SQLSERVER2005)
SQL Server Reporting Serv
SQL Server Agent (SQLSE
SQL Server Browser
Start
Stop
Pause
Resume
Right click

4-Steps to Develop a JDBC Application

Step	Description	Use (java.sql package)	Methods
1	**Load JDBC Driver**	Java.lang.Class	forName(…)
2	**Establish a DB connection**	java.sql.Connection java.sql.DriverManager	DriverManager getConnection(…) → Connection
3	**Create & execute SQL statements**	java.sql.Statement java.sql.PrepareStatement java.sql.CallableStatement	execute(…) executeQuery(…) → SELECT executeUpdate(…) → INSERT/UPDATE/DELETE

4	**Process the results**	java.sql.ResultSet	first(), last(), next(), previous()
			getXXX(..)

5	**Close**	ResultSet, Statement, Connection	close()

(Demo Type 1-Driver) Register a DSN

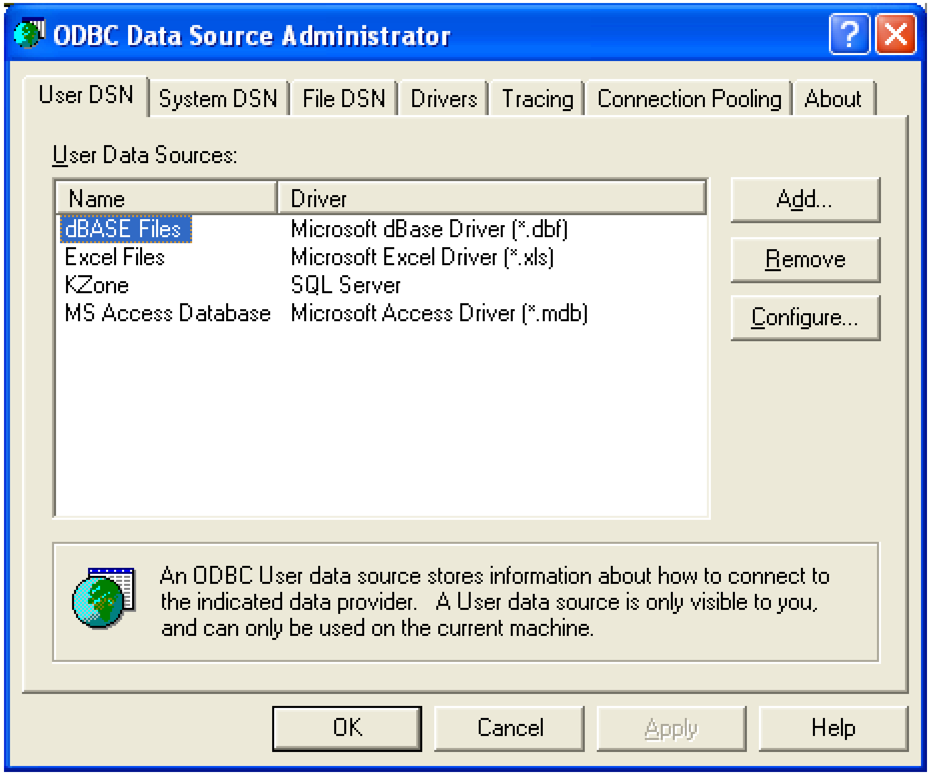
(1) Open the Control Panel

(2) Select Administrative Tools.

(3) Select Data Sources (ODBC)

(4) Select User DSN or System DSN tab.

(5) Click the Add button.



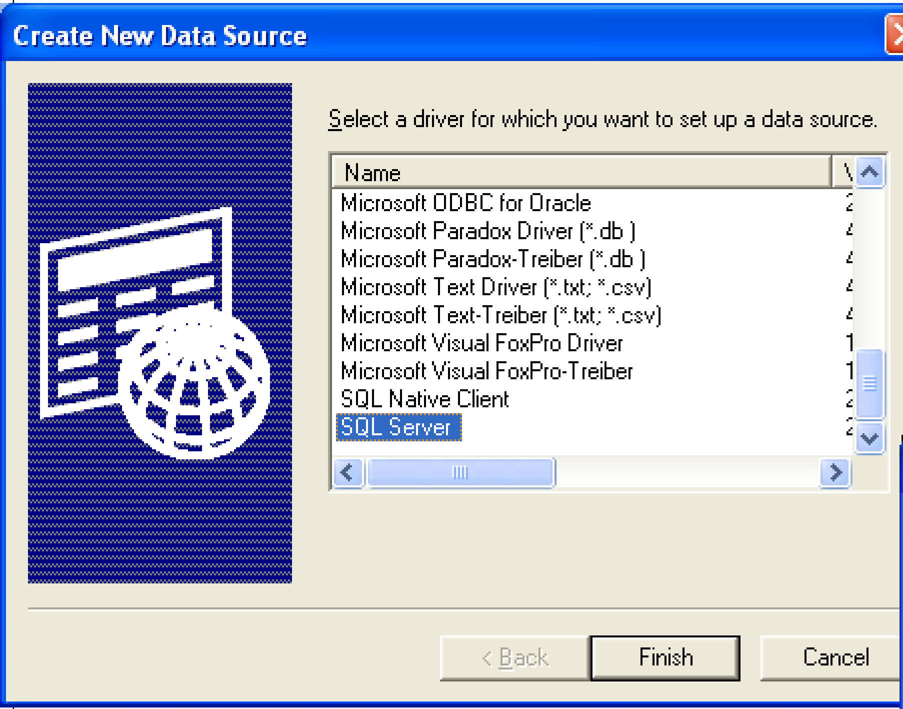
ODBC Data Source Administrator

User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
KZone	SQL Server
MS Access Database	Microsoft Access Driver (*.mdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

(Demo Type 1-Driver) Register a DSN

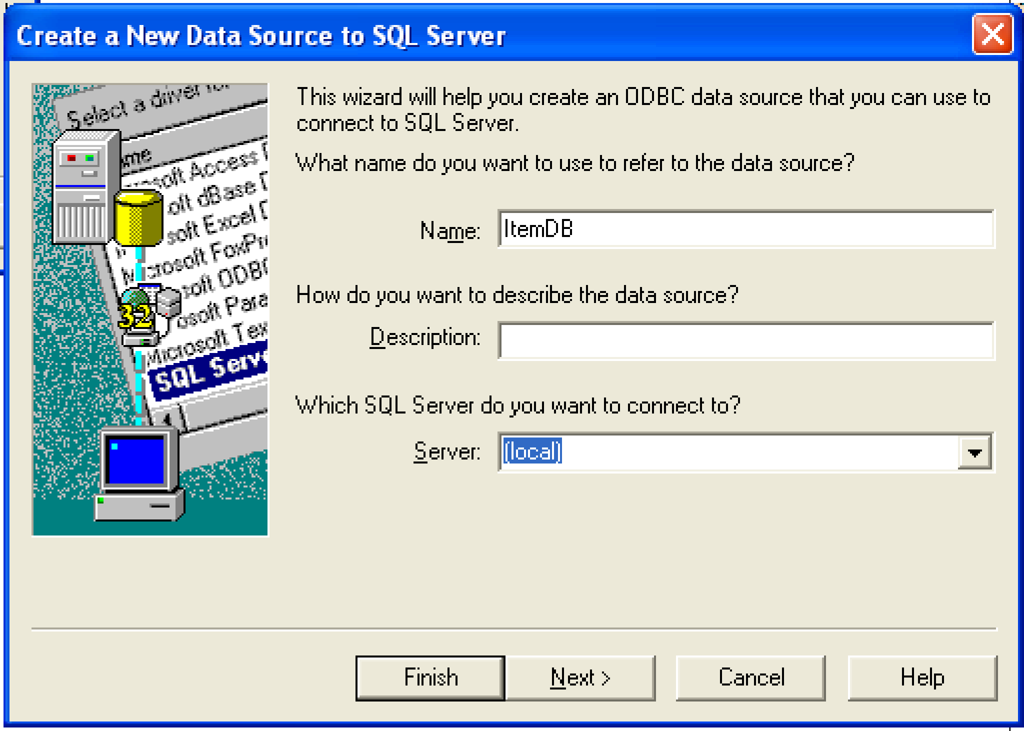


Create New Data Source

Select a driver for which you want to set up a data source.

Name	V
Microsoft ODBC for Oracle	2
Microsoft Paradox Driver (*.db)	4
Microsoft Paradox-Treiber (*.db)	4
Microsoft Text Driver (*.txt; *.csv)	4
Microsoft Text-Treiber (*.txt; *.csv)	4
Microsoft Visual FoxPro Driver	1
Microsoft Visual FoxPro-Treiber	1
SQL Native Client	2
SQL Server	2

- Select file type
- Click the Finish button.



Create a New Data Source to SQL Server

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: ItemDB

How do you want to describe the data source?

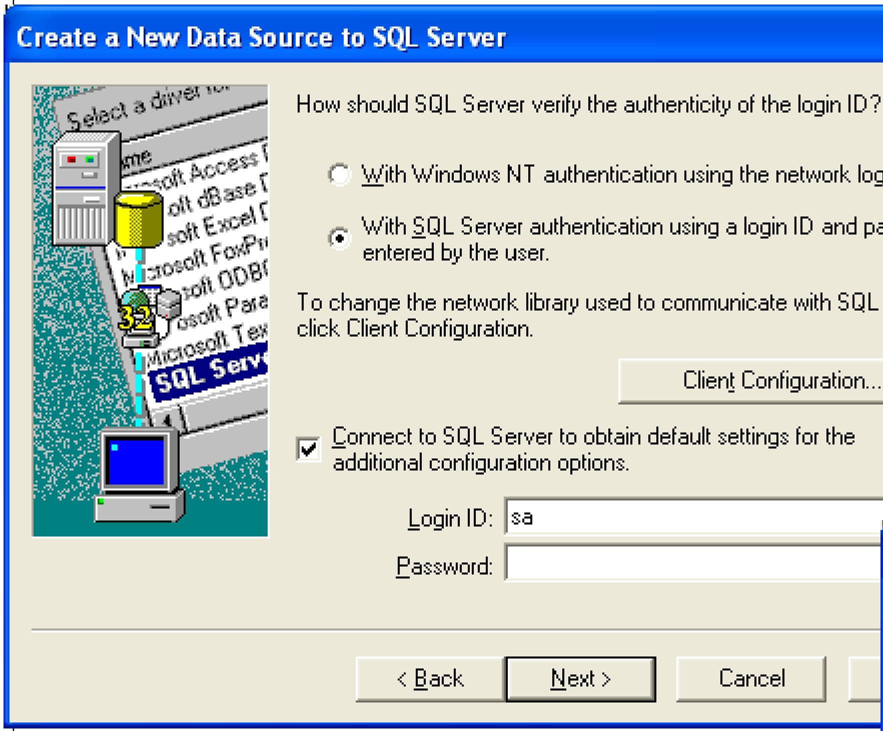
Description:

Which SQL Server do you want to connect to?

Server: (local)

- Give a data source name, such as ItemDB
- Select server (SQL Server)
- Click the Finish button

(Demo Type 1-Driver) Register a DSN



Create a New Data Source to SQL Server

How should SQL Server verify the authenticity of the login ID?

○ With Windows NT authentication using the network login ID.

● With SQL Server authentication using a login ID and password entered by the user.

To change the network library used to communicate with SQL Server, click Client Configuration.

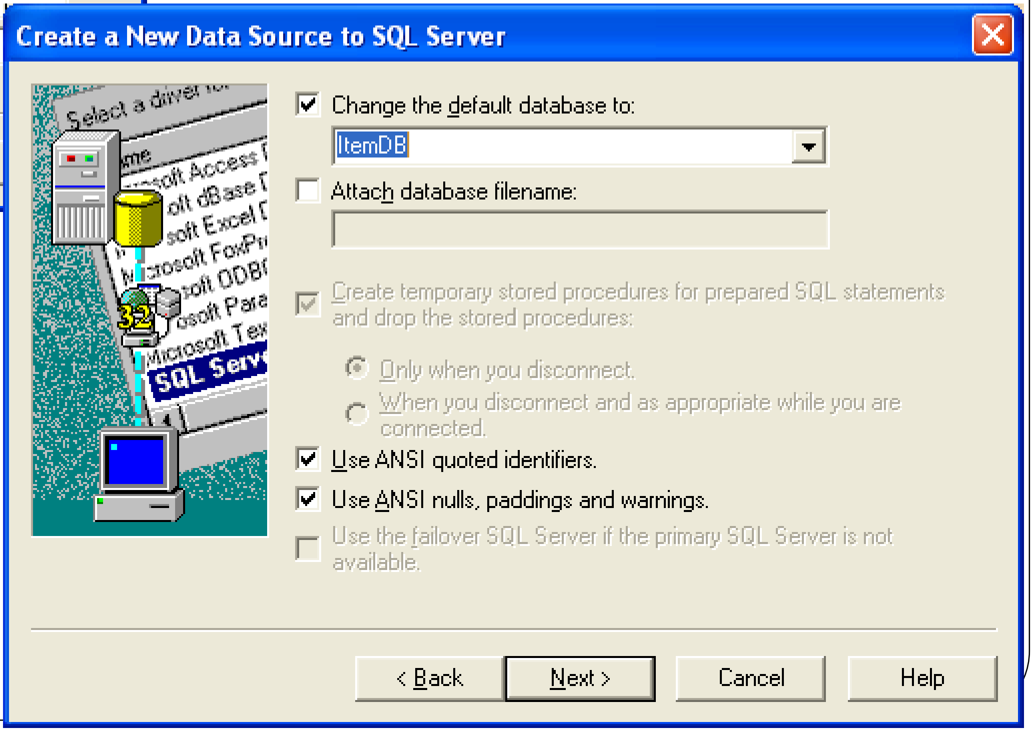
☑ Connect to SQL Server to obtain default settings for the additional configuration options.

Login ID: sa

Password:

- Select security mode.
- Supply Login ID and Password
- Click the next button

- Select database.
- Click the next button



Create a New Data Source to SQL Server

☑ Change the default database to:

ItemDB

☐ Attach database filename:

☑ Create temporary stored procedures for prepared SQL statements and drop the stored procedures:

● Only when you disconnect.

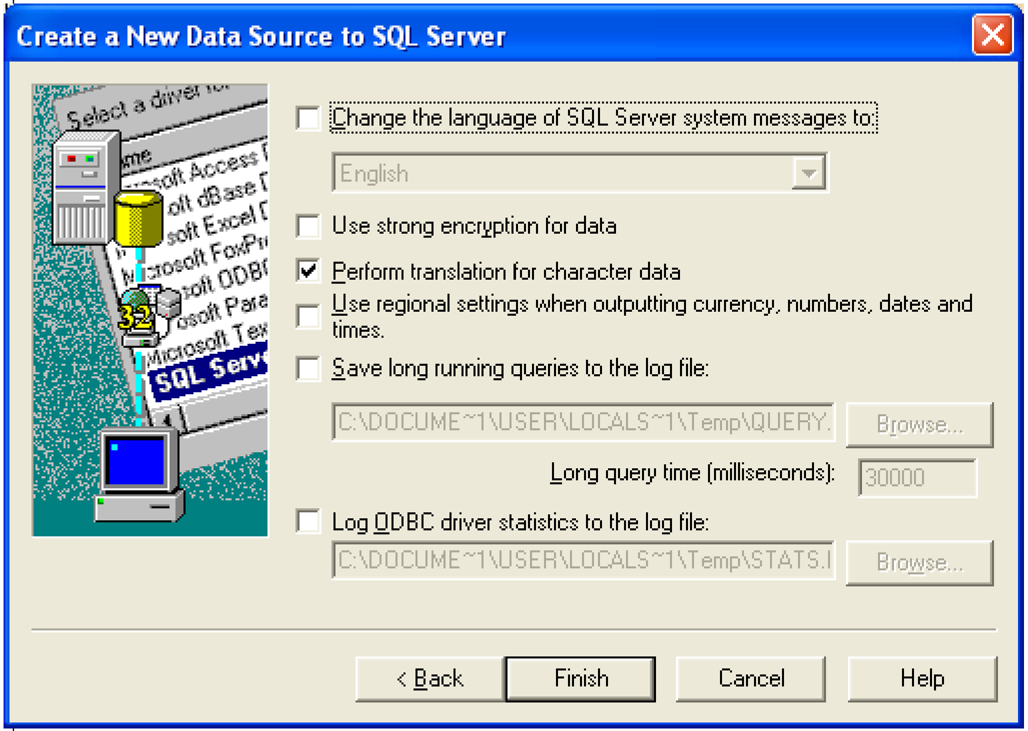
○ When you disconnect and as appropriate while you are connected.

☑ Use ANSI quoted identifiers.

☑ Use ANSI nulls, paddings and warnings.

☐ Use the failover SQL Server if the primary SQL Server is not available.

(Demo Type 1-Driver) Register a DSN



Create a New Data Source to SQL Server

□ Change the language of SQL Server system messages to:

English

□ Use strong encryption for data

☑ Perform translation for character data

□ Use regional settings when outputting currency, numbers, dates and times.

□ Save long running queries to the log file:

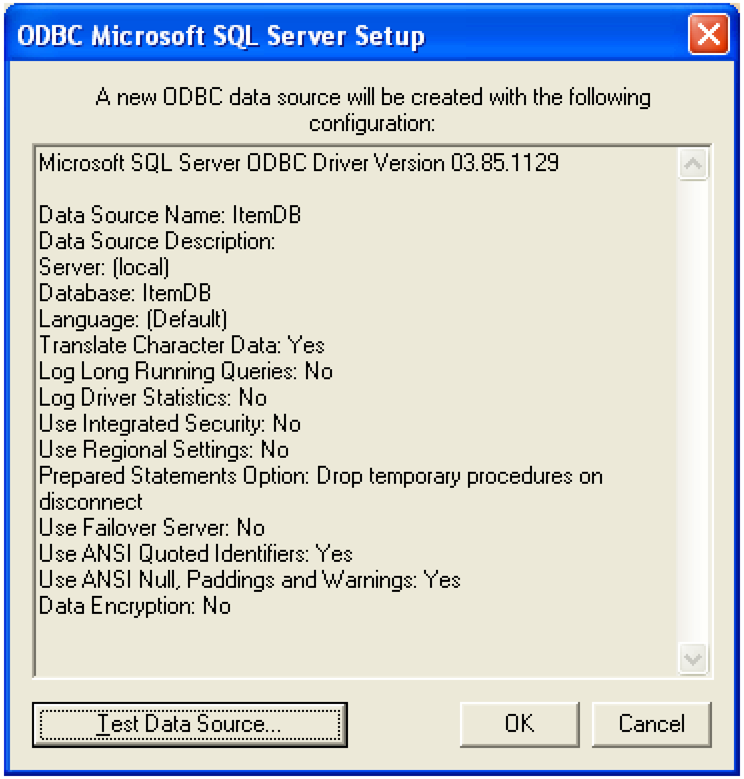
C:\DOCUME~1\USER\LOCALS~1\Temp\QUERY.

