Programiranje Internet alikacija

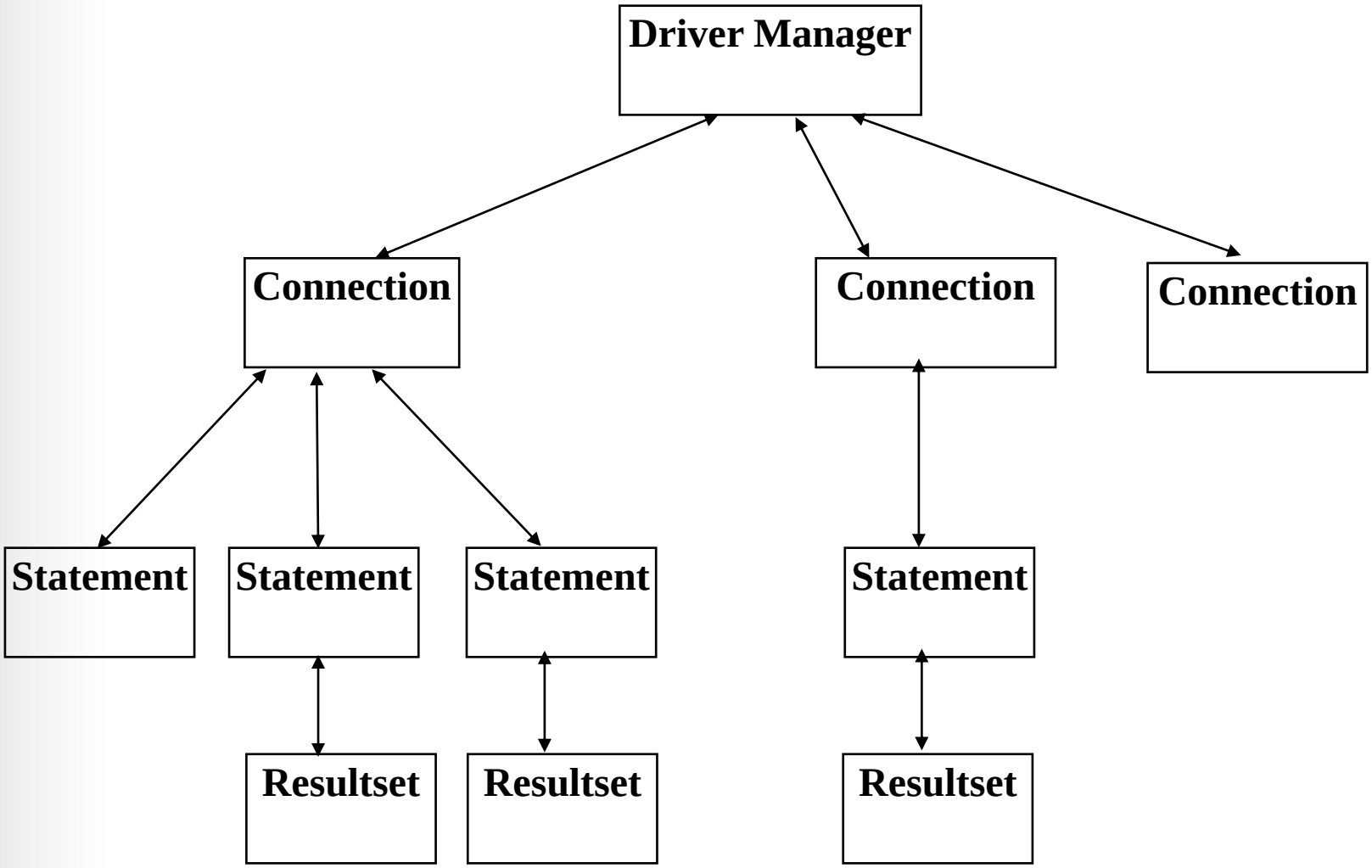
Java i rad sa podacima

Java i baze podataka

- podsistem za komunikaciju sa RDBMS (*Relational database management system*) serverima zasnovanim na SQL-u
- celokupan podsistem je definisan u standardnom paketu

  `java.sql`
- ODBC i JDBC
  - Open Database Connectivity (ODBC) je standardni API za povezivanje sa RDBMS
- JDBC je JAVA API za baze podataka bazirane na SQL jeziku.
- pomoću klasa iz ovog paketa se mogu izvršavati SQL naredbe i manipulisati sa rezultatima dobijenim na osnovu ovih naredbi.
- JDBC API sadrži niz apstraktnih Java interface-a koji dozvoljavaju programeru da ostvari konekciju sa određenom bazom podataka, izvrši SQL naredbe i obradi dobijene rezultate.