Long query time (milliseconds): 30000

□ Log ODBC driver statistics to the log file:

C:\DOCUME~1\USER\LOCALS~1\Temp\STATS.I

- Click the Finish button

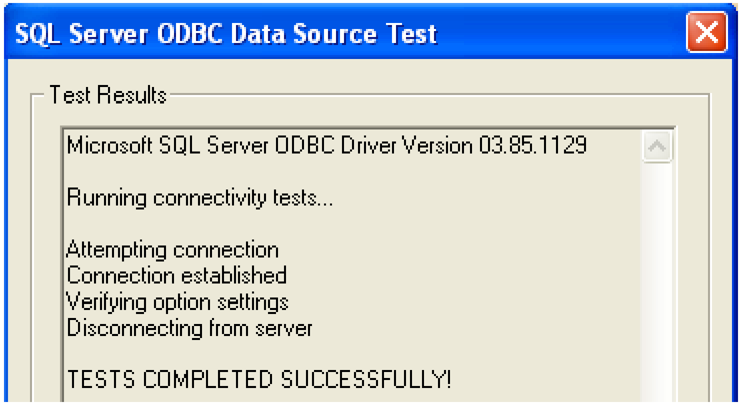


ODBC Microsoft SQL Server Setup

A new ODBC data source will be created with the following configuration:

```
Microsoft SQL Server ODBC Driver Version 03.85.1129

Data Source Name: ItemDB
Data Source Description:
Server: (local)
Database: ItemDB
Language: (Default)
Translate Character Data: Yes
Log Long Running Queries: No
Log Driver Statistics: No
Use Integrated Security: No
Use Regional Settings: No
Prepared Statements Option: Drop temporary procedures on
disconnect
Use Failover Server: No
Use ANSI Quoted Identifiers: Yes
Use ANSI Null, Paddings and Warnings: Yes
Data Encryption: No
```

- **Click the Test…button**



SQL Server ODBC Data Source Test

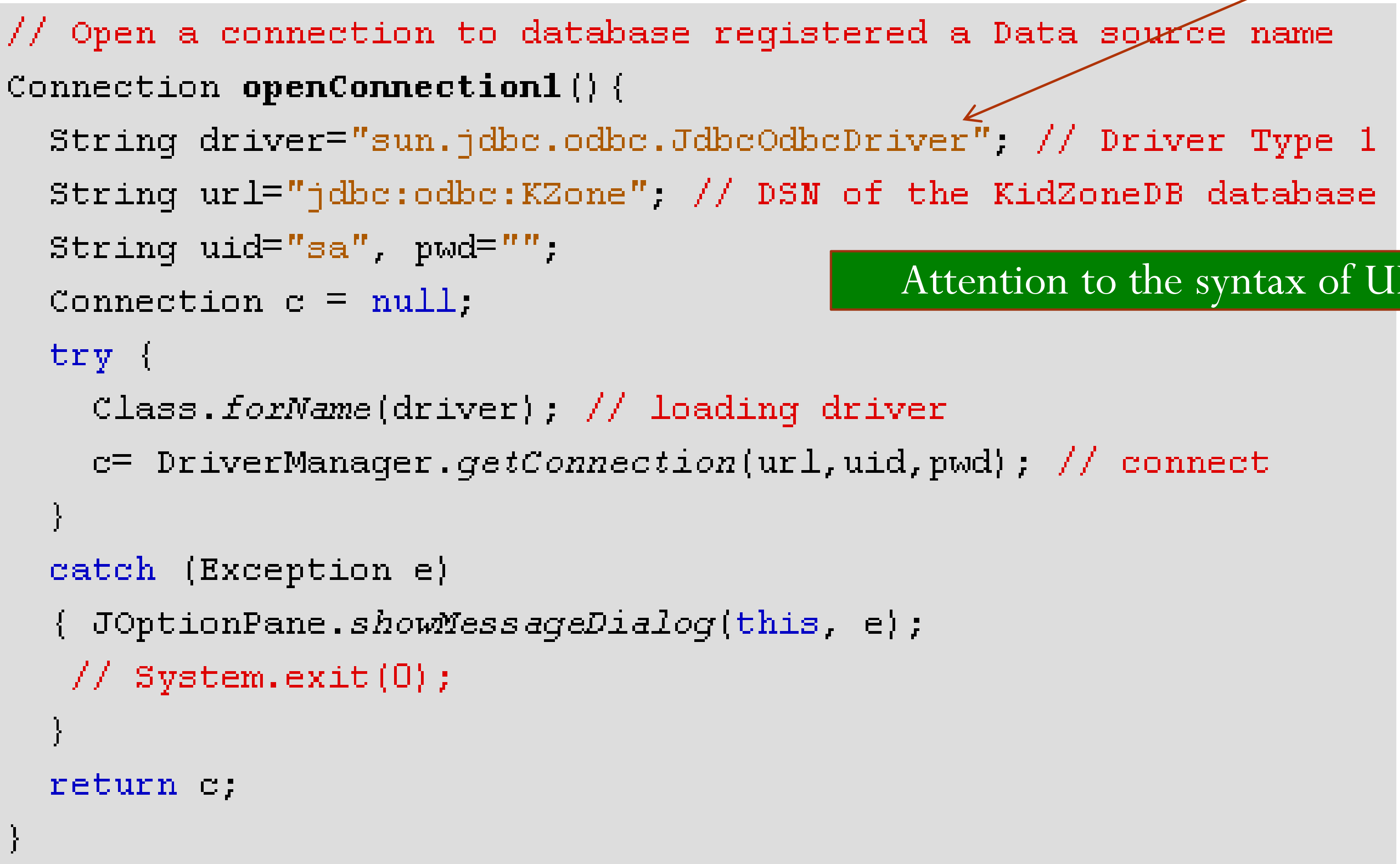
Test Results

```
Microsoft SQL Server ODBC Driver Version 03.85.1129

Running connectivity tests...

Attempting connection
Connection established
Verifying option settings
Disconnecting from server

TESTS COMPLETED SUCCESSFULLY!
```

Step 1: Register JDBC Driver
Step 2: Establish a connection to DB

Driver Class

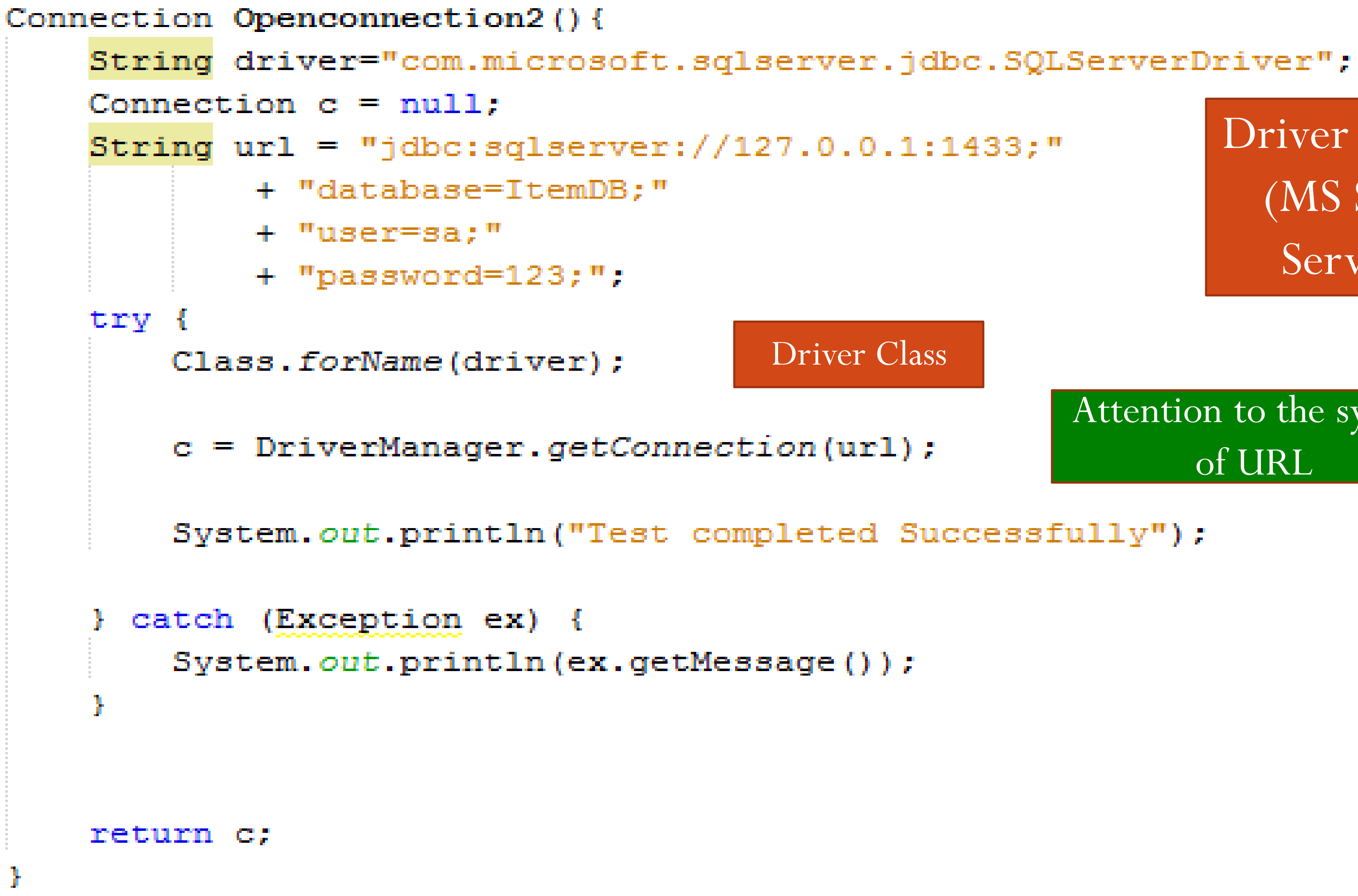
Driver Type 1 with Data Source Name registered in ODBC



```java
// Open a connection to database registered a Data source name
Connection openConnection1(){
  String driver="sun.jdbc.odbc.JdbcOdbcDriver"; // Driver Type 1
  String url="jdbc:odbc:KZone"; // DSN of the KidZoneDB database
  String uid="sa", pwd="";
  Connection c = null;
  try {
    Class.forName(driver); // loading driver
    c= DriverManager.getConnection(url,uid,pwd); // connect
  }
  catch (Exception e)
  { JOptionPane.showMessageDialog(this, e);
   // System.exit(0);
  }
  return c;
}
```

Attention to the syntax of URL

Step 1: Register JDBC Driver
Step 2: Establish a connection to DB



```
Connection Openconnection2(){
        String driver="com.microsoft.sqlserver.jdbc.SQLServerDriver";
        Connection c = null;
        String url = "jdbc:sqlserver://127.0.0.1:1433;"
                + "database=ItemDB;"
                + "user=sa;"
                + "password=123;";
        try {
            Class.forName(driver);

            c = DriverManager.getConnection(url);

            System.out.println("Test completed Successfully");

        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }


        return c;
}
```

Driver type 4 (MS SQL Server)

Driver Class

Attention to the syntax of URL

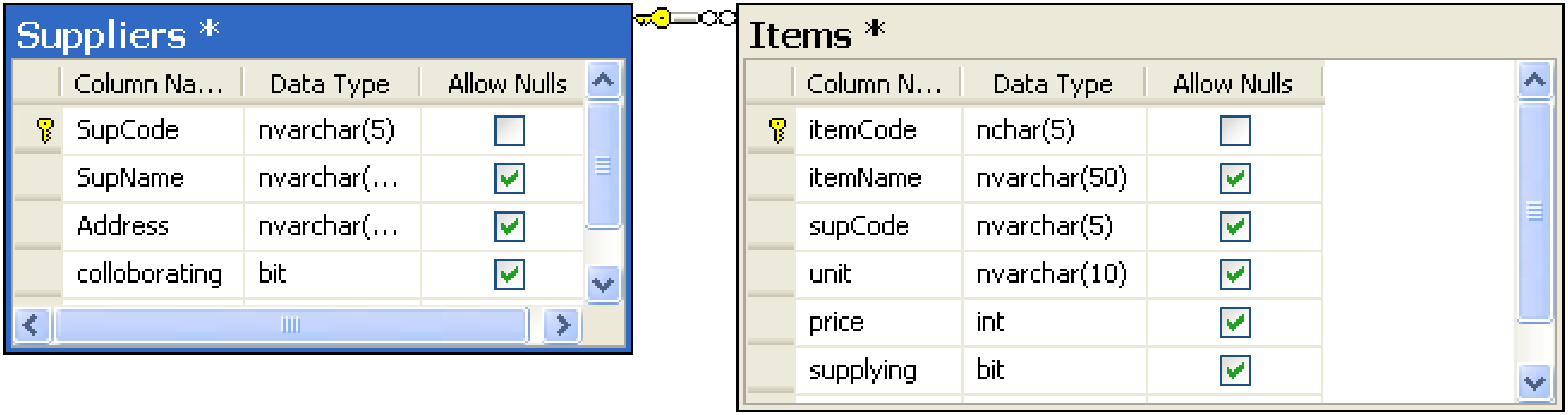
Step 3: Create &Execute a SQL statement

```
String sql1 = “SELECT  columns FROM table1, table2, …  WHERE condition”;
String sql2 = “UPDATE table SET column = value, … WHERE  condition”;
String sql3 = “INSERT INTO table VALUES ( val1, val2, … )” ;
String sql4 = “INSERT INTO table  (col1, col2, col3) VALUES ( val1, val2, val3)” ;
String sql5 =  “UPDATE table SET col1 = ?, col2=?  WHERE  condition”;
```

```
// Connection con was created
Statement stmt= con.createStatement();
ResultSet rs= stmt.executeQuery(sql1);
int numOfInfectedRows = stmt.executeUpdate(sql2);
int numOfInfectedRows = stmt.executeUpdate(sql3);
int numOfInfectedRows = stmt.executeUpdate(sql4);

PreparedStatement pStmt = con.preparedStatement(sql5);
pStmt.setXXX (index, val);  // from 1
int numOfInfectedRows = pStmt.executeUpdate(); // no argument
```

(Demo 1) Create database

- Use MS Access or MS SQL Server 2005
- Database name: ItemDB
- Tables and Relationship:



Suppliers *

	Column Na...	Data Type	Allow Nulls
🔑	SupCode	nvarchar(5)	□
	SupName	nvarchar(...	☑
	Address	nvarchar(...	☑
	colloborating	bit	☑

Items *

	Column N...	Data Type	Allow Nulls
🔑	itemCode	nchar(5)	□
	itemName	nvarchar(50)	☑
	supCode	nvarchar(5)	☑
	unit	nvarchar(10)	☑
	price	int	☑
	supplying	bit	☑

You can download this database file from CMS.

(Demo 1) Create database…

Initial data:

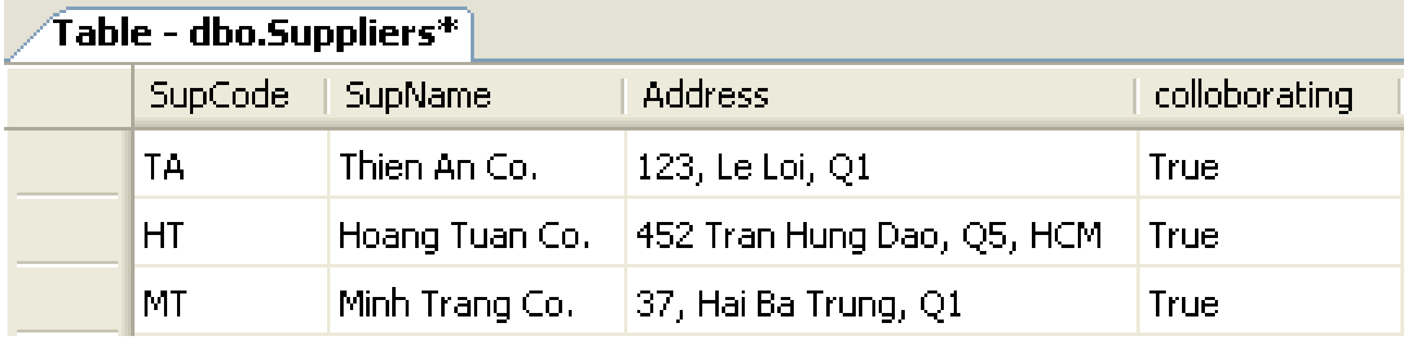


Table - dbo.Suppliers*

SupCode	SupName	Address	colloborating
TA	Thien An Co.	123, Le Loi, Q1	True
HT	Hoang Tuan Co.	452 Tran Hung Dao, Q5, HCM	True
MT	Minh Trang Co.	37, Hai Ba Trung, Q1	True

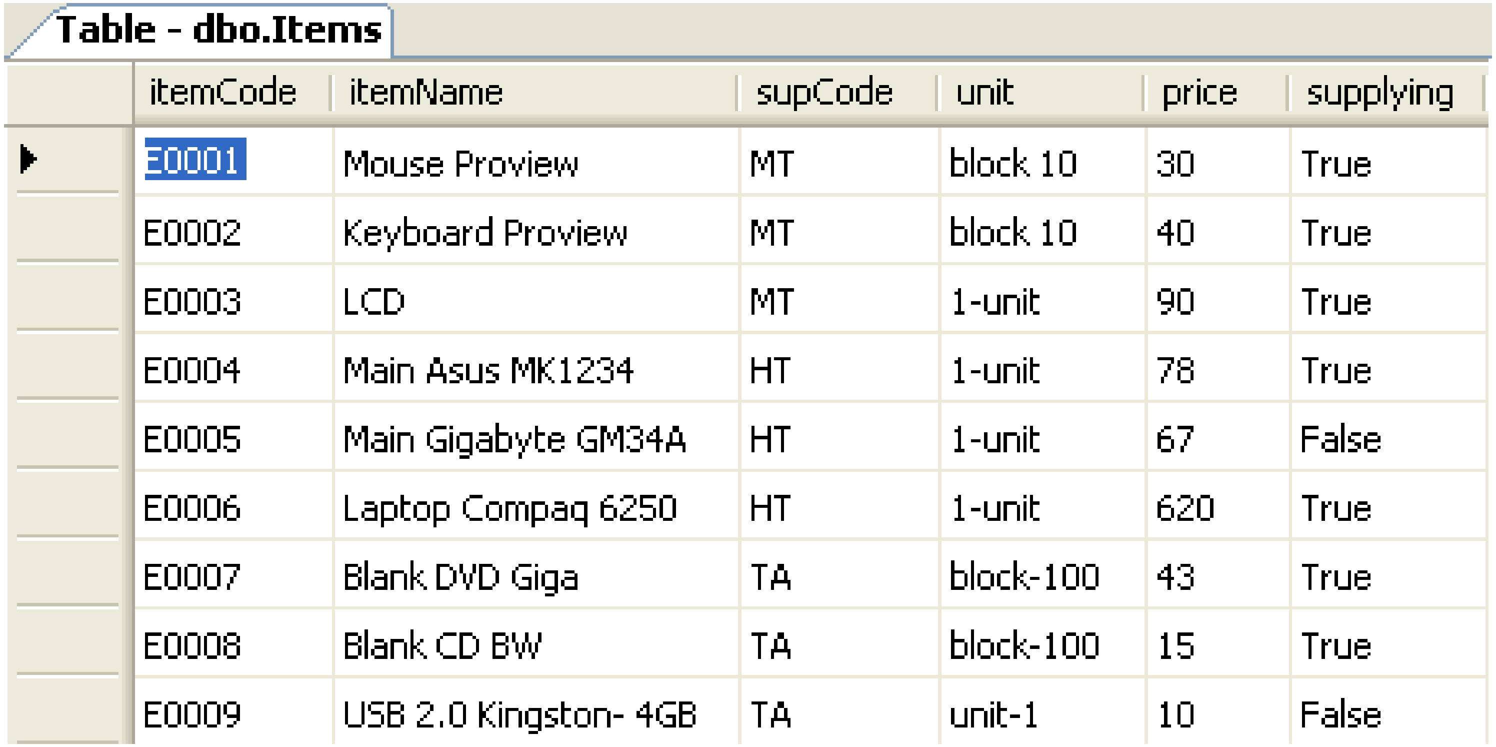
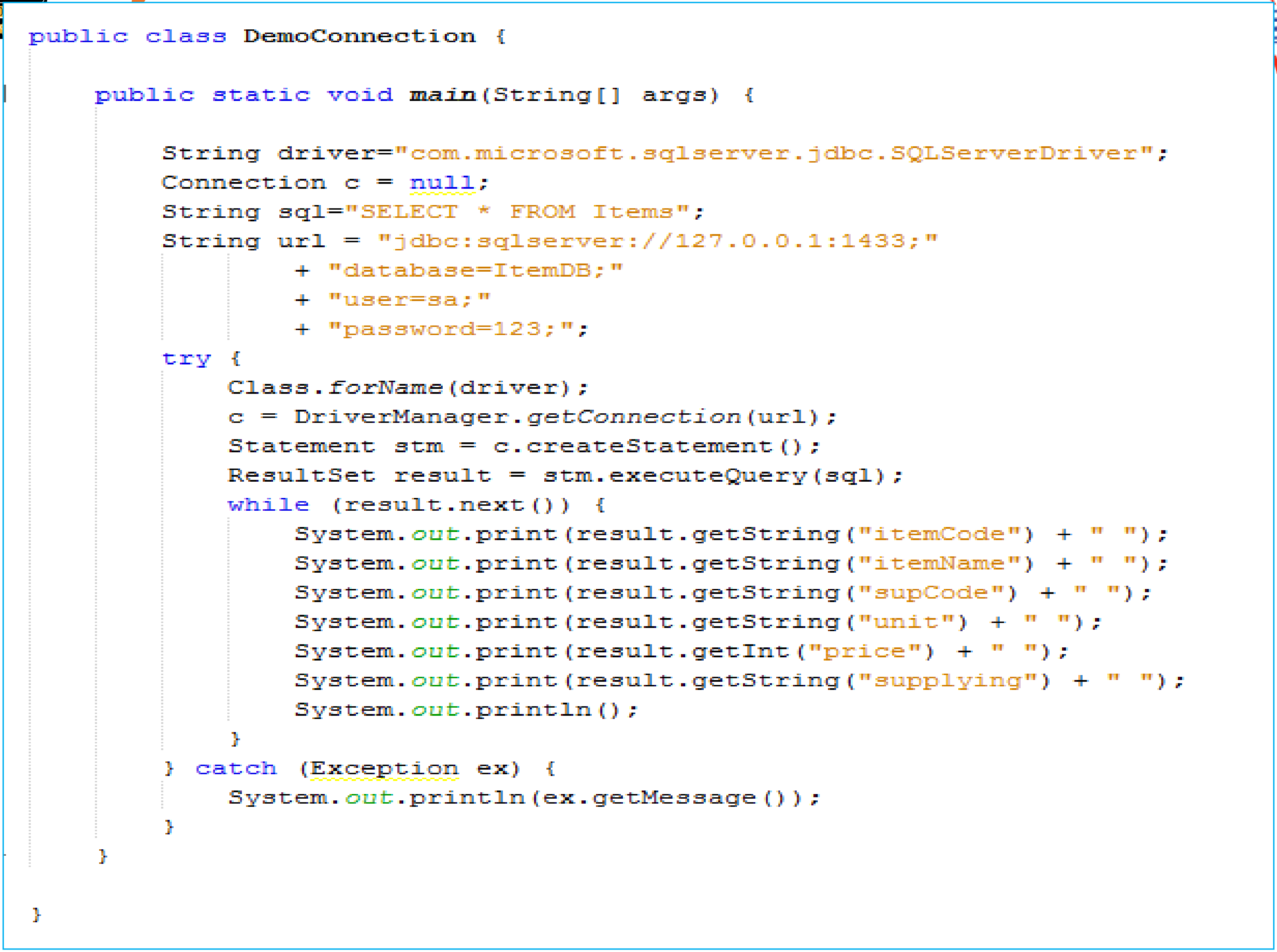
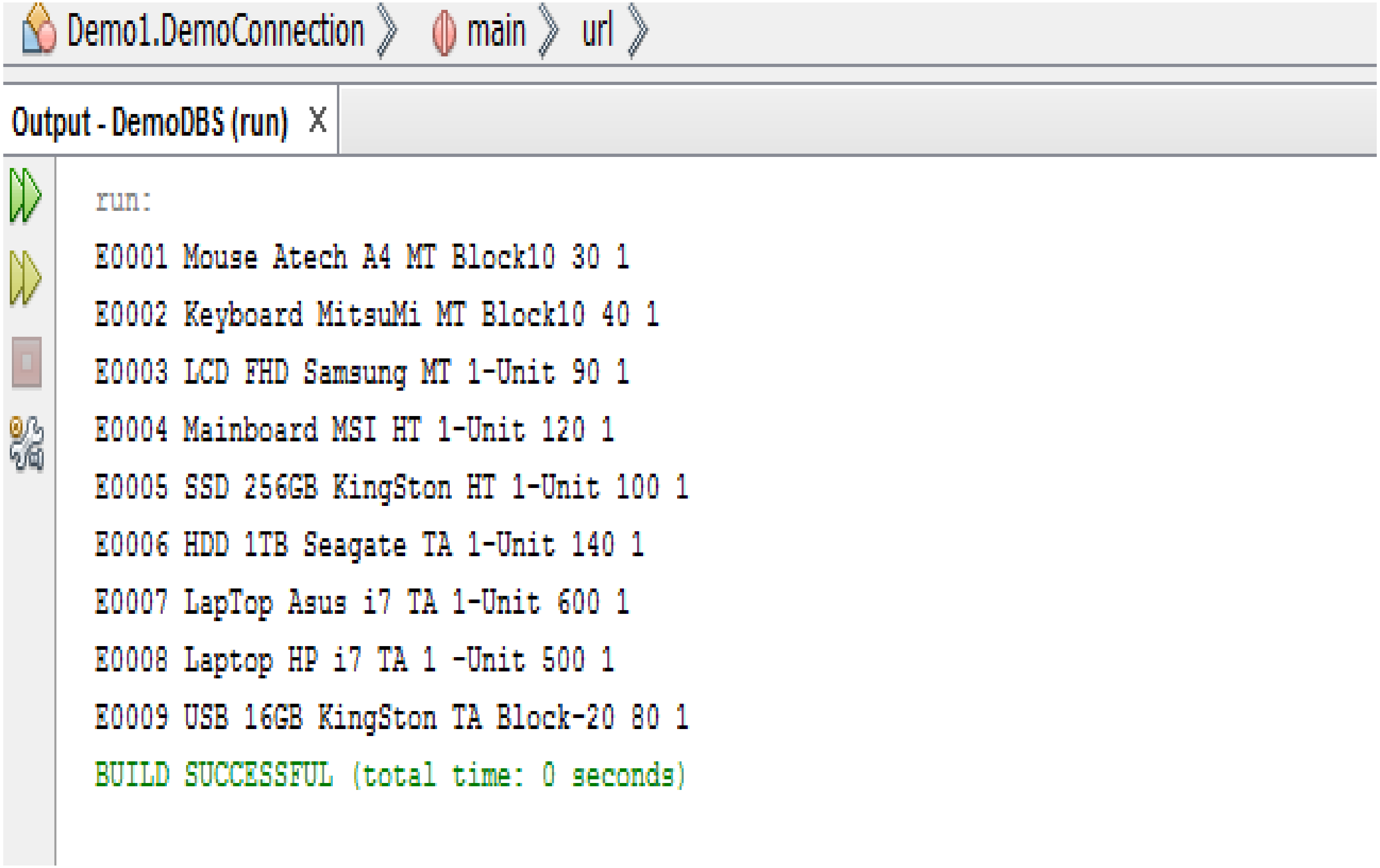


Table - dbo.Items

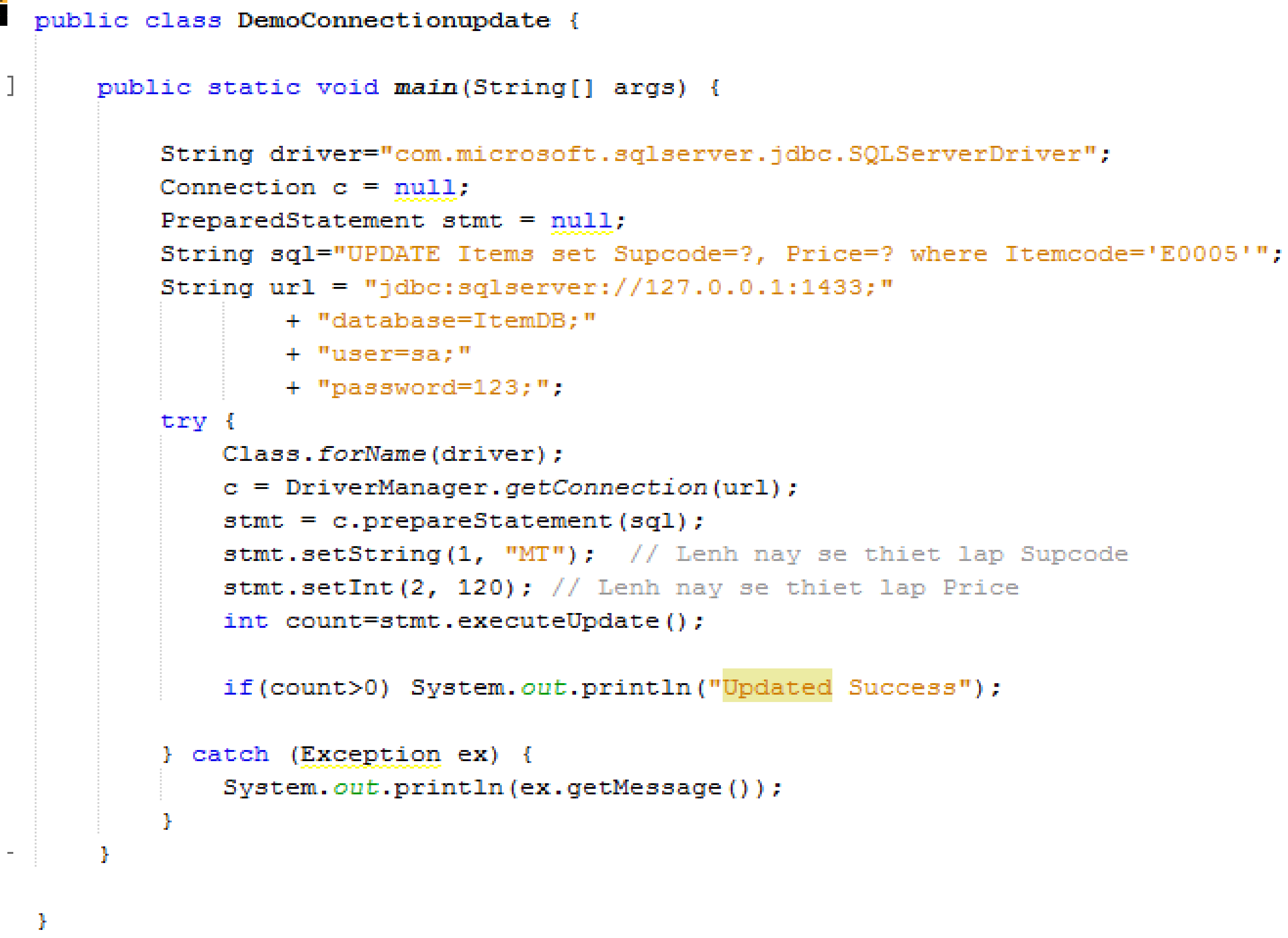
itemCode	itemName	supCode	unit	price	supplying
E0001	Mouse Proview	MT	block 10	30	True
E0002	Keyboard Proview	MT	block 10	40	True
E0003	LCD	MT	1-unit	90	True
E0004	Main Asus MK1234	HT	1-unit	78	True
E0005	Main Gigabyte GM34A	HT	1-unit	67	False
E0006	Laptop Compaq 6250	HT	1-unit	620	True
E0007	Blank DVD Giga	TA	block-100	43	True
E0008	Blank CD BW	TA	block-100	15	True
E0009	USB 2.0 Kingston- 4GB	TA	unit-1	10	False



```
public class DemoConnection {

    public static void main(String[] args) {

        String driver="com.microsoft.sqlserver.jdbc.SQLServerDriver";
        Connection c = null;
        String sql="SELECT * FROM Items";
        String url = "jdbc:sqlserver://127.0.0.1:1433;"
                + "database=ItemDB;"
                + "user=sa;"
                + "password=123;";
        try {
            Class.forName(driver);
            c = DriverManager.getConnection(url);
            Statement stm = c.createStatement();
            ResultSet result = stm.executeQuery(sql);
            while (result.next()) {
                System.out.print(result.getString("itemCode") + " ");
                System.out.print(result.getString("itemName") + " ");
                System.out.print(result.getString("supCode") + " ");
                System.out.print(result.getString("unit") + " ");
                System.out.print(result.getInt("price") + " ");
                System.out.print(result.getString("supplying") + " ");
                System.out.println();
            }
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    }

}
```

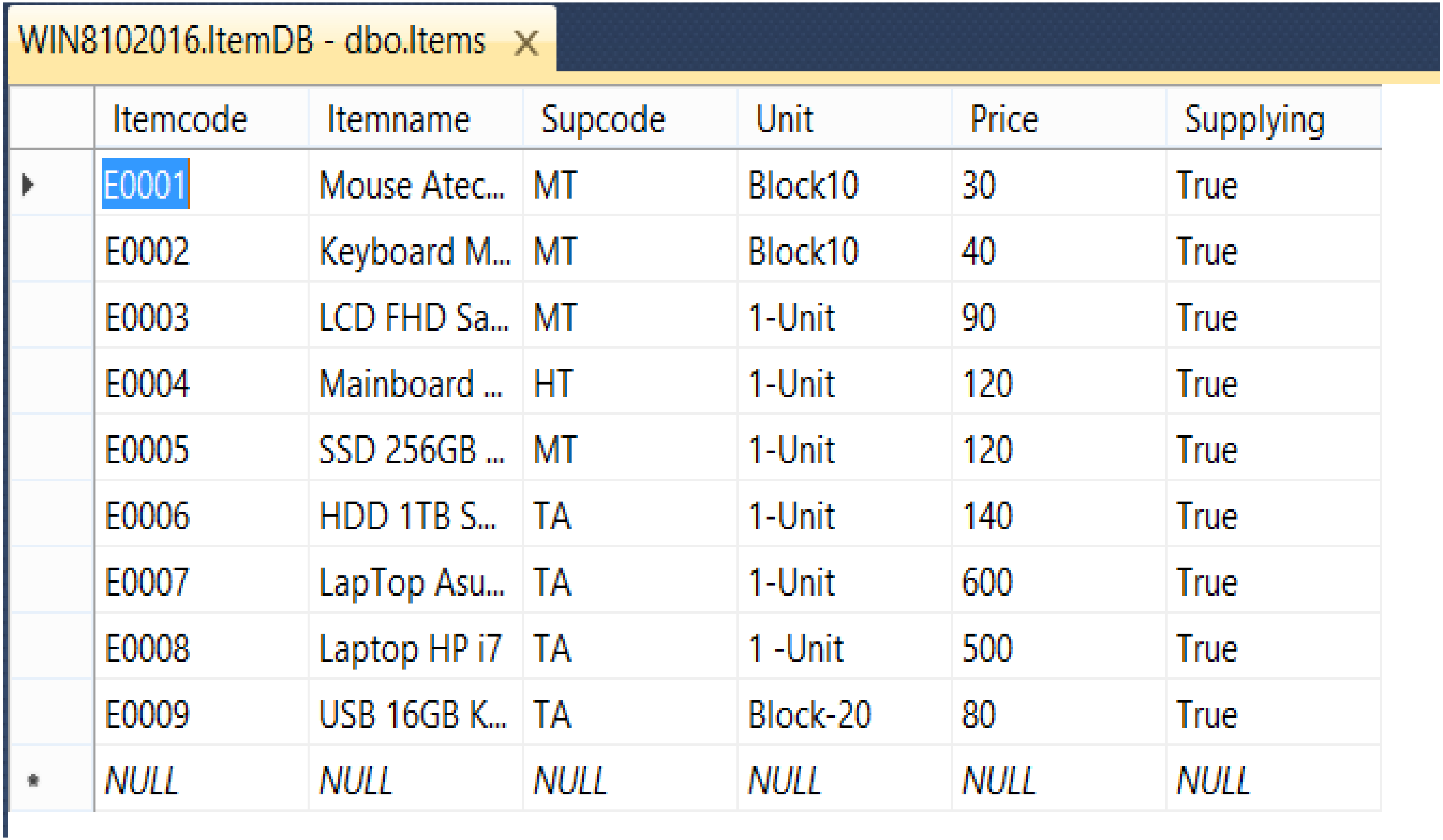


```
run:
E0001 Mouse Atech A4 MT Block10 30 1
E0002 Keyboard MitsuMi MT Block10 40 1
E0003 LCD FHD Samsung MT 1-Unit 90 1
E0004 Mainboard MSI HT 1-Unit 120 1
E0005 SSD 256GB KingSton HT 1-Unit 100 1
E0006 HDD 1TB Seagate TA 1-Unit 140 1
E0007 LapTop Asus i7 TA 1-Unit 600 1
E0008 Laptop HP i7 TA 1 -Unit 500 1
E0009 USB 16GB KingSton TA Block-20 80 1
BUILD SUCCESSFUL (total time: 0 seconds)
```



```java
public class DemoConnectionupdate {

    public static void main(String[] args) {

        String driver="com.microsoft.sqlserver.jdbc.SQLServerDriver";
        Connection c = null;
        PreparedStatement stmt = null;
        String sql="UPDATE Items set Supcode=?, Price=? where Itemcode='E0005'";
        String url = "jdbc:sqlserver://127.0.0.1:1433;"
                + "database=ItemDB;"
                + "user=sa;"
                + "password=123;";
        try {
            Class.forName(driver);
            c = DriverManager.getConnection(url);
            stmt = c.prepareStatement(sql);
            stmt.setString(1, "MT");  // Lenh nay se thiet lap Supcode
            stmt.setInt(2, 120); // Lenh nay se thiet lap Price
            int count=stmt.executeUpdate();

            if(count>0) System.out.println("Updated Success");

        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    }

}
```

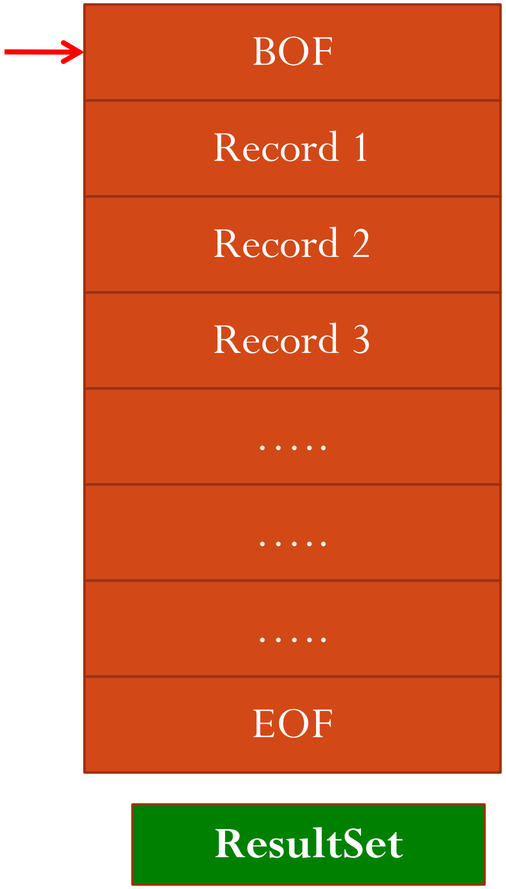
Result after update



WIN8102016.ItemDB - dbo.Items

	Itemcode	Itemname	Supcode	Unit	Price	Supplying
▶	E0001	Mouse Atec...	MT	Block10	30	True
	E0002	Keyboard M...	MT	Block10	40	True
	E0003	LCD FHD Sa...	MT	1-Unit	90	True
	E0004	Mainboard ...	HT	1-Unit	120	True
	E0005	SSD 256GB ...	MT	1-Unit	120	True
	E0006	HDD 1TB S...	TA	1-Unit	140	True
	E0007	LapTop Asu...	TA	1-Unit	600	True
	E0008	Laptop HP i7	TA	1 -Unit	500	True
	E0009	USB 16GB K...	TA	Block-20	80	True
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Step 4: Process the results



Move the current row:

boolean next(), previous(), first(), last()

Default: Result set moves forward only.

Get data in columns of the current row:

TYPE getTYPE (int columnIndex) // begin from 1

TYPE getTYPE (String columnLabel)

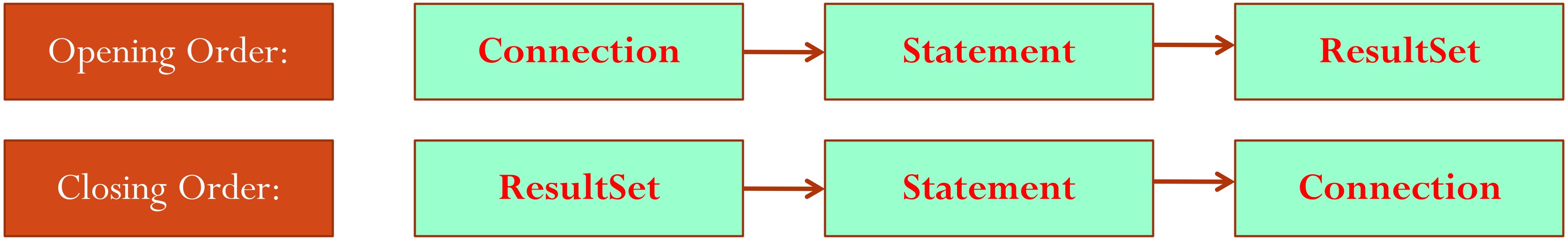
SELECT desc AS description FROM T_employee

➔Column name: desc

➔Column Label: description

At a time, resultset maintains a current position. When the resultset is initialized, the position is the BOF position. An exception is thrown when the current position is out of its scope.

Step 5: Close the connection



Opening Order: Connection → Statement → ResultSet

Closing Order: ResultSet → Statement → Connection

Attention!!!

At a time, a connection can be bound with ONLY ONE result set.

An exception will be thrown if we try binding a connection with another result set.

EX:

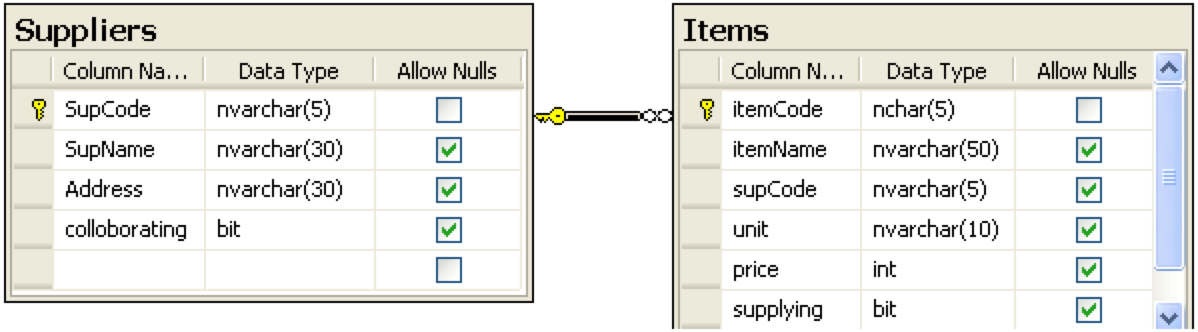
```
String sql1 ="SELECT…";
String sql2 ="SELECT…";
ResultSet rs1= stmt.executeQuery(sql1);
ResultSet rs2= stmt.executeQuery(sql2); → EXCEPTION
```

→You should close the rs1 before trying get the rs2 result set

→Solution: Transfer data in the rs1 to ArrayList (or Vector) then close rs1 before get new data to rs2.