Struktura


Driver Manager
Connection
Connection
Connection
Statement
Statement
Statement
Statement
Resultset
Resultset
Resultset

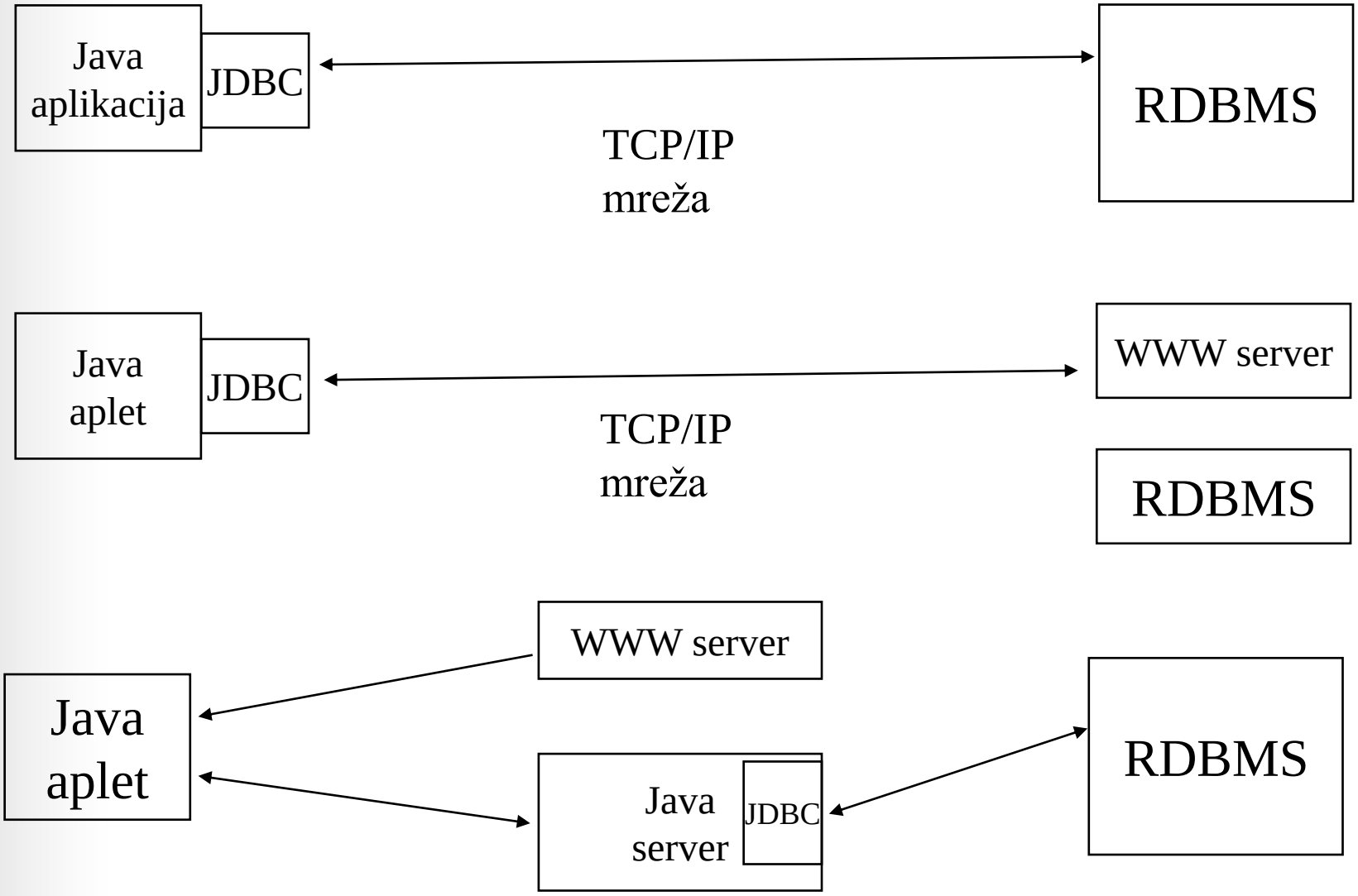
JDBC drajveri

- skup klasa koje implementiraju interfejse iz paketa `java.sql`
- obezbeđuje ih svaki proizvođač RDBMS servera za svoje sisteme
- svi drajveri se na isti način koriste
- instalacija drajvera = uključivanje u CLASSPATH
- isti Java kod se može koristiti za rad sa serverima različitih proizvođača, uz korišćenje odgovarajućeg JDBC drajvera