(Demo 2) Develop the program for managing items using MS Sql Server JDBC

- Database:
- Program GUI

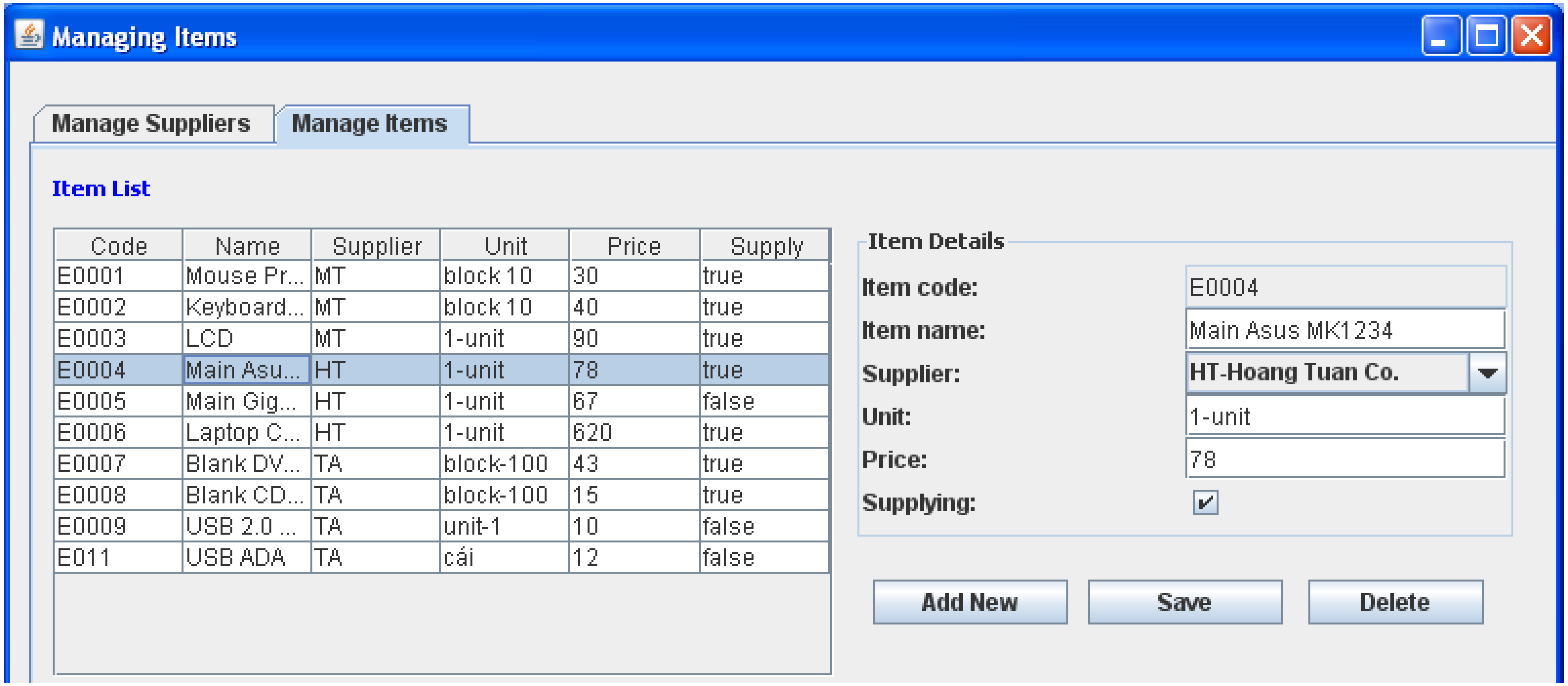


Suppliers

	Column Na...	Data Type	Allow Nulls
🔑	SupCode	nvarchar(5)	□
	SupName	nvarchar(30)	☑
	Address	nvarchar(30)	☑
	colloborating	bit	☑
			□

Items

	Column N...	Data Type	Allow Nulls
🔑	itemCode	nchar(5)	□
	itemName	nvarchar(50)	☑
	supCode	nvarchar(5)	☑
	unit	nvarchar(10)	☑
	price	int	☑
	supplying	bit	☑



Item List

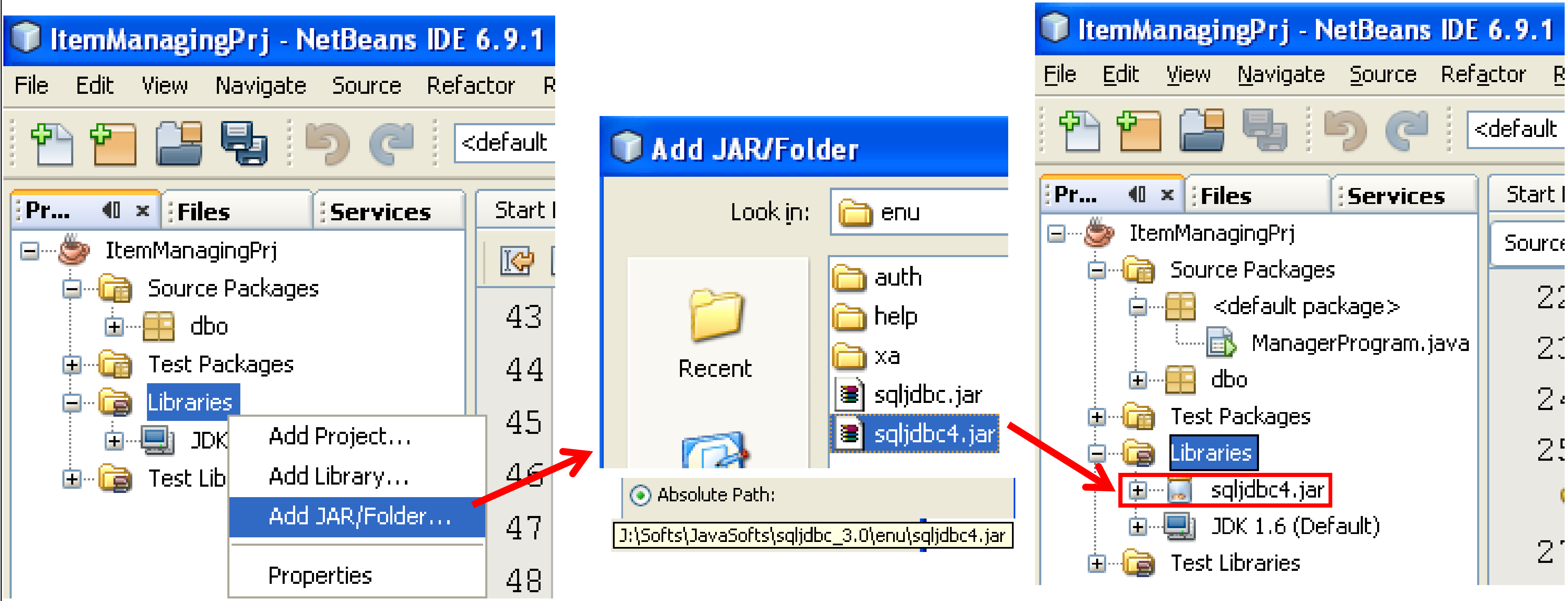
Code	Name	Supplier	Unit	Price	Supply
E0001	Mouse Pr...	MT	block 10	30	true
E0002	Keyboard...	MT	block 10	40	true
E0003	LCD	MT	1-unit	90	true
E0004	Main Asu...	HT	1-unit	78	true
E0005	Main Gig...	HT	1-unit	67	false
E0006	Laptop C...	HT	1-unit	620	true
E0007	Blank DV...	TA	block-100	43	true
E0008	Blank CD...	TA	block-100	15	true
E0009	USB 2.0 ...	TA	unit-1	10	false
E011	USB ADA	TA	cái	12	false

Item Details

Item code:	E0004
Item name:	Main Asus MK1234
Supplier:	HT-Hoang Tuan Co.
Unit:	1-unit
Price:	78
Supplying:	☑

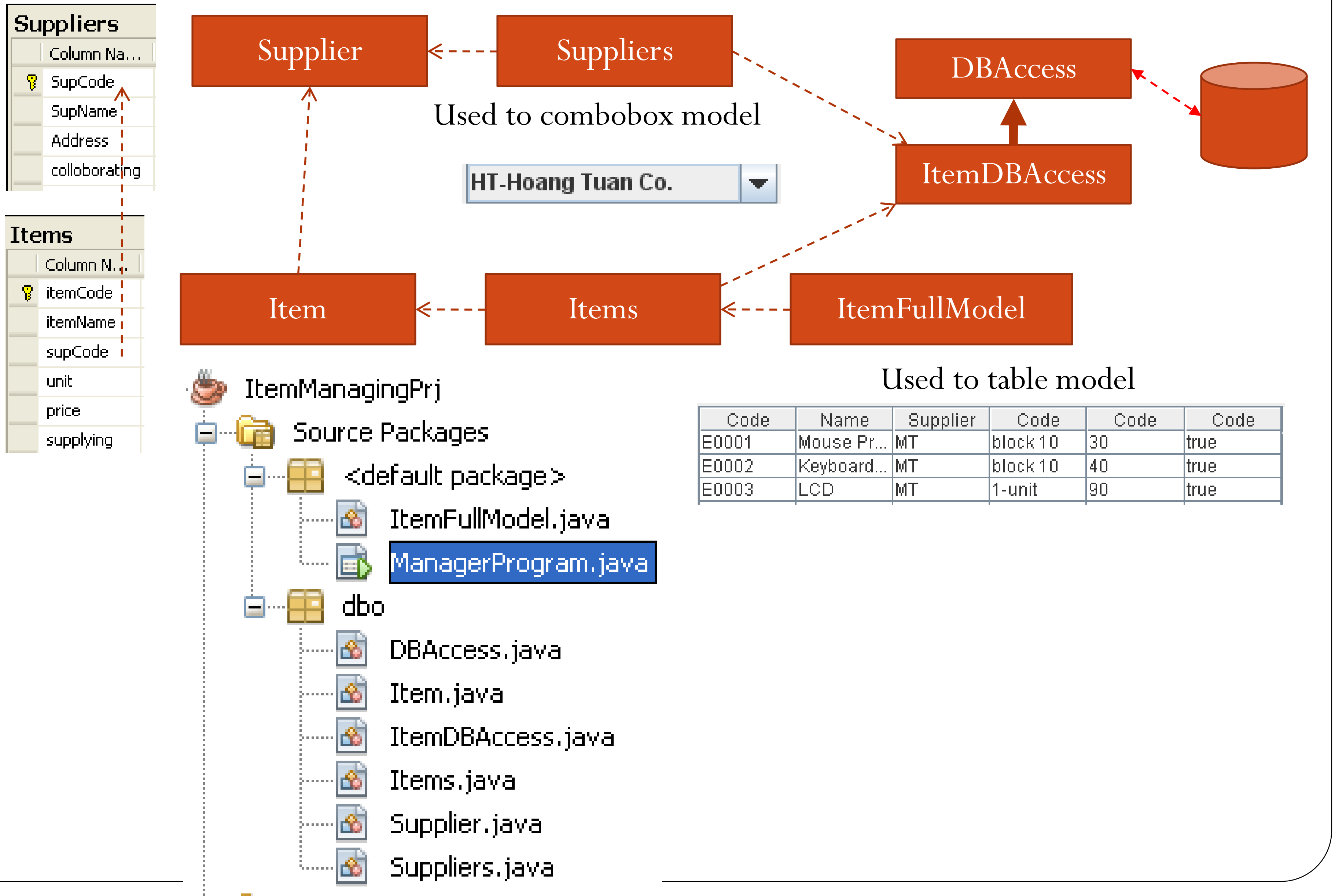
Demo 2…

- **Add MS SQL Server JDBC to the NetBeans:**



J:\Softs\JavaSofts\sqljdbc_3.0\enu\sqljdbc4.jar

Demo 2…: Class Diagram



Suppliers

	Column Na…
🔑	SupCode
	SupName
	Address
	colloborating

Items

	Column N…
🔑	itemCode
	itemName
	supCode
	unit
	price
	supplying

Used to combobox model

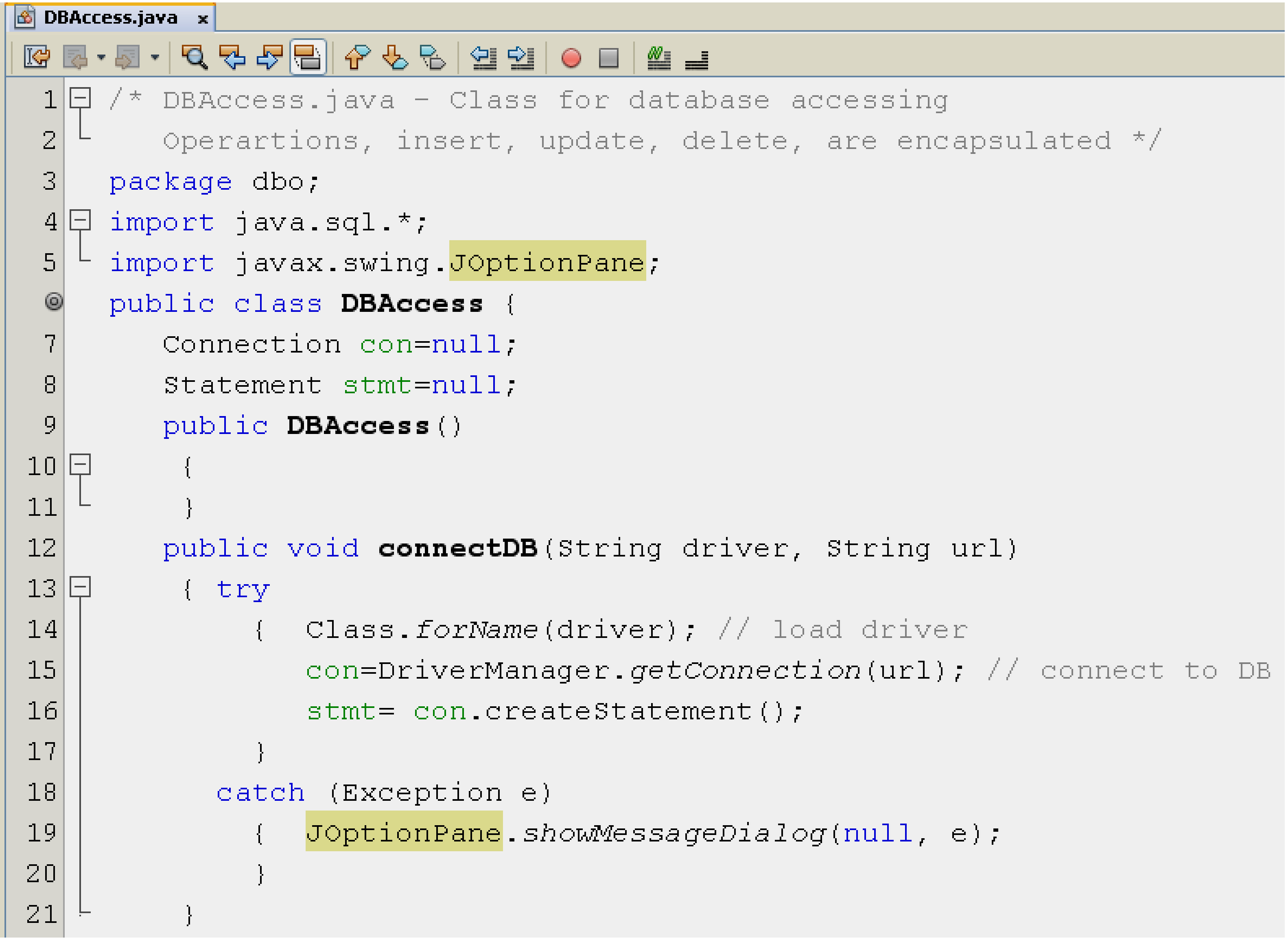
HT-Hoang Tuan Co.

Used to table model

Code	Name	Supplier	Code	Code	Code
E0001	Mouse Pr...	MT	block 10	30	true
E0002	Keyboard...	MT	block 10	40	true
E0003	LCD	MT	1-unit	90	true

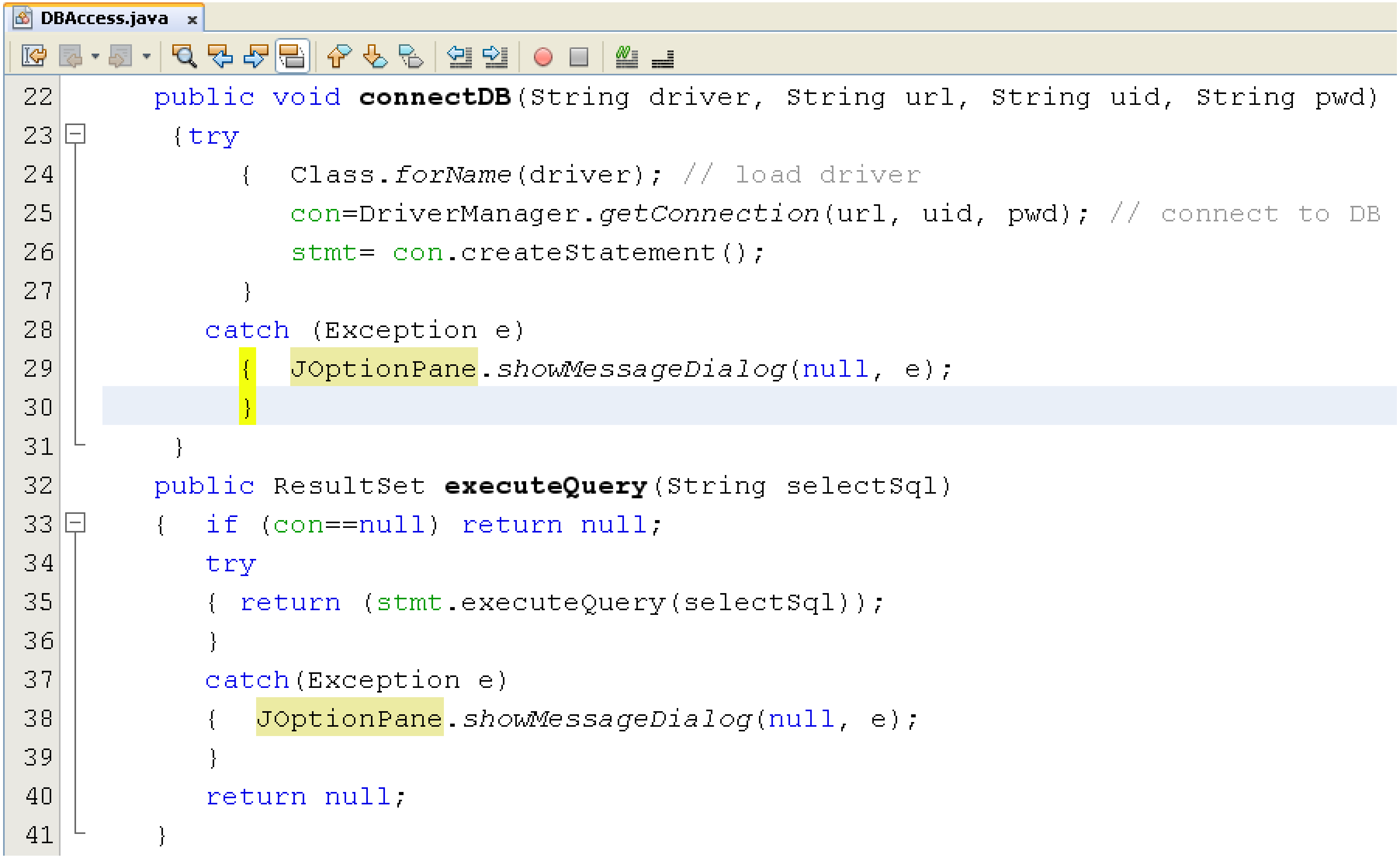
ItemManagingPrj
- Source Packages
  - <default package>
    - ItemFullModel.java
    - ManagerProgram.java
  - dbo
    - DBAccess.java
    - Item.java
    - ItemDBAccess.java
    - Items.java
    - Supplier.java
    - Suppliers.java

Demo 2…



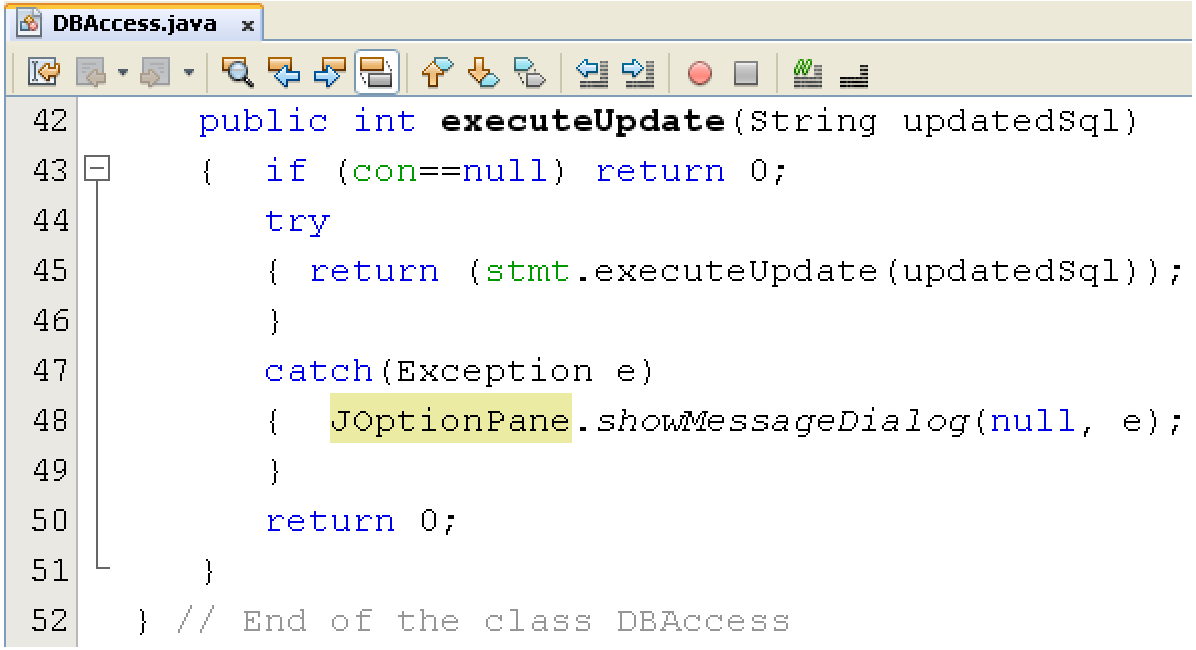
```java
/* DBAccess.java - Class for database accessing
   Operartions, insert, update, delete, are encapsulated */
package dbo;
import java.sql.*;
import javax.swing.JOptionPane;
public class DBAccess {
   Connection con=null;
   Statement stmt=null;
   public DBAccess()
    {
    }
   public void connectDB(String driver, String url)
    { try
        {  Class.forName(driver); // load driver
           con=DriverManager.getConnection(url); // connect to DB
           stmt= con.createStatement();
        }
      catch (Exception e)
        {  JOptionPane.showMessageDialog(null, e);
        }
    }
```

Demo 2…



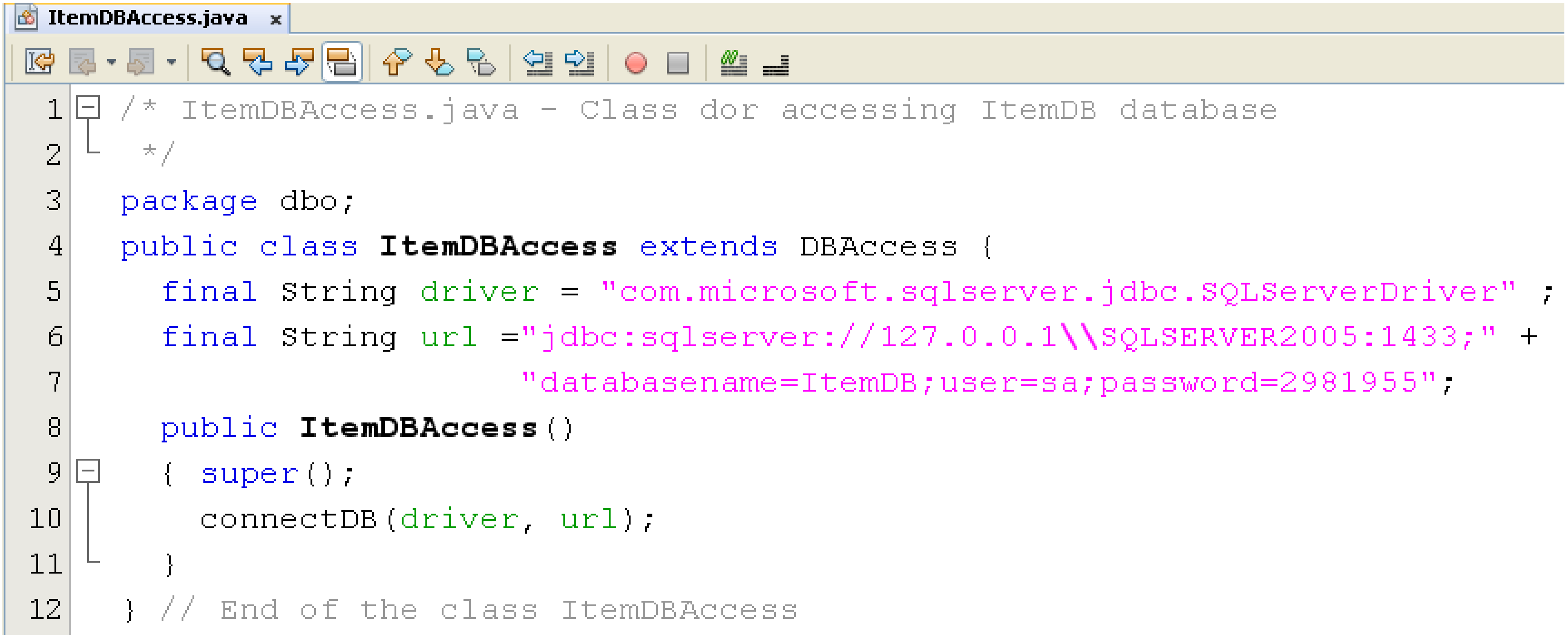
```
public void connectDB(String driver, String url, String uid, String pwd)
 {try
     {  Class.forName(driver); // load driver
        con=DriverManager.getConnection(url, uid, pwd); // connect to DB
        stmt= con.createStatement();
     }
   catch (Exception e)
     {  JOptionPane.showMessageDialog(null, e);
     }
 }
public ResultSet executeQuery(String selectSql)
{  if (con==null) return null;
   try
   { return (stmt.executeQuery(selectSql));
   }
   catch(Exception e)
   {  JOptionPane.showMessageDialog(null, e);
   }
   return null;
}
```

Demo 2…



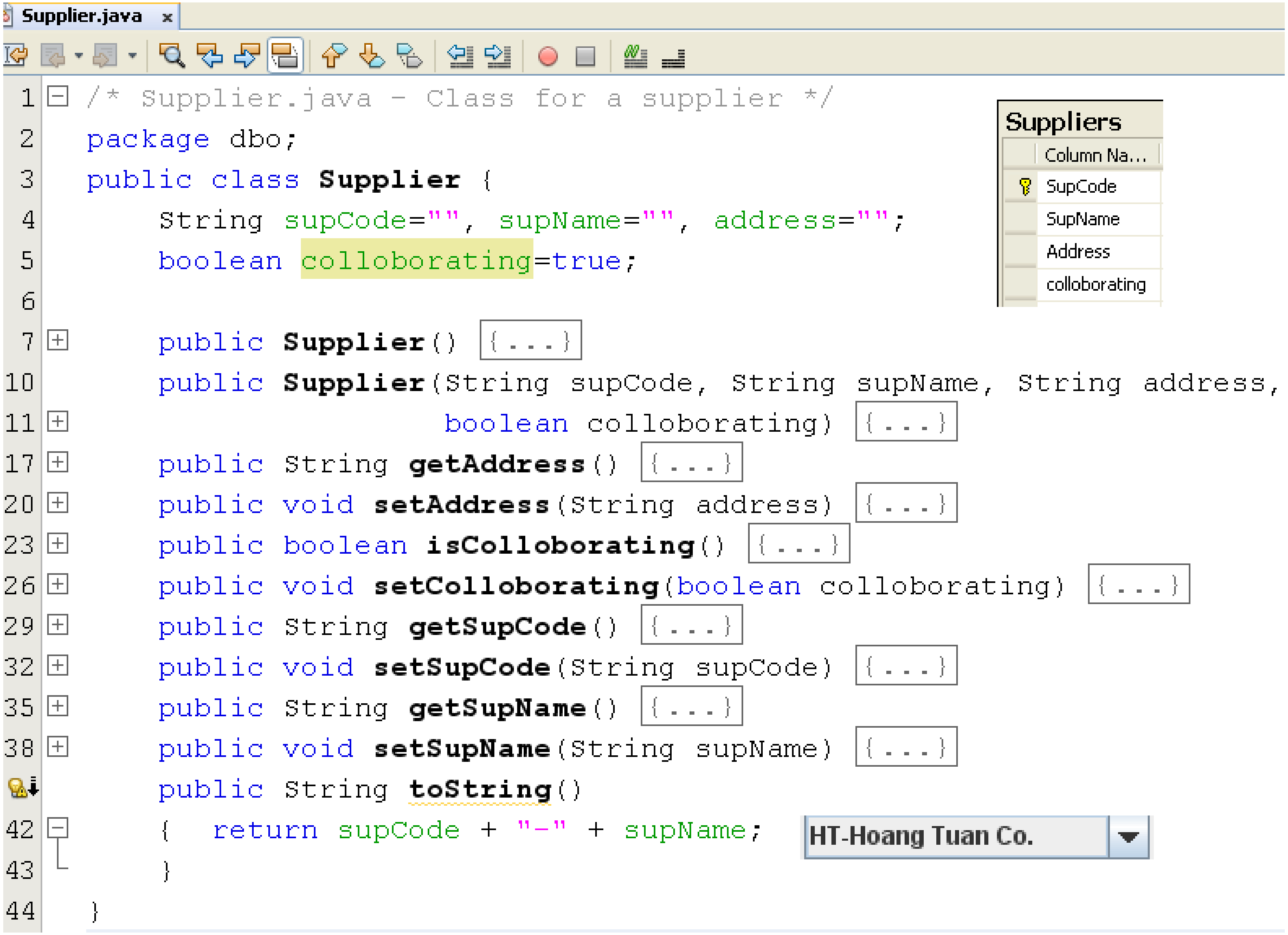
```
   public int executeUpdate(String updatedSql)
   {  if (con==null) return 0;
      try
      { return (stmt.executeUpdate(updatedSql));
      }
      catch(Exception e)
      {  JOptionPane.showMessageDialog(null, e);
      }
      return 0;
   }
} // End of the class DBAccess
```

Demo 2…



```java
/* ItemDBAccess.java - Class dor accessing ItemDB database
 */
package dbo;
public class ItemDBAccess extends DBAccess {
  final String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver" ;
  final String url ="jdbc:sqlserver://127.0.0.1\\SQLSERVER2005:1433;" +
                    "databasename=ItemDB;user=sa;password=2981955";
  public ItemDBAccess()
  {  super();
     connectDB(driver, url);
  }
} // End of the class ItemDBAccess
```

Demo 2…



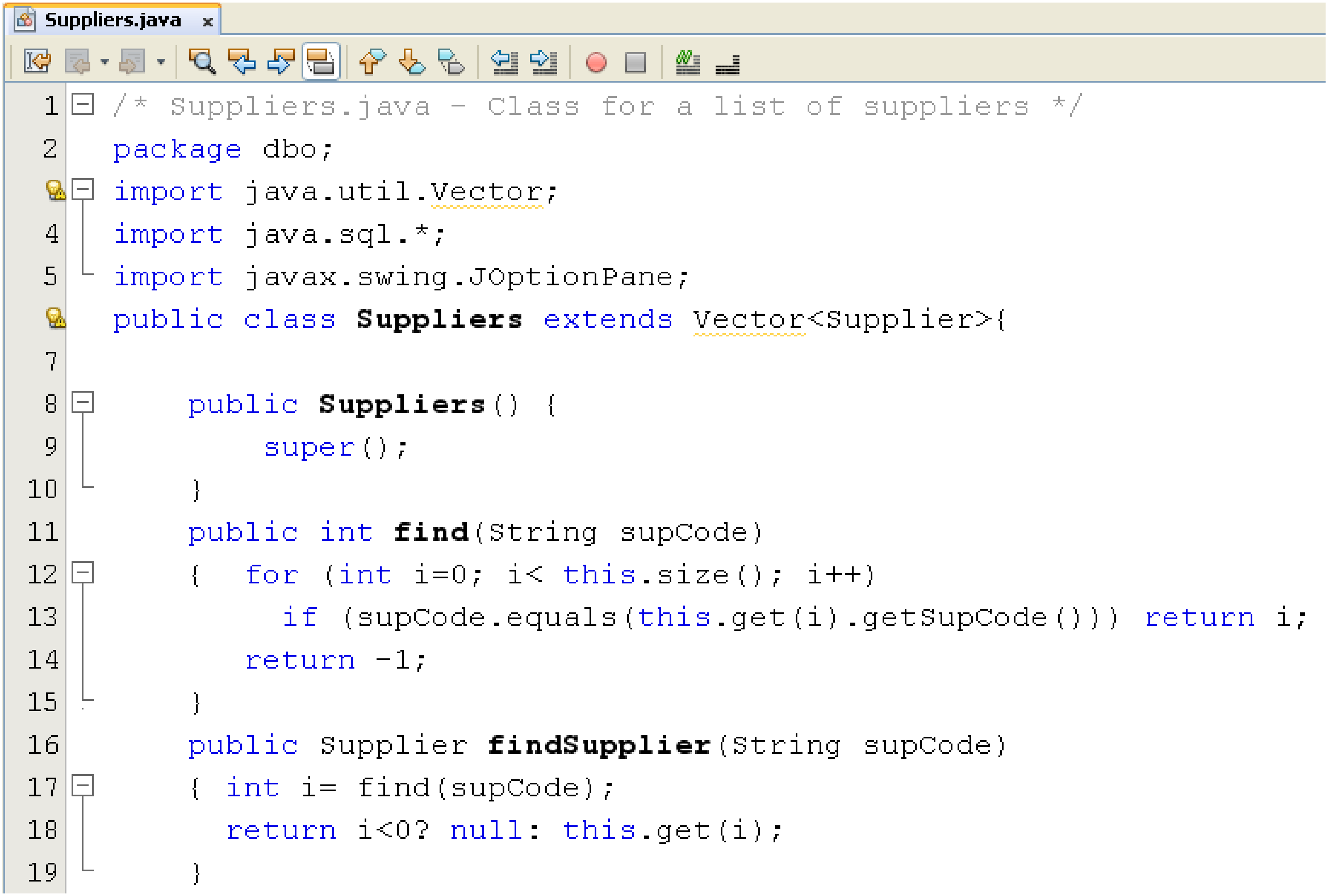
```
/* Supplier.java - Class for a supplier */
package dbo;
public class Supplier {
    String supCode="", supName="", address="";
    boolean colloborating=true;

    public Supplier() {...}
    public Supplier(String supCode, String supName, String address,
                    boolean colloborating) {...}
    public String getAddress() {...}
    public void setAddress(String address) {...}
    public boolean isColloborating() {...}
    public void setColloborating(boolean colloborating) {...}
    public String getSupCode() {...}
    public void setSupCode(String supCode) {...}
    public String getSupName() {...}
    public void setSupName(String supName) {...}
    public String toString()
    {   return supCode + "-" + supName;
    }
}
```

Suppliers

Column Na...
SupCode
SupName
Address
colloborating

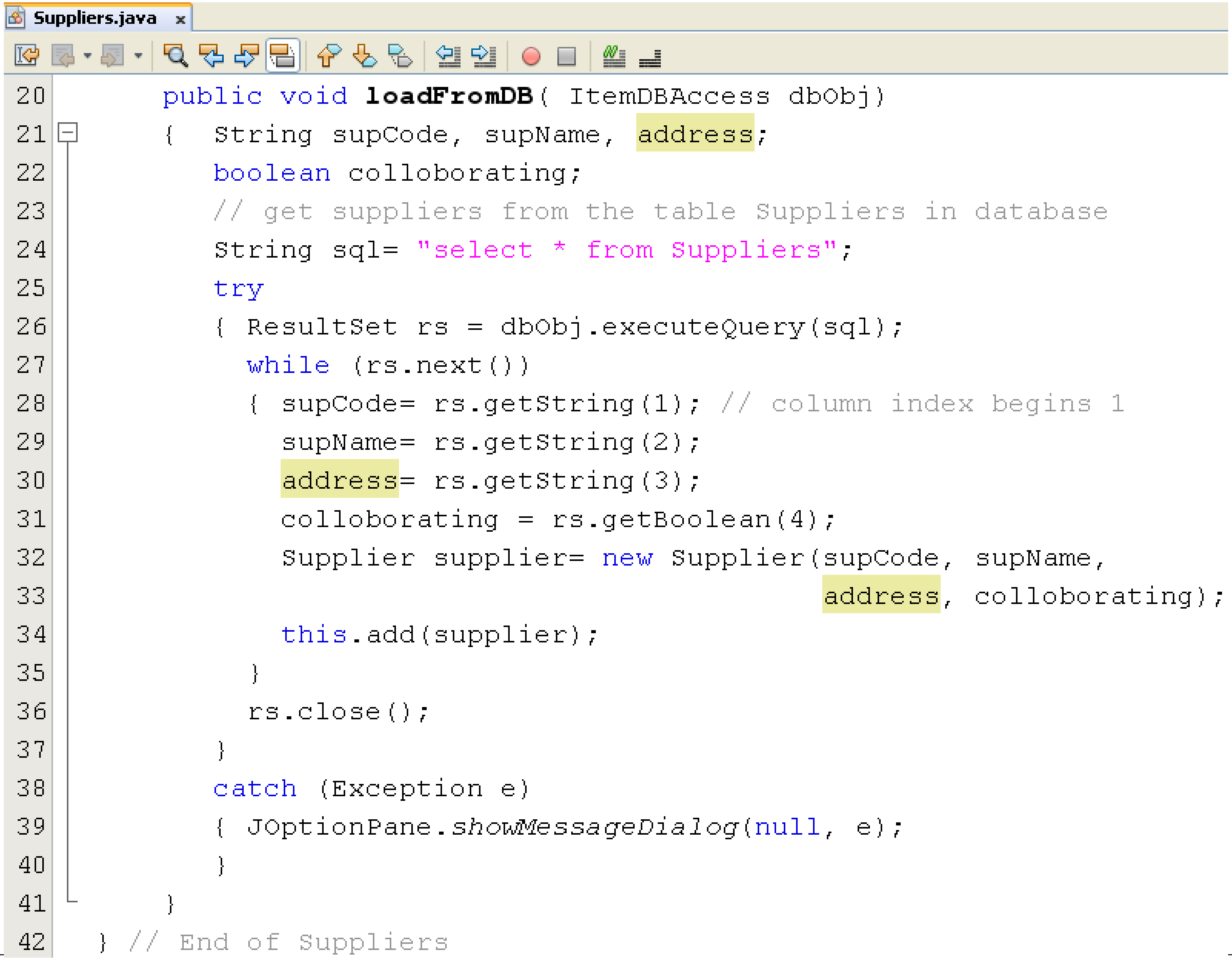
HT-Hoang Tuan Co.

Demo 2…



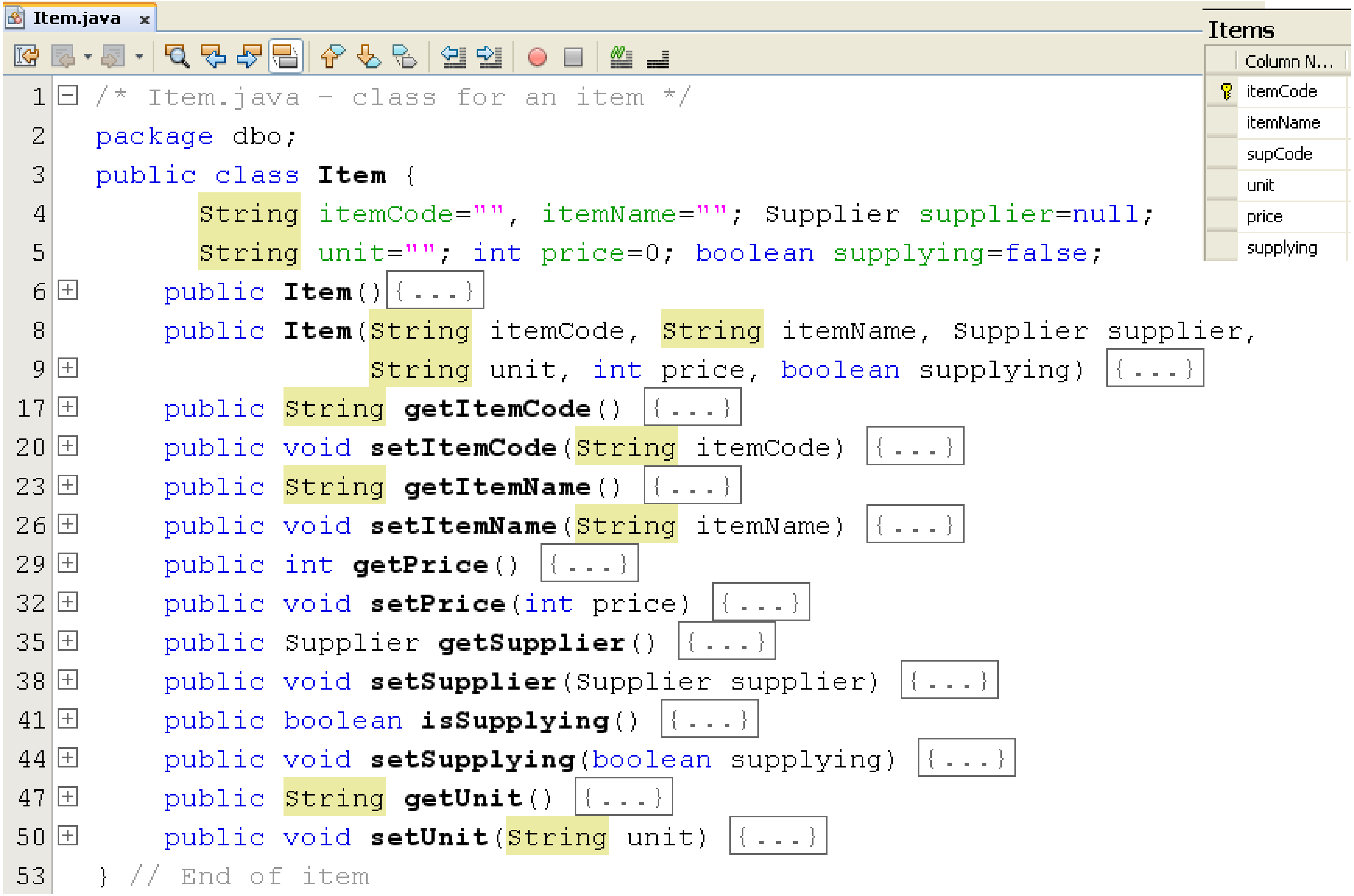
```
/* Suppliers.java - Class for a list of suppliers */
package dbo;
import java.util.Vector;
import java.sql.*;
import javax.swing.JOptionPane;
public class Suppliers extends Vector<Supplier>{

    public Suppliers() {
        super();
    }
    public int find(String supCode)
    {   for (int i=0; i< this.size(); i++)
         if (supCode.equals(this.get(i).getSupCode())) return i;
       return -1;
    }
    public Supplier findSupplier(String supCode)
    { int i= find(supCode);
      return i<0? null: this.get(i);
    }
```

Demo 2…



```
    public void loadFromDB( ItemDBAccess dbObj)
    {  String supCode, supName, address;
       boolean colloborating;
       // get suppliers from the table Suppliers in database
       String sql= "select * from Suppliers";
       try
       { ResultSet rs = dbObj.executeQuery(sql);
         while (rs.next())
         { supCode= rs.getString(1); // column index begins 1
           supName= rs.getString(2);
           address= rs.getString(3);
           colloborating = rs.getBoolean(4);
           Supplier supplier= new Supplier(supCode, supName,
                                           address, colloborating);
           this.add(supplier);
         }
         rs.close();
       }
       catch (Exception e)
       { JOptionPane.showMessageDialog(null, e);
       }
    }
} // End of Suppliers
```

Demo 2…

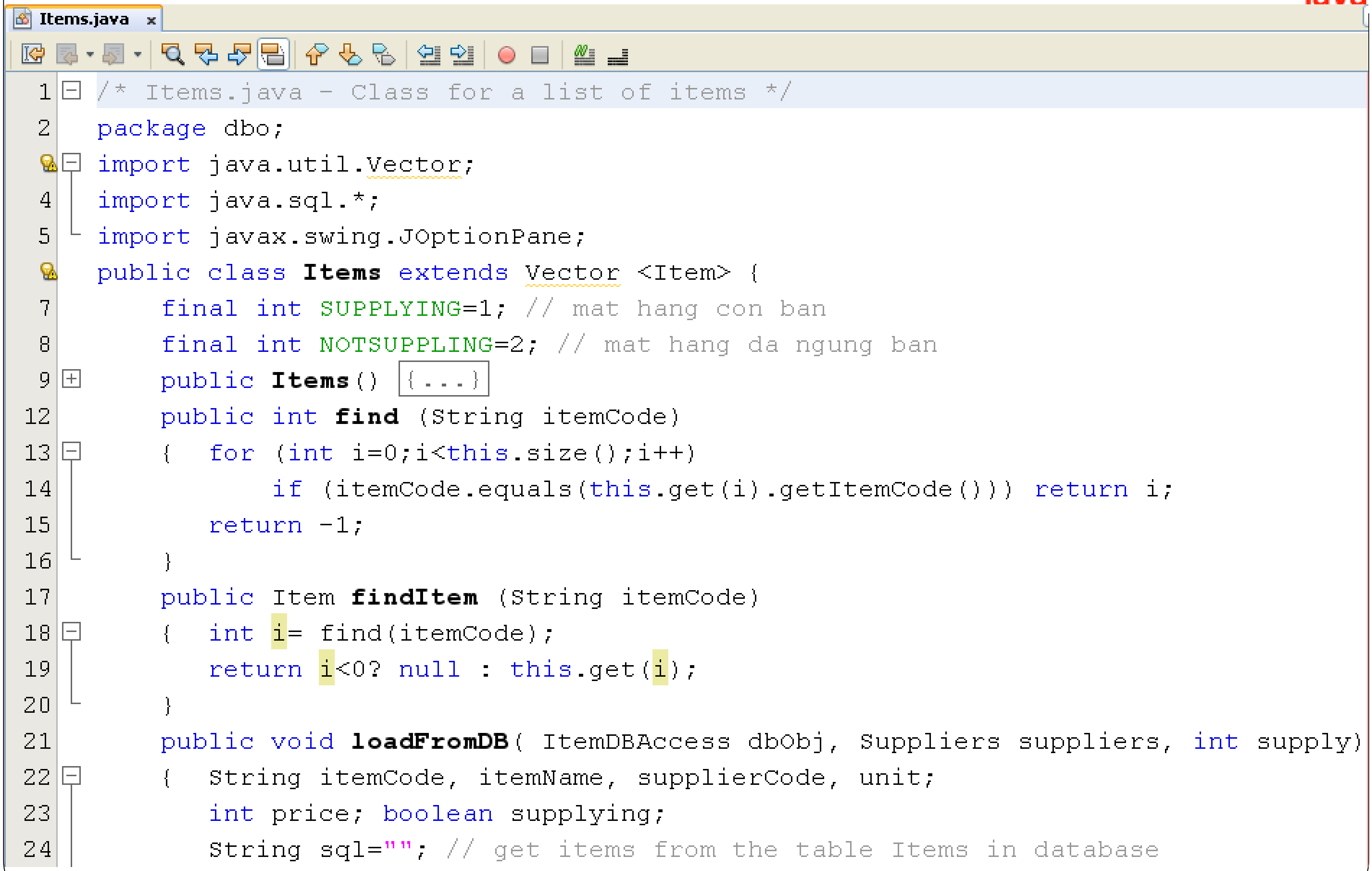


```
/* Item.java - class for an item */
package dbo;
public class Item {
        String itemCode="", itemName=""; Supplier supplier=null;
        String unit=""; int price=0; boolean supplying=false;
    public Item(){...}
    public Item(String itemCode, String itemName, Supplier supplier,
                String unit, int price, boolean supplying) {...}
    public String getItemCode() {...}
    public void setItemCode(String itemCode) {...}
    public String getItemName() {...}
    public void setItemName(String itemName) {...}
    public int getPrice() {...}
    public void setPrice(int price) {...}
    public Supplier getSupplier() {...}
    public void setSupplier(Supplier supplier) {...}
    public boolean isSupplying() {...}
    public void setSupplying(boolean supplying) {...}
    public String getUnit() {...}
    public void setUnit(String unit) {...}
} // End of item
```

Items

Column N...
itemCode
itemName
supCode
unit
price
supplying

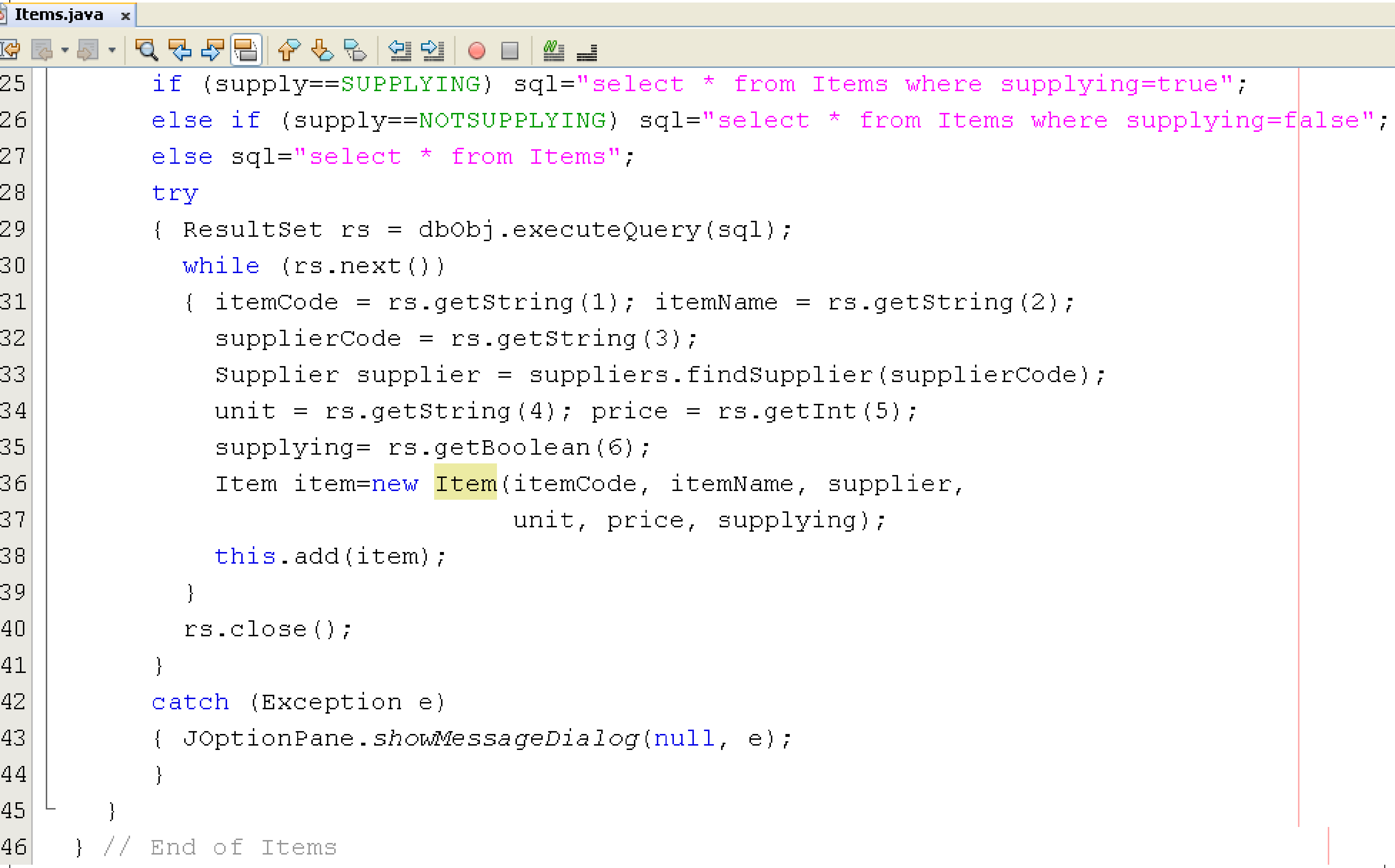
Demo 2…



Items.java

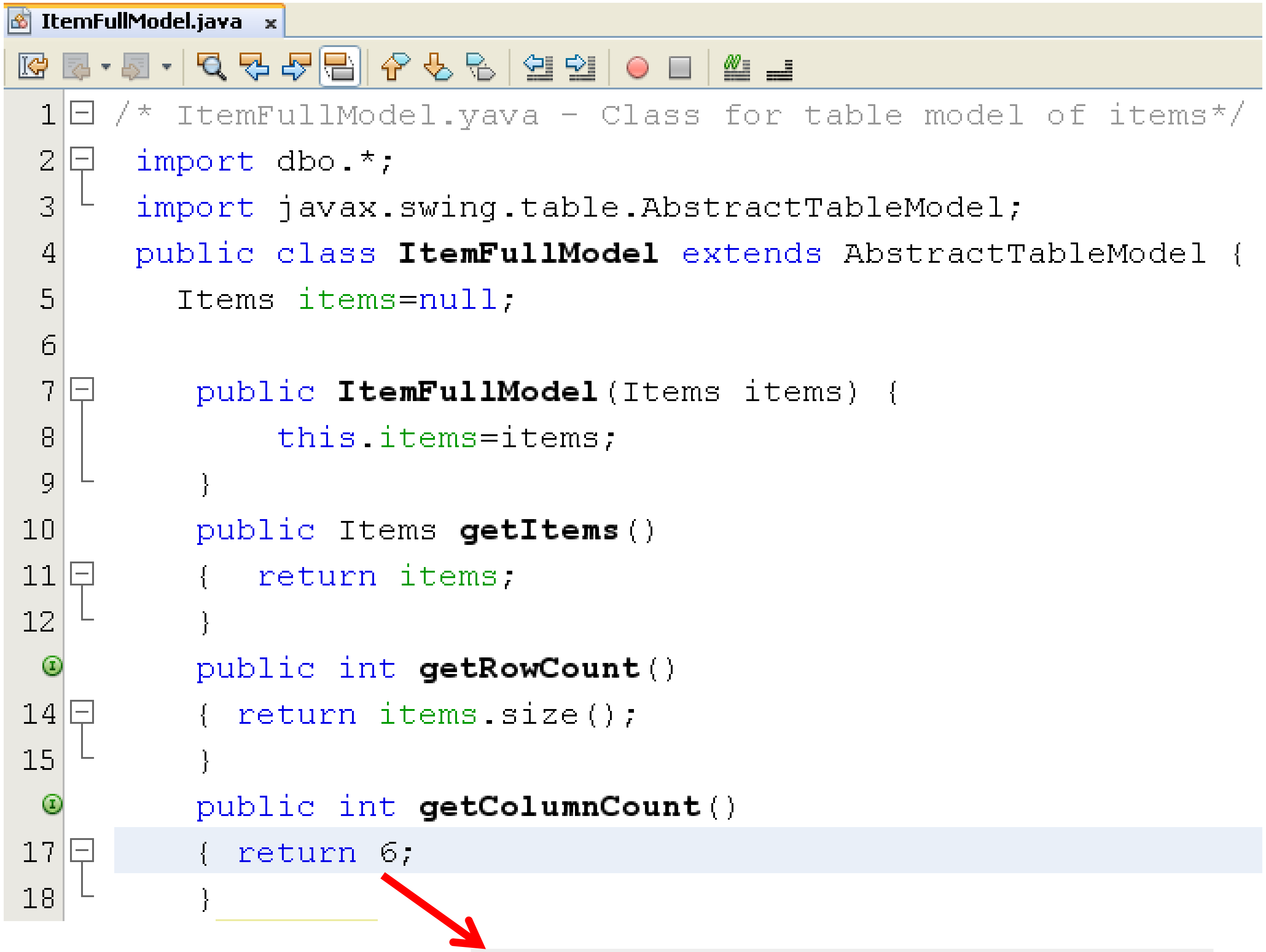
```
/* Items.java - Class for a list of items */
package dbo;
import java.util.Vector;
import java.sql.*;
import javax.swing.JOptionPane;
public class Items extends Vector <Item> {
    final int SUPPLYING=1; // mat hang con ban
    final int NOTSUPPLING=2; // mat hang da ngung ban
    public Items() {...}
    public int find (String itemCode)
    {   for (int i=0;i<this.size();i++)
           if (itemCode.equals(this.get(i).getItemCode())) return i;
        return -1;
    }
    public Item findItem (String itemCode)
    {   int i= find(itemCode);
        return i<0? null : this.get(i);
    }
    public void loadFromDB( ItemDBAccess dbObj, Suppliers suppliers, int supply)
    {   String itemCode, itemName, supplierCode, unit;
        int price; boolean supplying;
        String sql=""; // get items from the table Items in database
```

Demo 2…

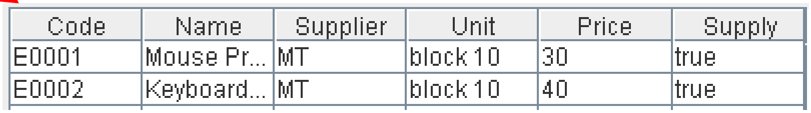


```
    if (supply==SUPPLYING) sql="select * from Items where supplying=true";
    else if (supply==NOTSUPPLYING) sql="select * from Items where supplying=false";
    else sql="select * from Items";
    try
    { ResultSet rs = dbObj.executeQuery(sql);
      while (rs.next())
      { itemCode = rs.getString(1); itemName = rs.getString(2);
        supplierCode = rs.getString(3);
        Supplier supplier = suppliers.findSupplier(supplierCode);
        unit = rs.getString(4); price = rs.getInt(5);
        supplying= rs.getBoolean(6);
        Item item=new Item(itemCode, itemName, supplier,
                           unit, price, supplying);
        this.add(item);
      }
      rs.close();
    }
    catch (Exception e)
    { JOptionPane.showMessageDialog(null, e);
    }
  }
} // End of Items
```

Demo 2…

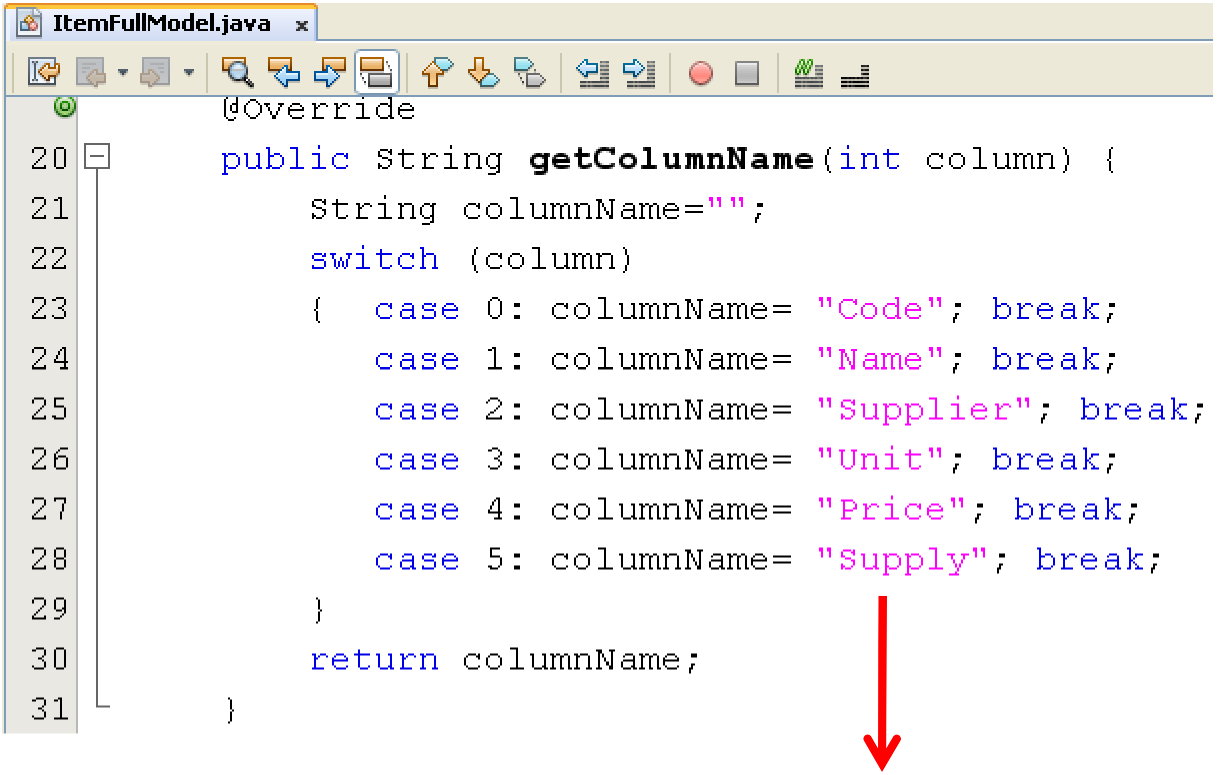


```
/* ItemFullModel.yava - Class for table model of items*/
 import dbo.*;
 import javax.swing.table.AbstractTableModel;
 public class ItemFullModel extends AbstractTableModel {
   Items items=null;

    public ItemFullModel(Items items) {
        this.items=items;
    }
    public Items getItems()
    {   return items;
    }
    public int getRowCount()
    {  return items.size();
    }
    public int getColumnCount()
    {  return 6;
    }
```

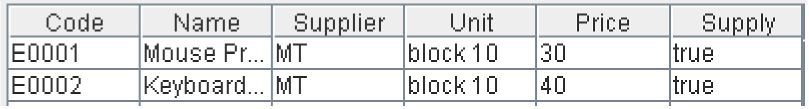


Code	Name	Supplier	Unit	Price	Supply
E0001	Mouse Pr...	MT	block 10	30	true
E0002	Keyboard...	MT	block 10	40	true

Demo 2…

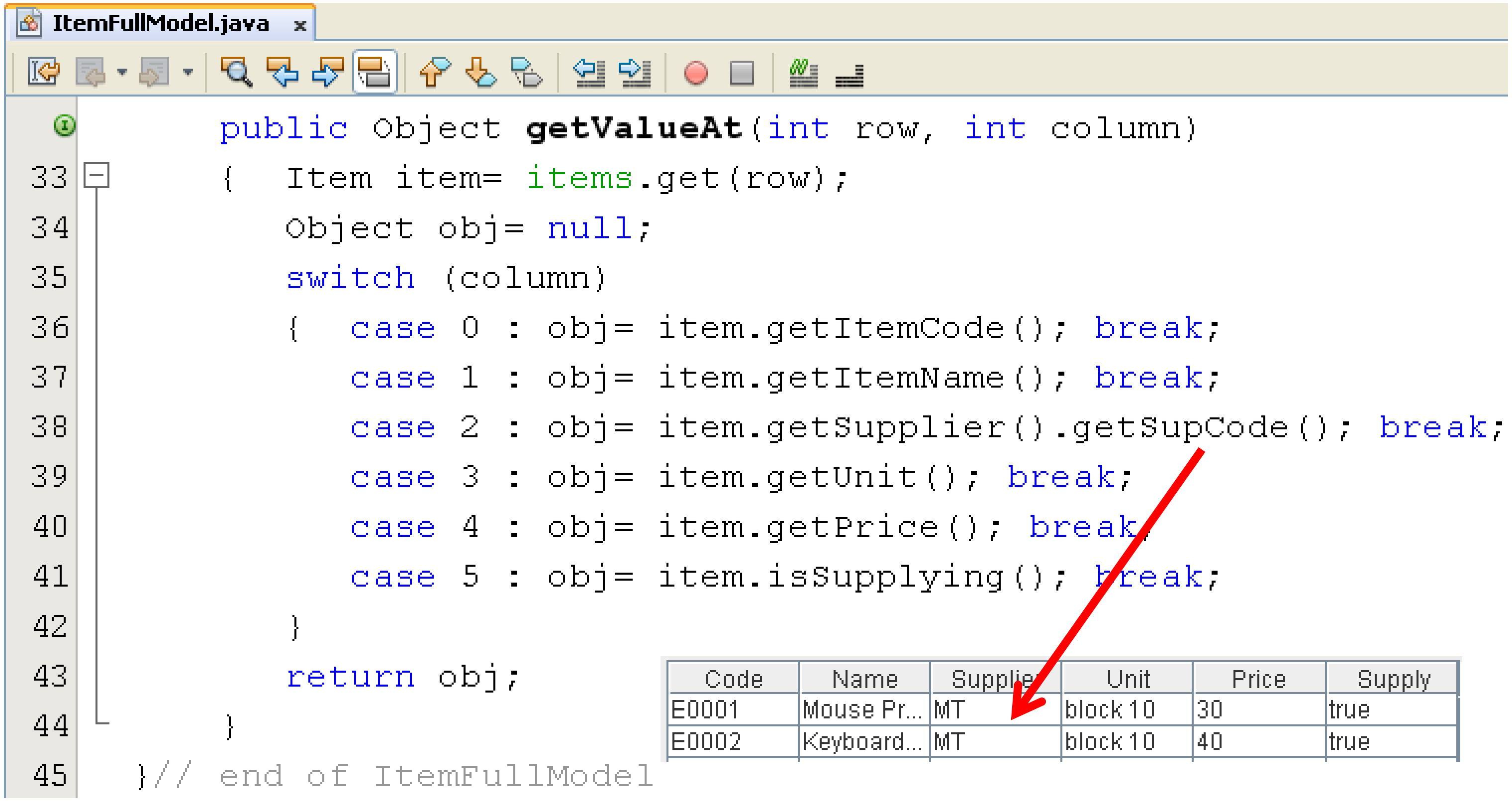


```
@Override
public String getColumnName(int column) {
    String columnName="";
    switch (column)
    {   case 0: columnName= "Code"; break;
        case 1: columnName= "Name"; break;
        case 2: columnName= "Supplier"; break;
        case 3: columnName= "Unit"; break;
        case 4: columnName= "Price"; break;
        case 5: columnName= "Supply"; break;
    }
    return columnName;
}
```



Code	Name	Supplier	Unit	Price	Supply
E0001	Mouse Pr...	MT	block 10	30	true
E0002	Keyboard...	MT	block 10	40	true

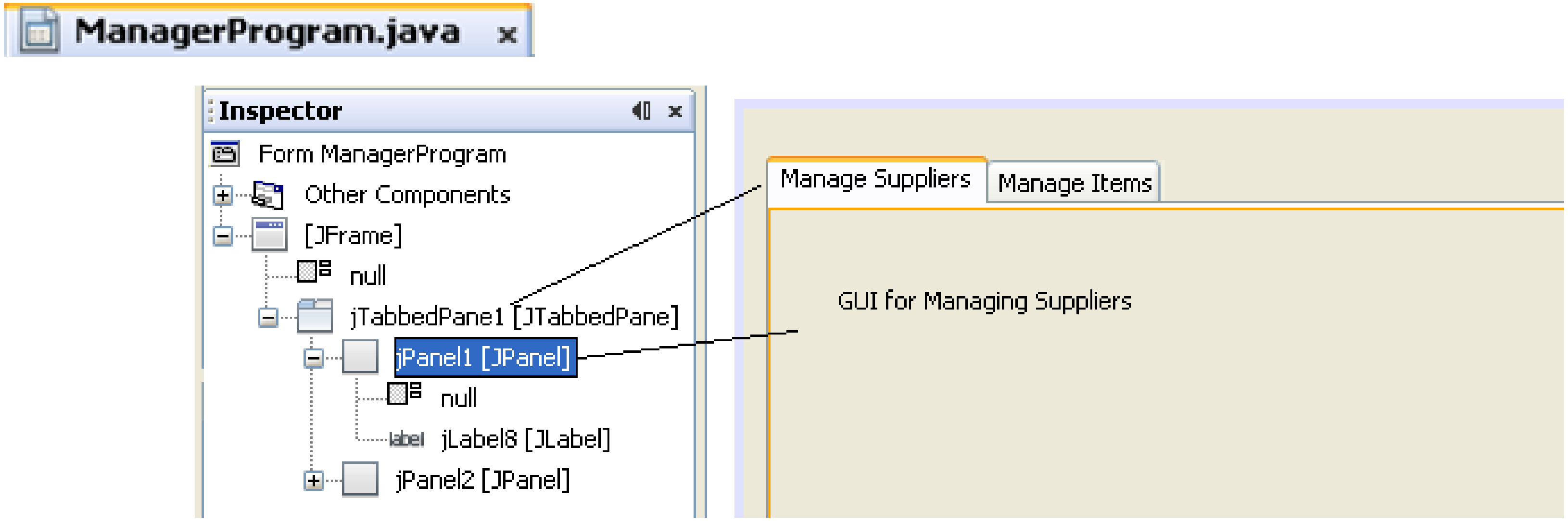
Demo 2...



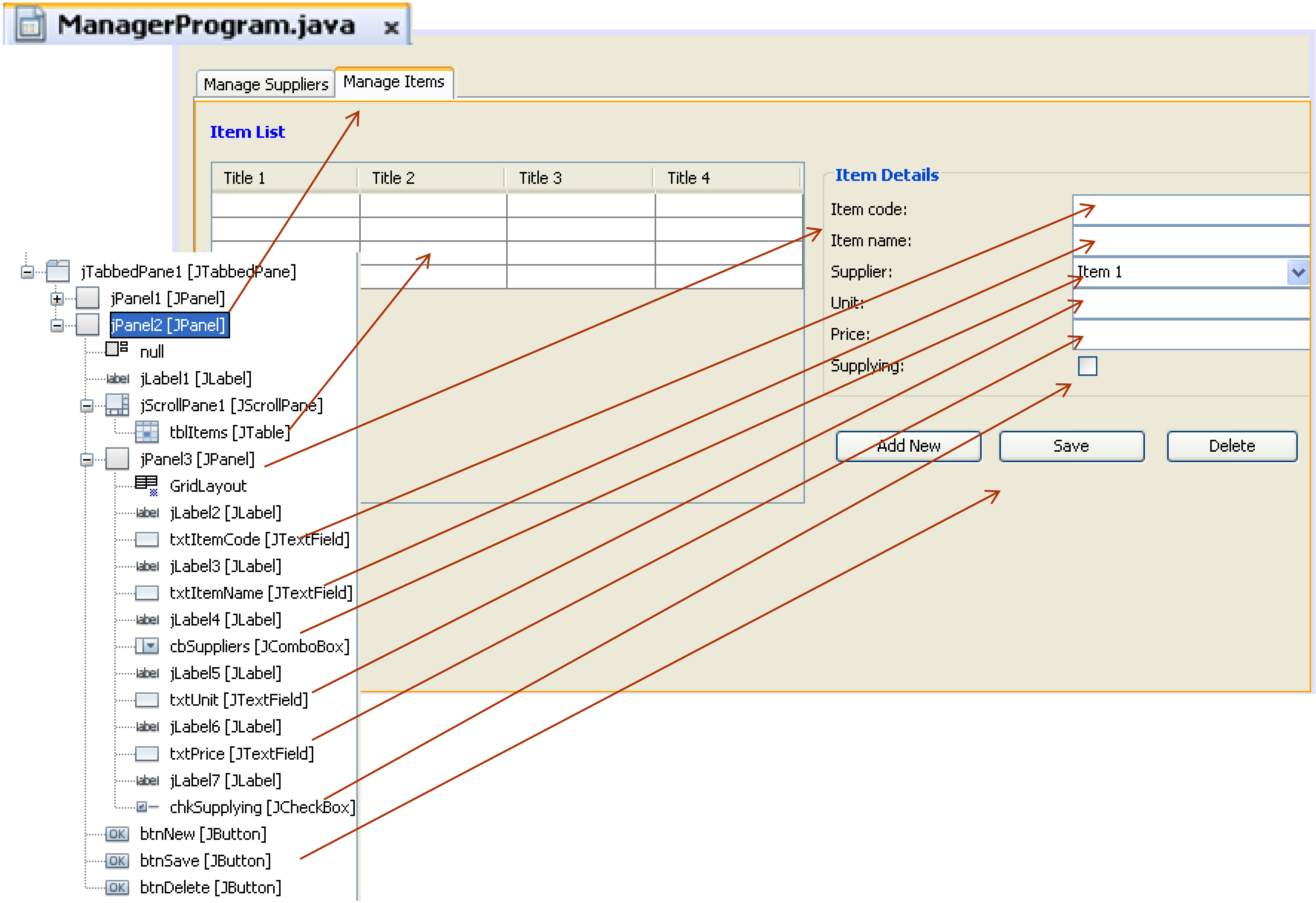
```
    public Object getValueAt(int row, int column)
    {   Item item= items.get(row);
        Object obj= null;
        switch (column)
        {   case 0 : obj= item.getItemCode(); break;
            case 1 : obj= item.getItemName(); break;
            case 2 : obj= item.getSupplier().getSupCode(); break;
            case 3 : obj= item.getUnit(); break;
            case 4 : obj= item.getPrice(); break;
            case 5 : obj= item.isSupplying(); break;
        }
        return obj;
    }
}// end of ItemFullModel
```

Code	Name	Supplier	Unit	Price	Supply
E0001	Mouse Pr...	MT	block 10	30	true
E0002	Keyboard...	MT	block 10	40	true

Demo 2…


ManagerProgram.java
Inspector
Form ManagerProgram
Other Components
[JFrame]
null
jTabbedPane1 [JTabbedPane]
jPanel1 [JPanel]
null
jLabel8 [JLabel]
jPanel2 [JPanel]
Manage Suppliers
Manage Items
GUI for Managing Suppliers

Demo 2…



ManagerProgram.java

Manage Suppliers | Manage Items

Item List

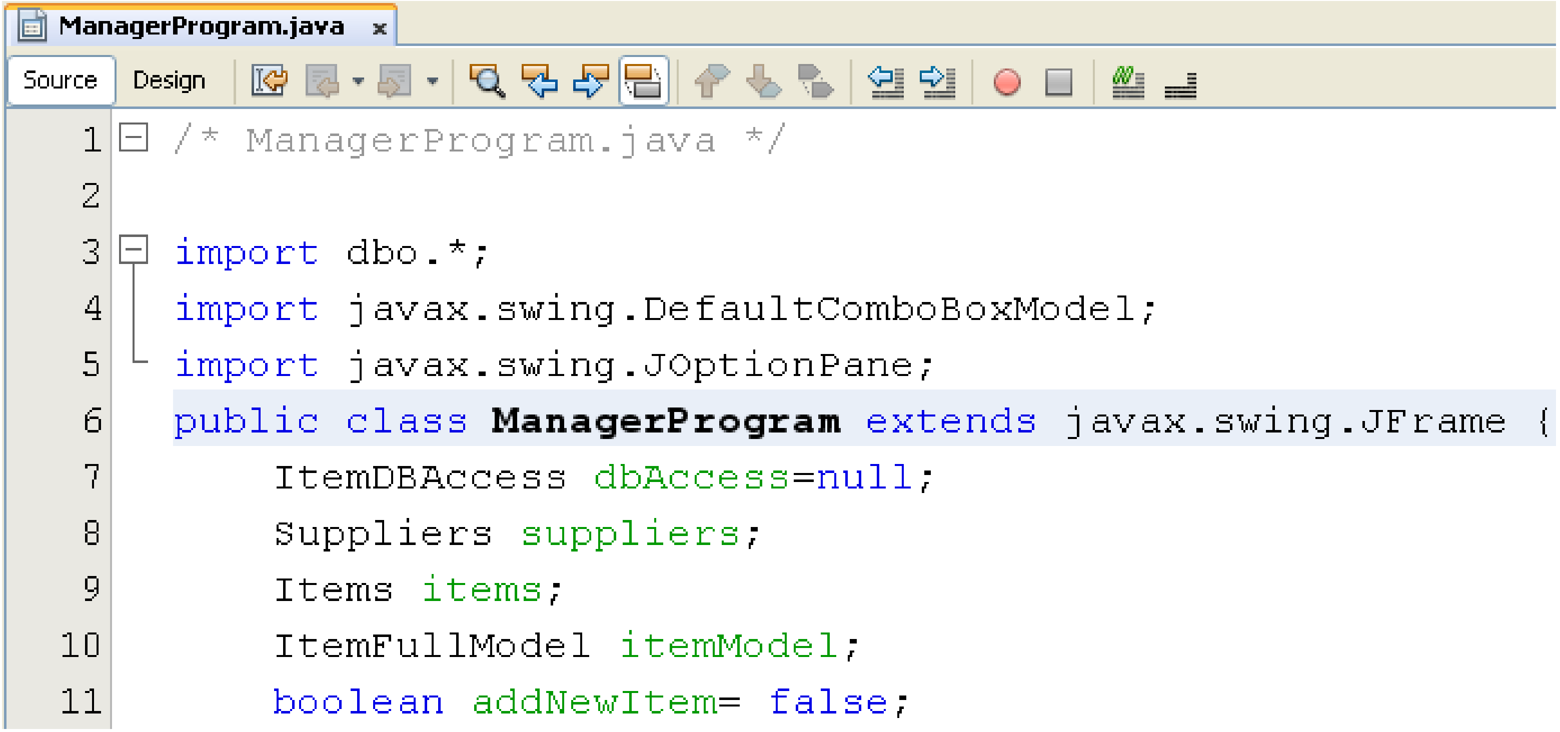
Title 1	Title 2	Title 3	Title 4

Item Details

Item code:	
Item name:	
Supplier:	Item 1
Unit:	
Price:	
Supplying:	□

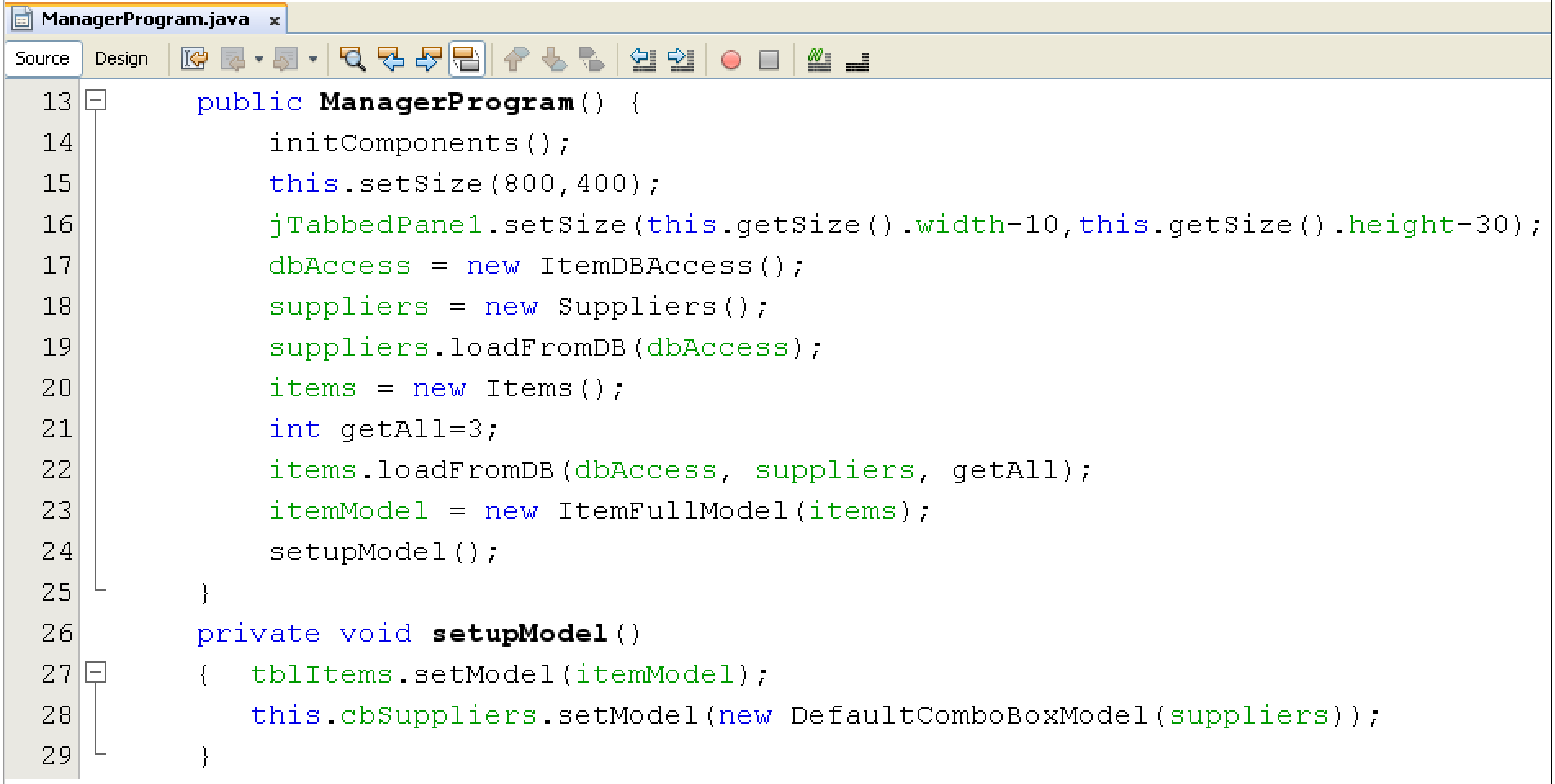
```
jTabbedPane1 [JTabbedPane]
    jPanel1 [JPanel]
    jPanel2 [JPanel]
        null
        jLabel1 [JLabel]
        jScrollPane1 [JScrollPane]
            tblItems [JTable]
        jPanel3 [JPanel]
            GridLayout
            jLabel2 [JLabel]
            txtItemCode [JTextField]
            jLabel3 [JLabel]
            txtItemName [JTextField]
            jLabel4 [JLabel]
            cbSuppliers [JComboBox]
            jLabel5 [JLabel]
            txtUnit [JTextField]
            jLabel6 [JLabel]
            txtPrice [JTextField]
            jLabel7 [JLabel]
            chkSupplying [JCheckBox]
        btnNew [JButton]
        btnSave [JButton]
        btnDelete [JButton]
```

Demo 2…



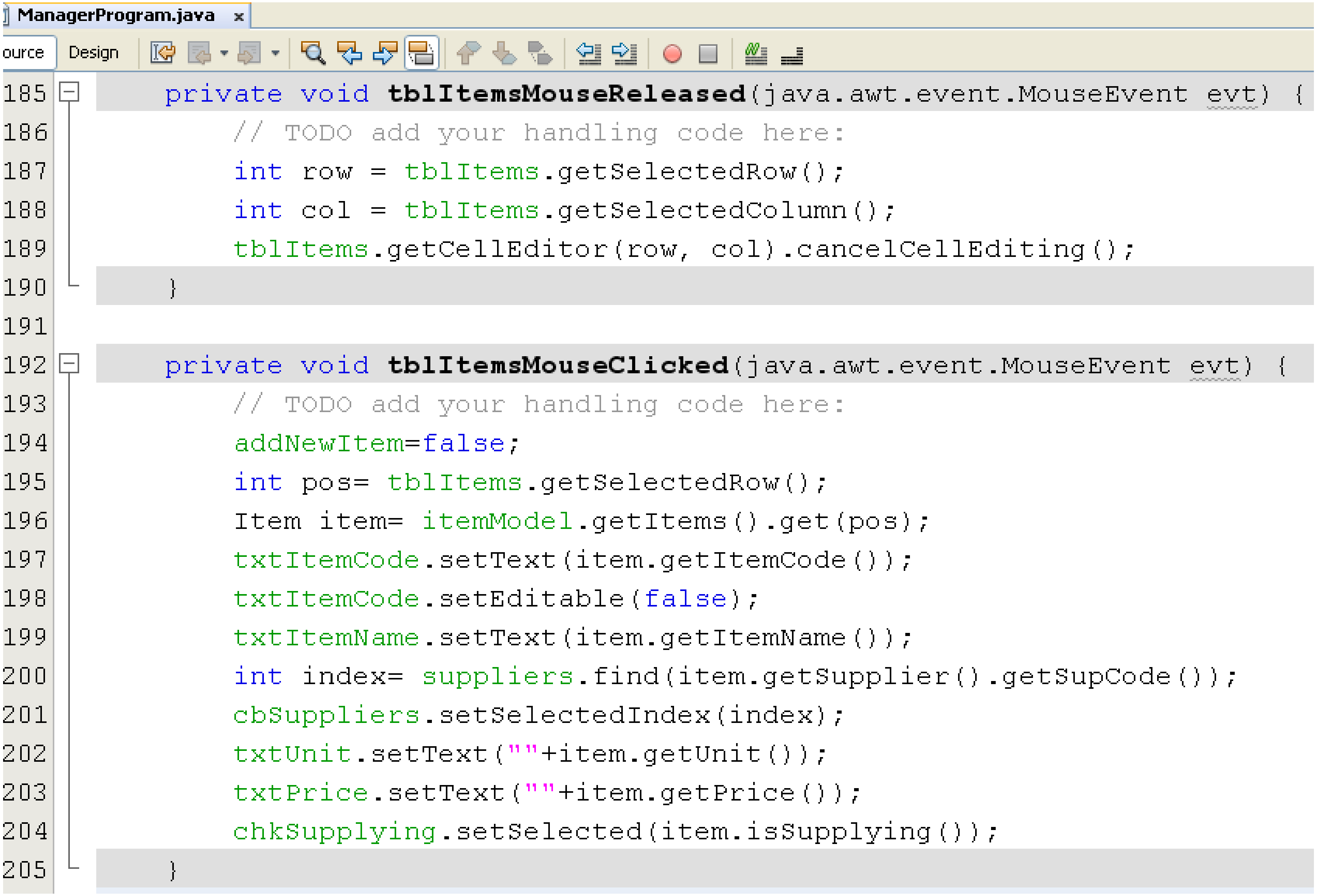
```
/* ManagerProgram.java */

import dbo.*;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JOptionPane;
public class ManagerProgram extends javax.swing.JFrame {
    ItemDBAccess dbAccess=null;
    Suppliers suppliers;
    Items items;
    ItemFullModel itemModel;
    boolean addNewItem= false;
```

Demo 2…



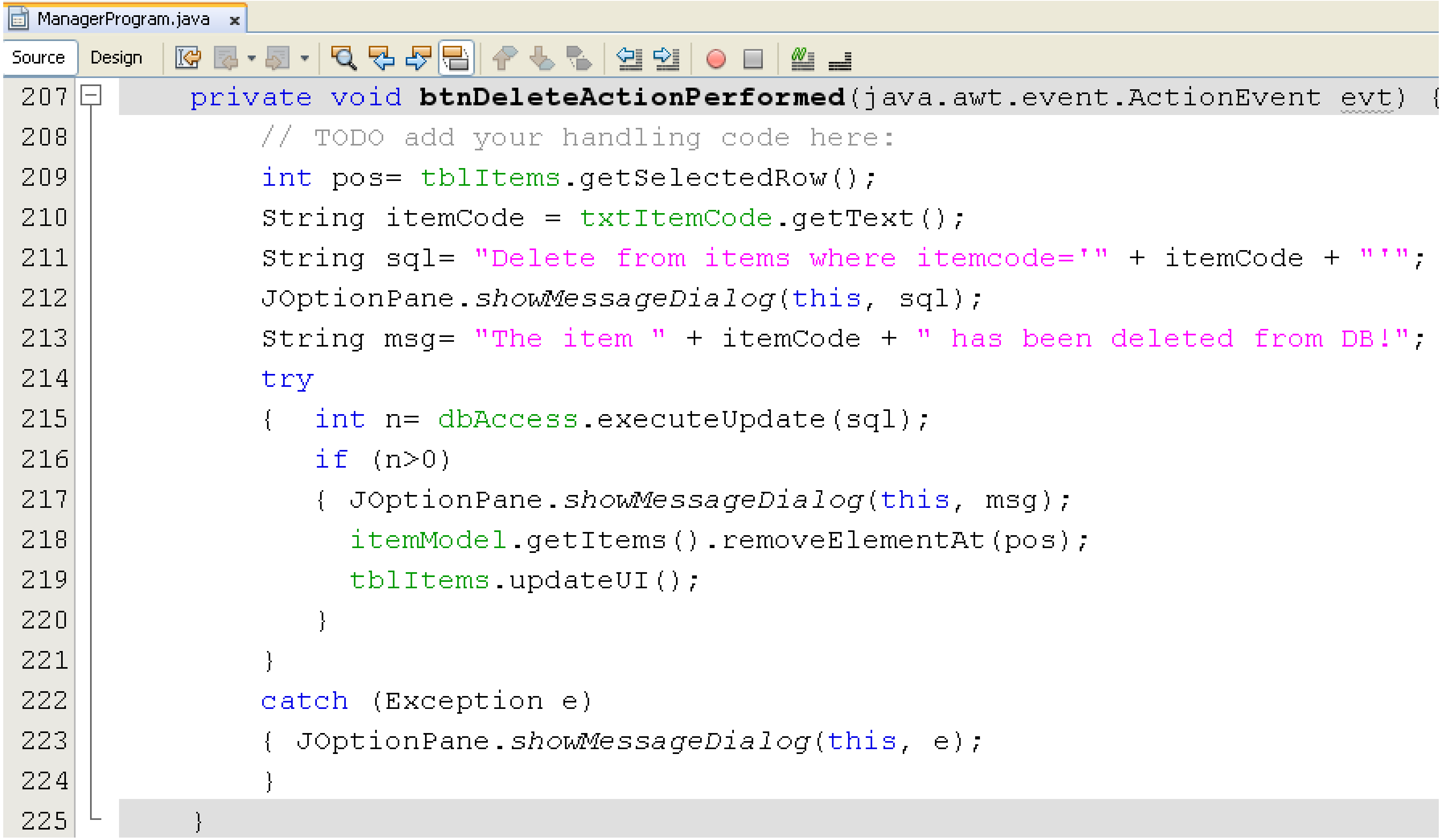
```java
public ManagerProgram() {
    initComponents();
    this.setSize(800,400);
    jTabbedPane1.setSize(this.getSize().width-10,this.getSize().height-30);
    dbAccess = new ItemDBAccess();
    suppliers = new Suppliers();
    suppliers.loadFromDB(dbAccess);
    items = new Items();
    int getAll=3;
    items.loadFromDB(dbAccess, suppliers, getAll);
    itemModel = new ItemFullModel(items);
    setupModel();
}
private void setupModel()
{  tblItems.setModel(itemModel);
   this.cbSuppliers.setModel(new DefaultComboBoxModel(suppliers));
}
```

Demo 3…



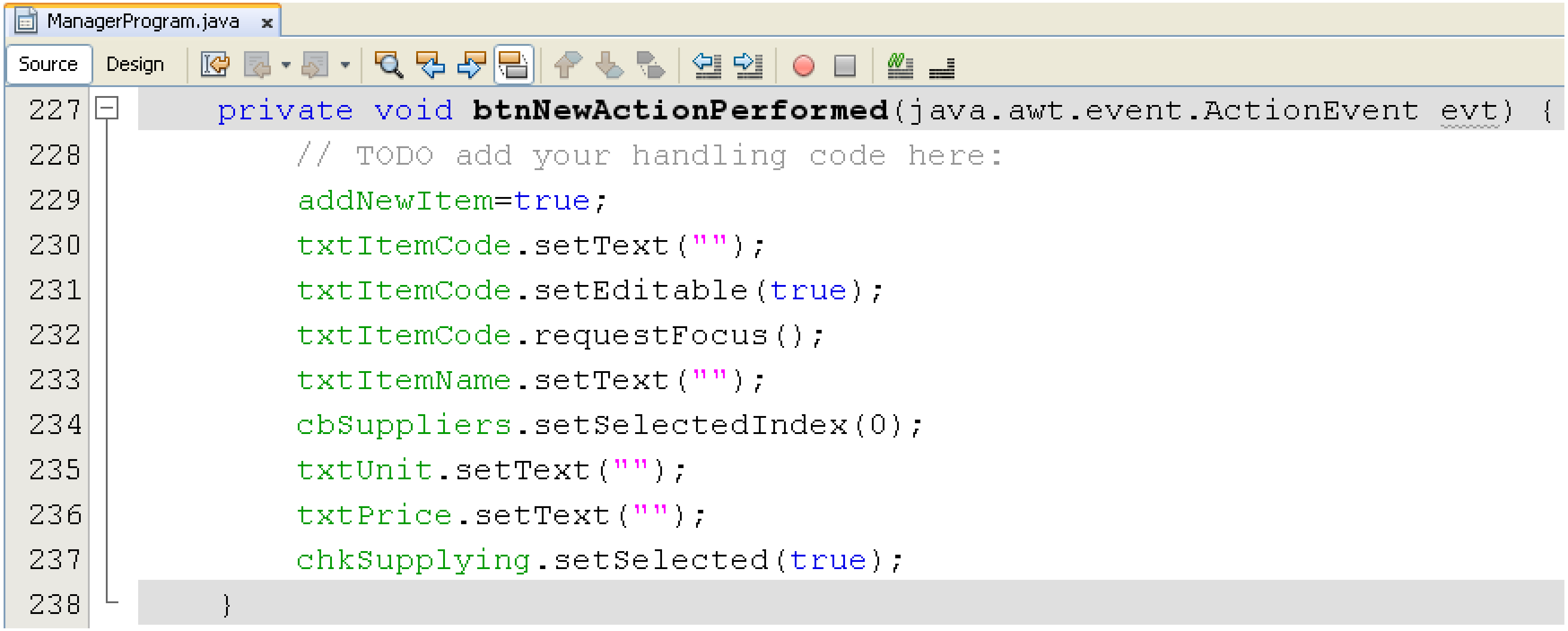
```java
    private void tblItemsMouseReleased(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        int row = tblItems.getSelectedRow();
        int col = tblItems.getSelectedColumn();
        tblItems.getCellEditor(row, col).cancelCellEditing();
    }

    private void tblItemsMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        addNewItem=false;
        int pos= tblItems.getSelectedRow();
        Item item= itemModel.getItems().get(pos);
        txtItemCode.setText(item.getItemCode());
        txtItemCode.setEditable(false);
        txtItemName.setText(item.getItemName());
        int index= suppliers.find(item.getSupplier().getSupCode());
        cbSuppliers.setSelectedIndex(index);
        txtUnit.setText(""+item.getUnit());
        txtPrice.setText(""+item.getPrice());
        chkSupplying.setSelected(item.isSupplying());
    }
```

Demo 2…



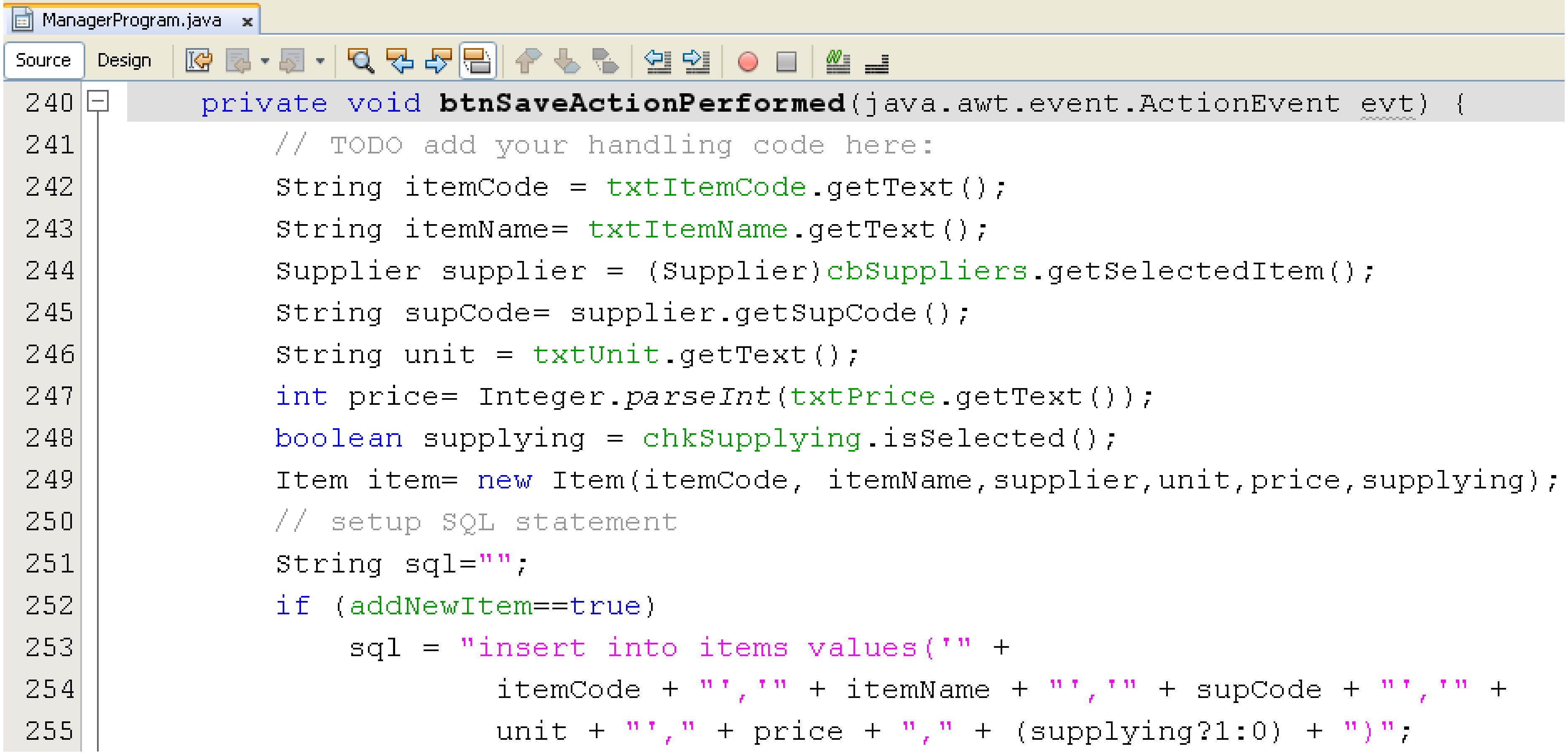
```java
    private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int pos= tblItems.getSelectedRow();
        String itemCode = txtItemCode.getText();
        String sql= "Delete from items where itemcode='" + itemCode + "'";
        JOptionPane.showMessageDialog(this, sql);
        String msg= "The item " + itemCode + " has been deleted from DB!";
        try
        {   int n= dbAccess.executeUpdate(sql);
            if (n>0)
            { JOptionPane.showMessageDialog(this, msg);
              itemModel.getItems().removeElementAt(pos);
              tblItems.updateUI();
            }
        }
        catch (Exception e)
        { JOptionPane.showMessageDialog(this, e);
        }
    }
```

Demo 2…



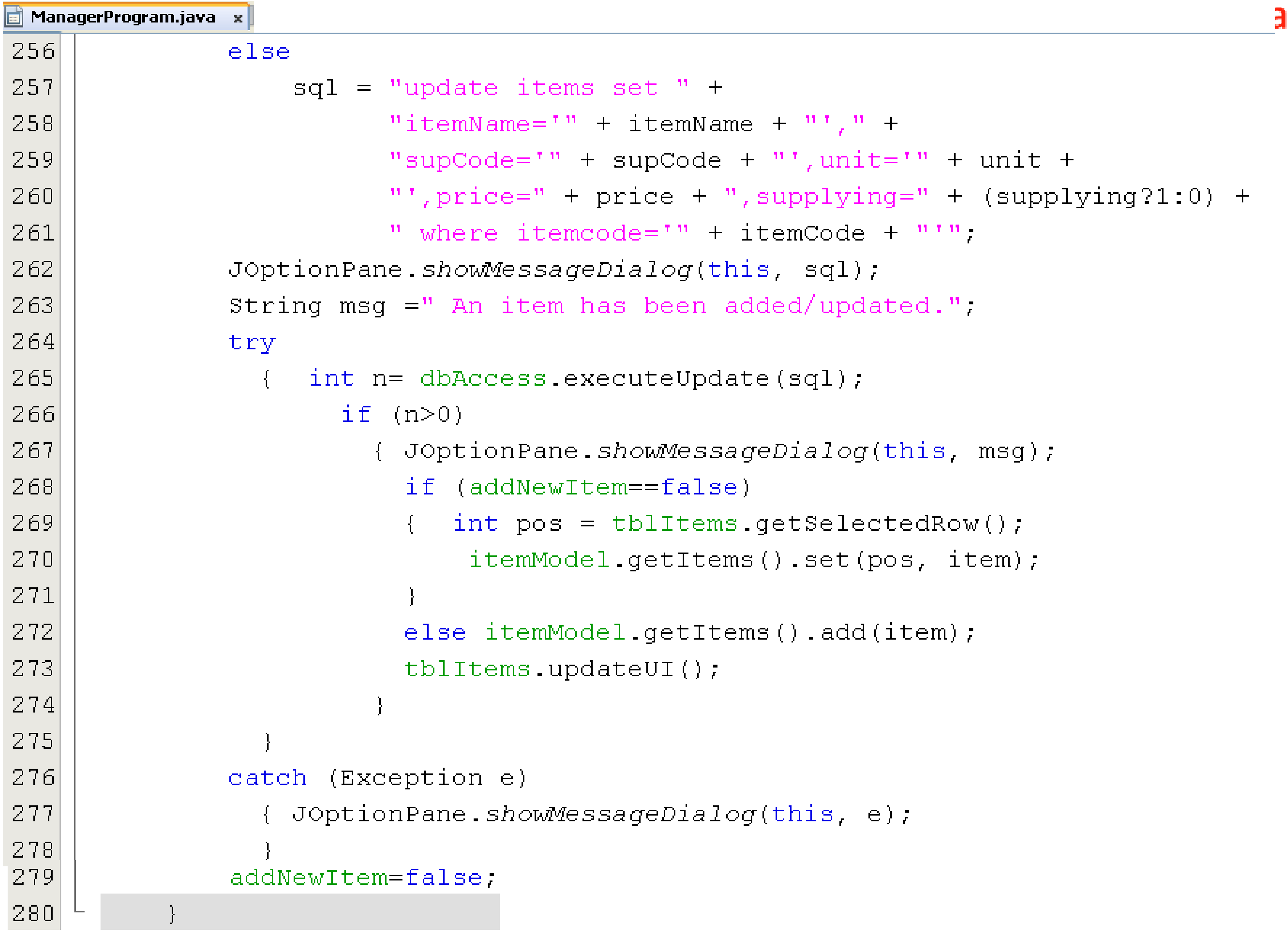
```
    private void btnNewActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        addNewItem=true;
        txtItemCode.setText("");
        txtItemCode.setEditable(true);
        txtItemCode.requestFocus();
        txtItemName.setText("");
        cbSuppliers.setSelectedIndex(0);
        txtUnit.setText("");
        txtPrice.setText("");
        chkSupplying.setSelected(true);
    }
```

Demo 2…



```
    private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String itemCode = txtItemCode.getText();
        String itemName= txtItemName.getText();
        Supplier supplier = (Supplier)cbSuppliers.getSelectedItem();
        String supCode= supplier.getSupCode();
        String unit = txtUnit.getText();
        int price= Integer.parseInt(txtPrice.getText());
        boolean supplying = chkSupplying.isSelected();
        Item item= new Item(itemCode, itemName,supplier,unit,price,supplying);
        // setup SQL statement
        String sql="";
        if (addNewItem==true)
            sql = "insert into items values('" +
                    itemCode + "','" + itemName + "','" + supCode + "','" +
                    unit + "'," + price + "," + (supplying?1:0) + ")";
```

Demo 2…



```
        else
            sql = "update items set " +
                  "itemName='" + itemName + "'," +
                  "supCode='" + supCode + "',unit='" + unit +
                  "',price=" + price + ",supplying=" + (supplying?1:0) +
                  " where itemcode='" + itemCode + "'";
        JOptionPane.showMessageDialog(this, sql);
        String msg =" An item has been added/updated.";
        try
          {   int n= dbAccess.executeUpdate(sql);
               if (n>0)
                 { JOptionPane.showMessageDialog(this, msg);
                   if (addNewItem==false)
                   {   int pos = tblItems.getSelectedRow();
                        itemModel.getItems().set(pos, item);
                   }
                   else itemModel.getItems().add(item);
                   tblItems.updateUI();
                 }
          }
        catch (Exception e)
          { JOptionPane.showMessageDialog(this, e);
          }
        addNewItem=false;
    }
```

Summary

- Introduction to databases
- Relational Database Overview
- JDBC and JDBC Drivers
- Steps to develop a JDBC application.
- Demonstrations

Thank You