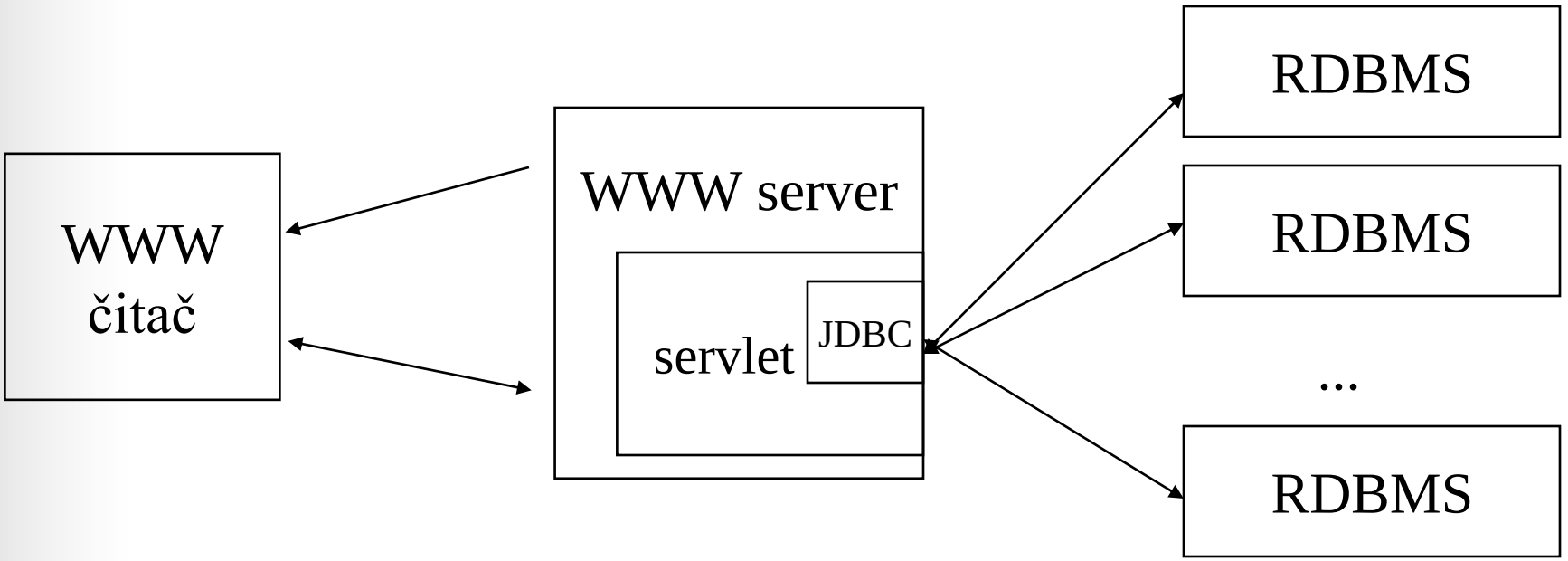
JDBC i klijent/server model

- pristup bazi podataka iz Java aplikacije
- dvoslojni (two-tier) model
- troslojni (three-tier) model
- CGI skriptovi i servleti
- Web servisi

Pristup bazi podataka iz Java aplikacije



Servleti



Osnovne klase

- java.sql.DriverManager sprovodi učitavanje driver-a baze podataka i omogućava podršku za kreiranje nove konekcije
- java.sql.Connection predstavlja konekciju sa određenom bazom podataka
- java.sql.Statement izvršava se u obliku container-a za izvršavanje SQL naredbi u okviru uspostavljene konekcije
- java.sql.ResultSet kontroliše pristup rezultatima dobijenim izvršavanjem određene SQL naredbe
- java.sql.Statement interface ima dva važna podtipa:
  - **java.sql.PreparedStatement** za izvršavanje pre-kompajlirane SQL naredbe
  - **java.sql.CallableStatement** za izvršavanje poziva stored procedura koje postoje u okviru baze podataka.

Inicijalizacija

- JDBC URL ima sledeću strukturu: jdbc:<subprotocol>:<name>
- *subprotocol* je ime određene vrste mehanizma pristupa bazi podataka koji može biti podržan od jednog ili više drajvera. Sadržaj i sintaksa dela *name* zavisi od subprotocola.
- JDBC management nivo mora da zna koji drajver je raspoloživ i koji drajver se koristi.
- primer inicijalizacije drajvera baze podataka:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Class.forName("org.gjt.mm.mysql.Driver”);
```


Elementi JDBC drajvera

- Connection
- Statement
- PreparedStatement
- CallableStatement
- ResultSet
- ResultSetMetaData

Primer 1

STUDENT	
STUDENT_ID	INTEGER
IME	VARCHAR(25)
PREZIME	VARCHAR(25)

Uspostavljanje veze

- Učitavanje JDBC drajvera

```
Class.forName(“com.mysql.jdbc.Driver");
Class.forName(“com.informix.jdbc.IfxDriver");
Class.forName("oracle.jdbc.driver.OracleDriver");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

- Uspostavljanje veze:

```
Connection conn =  DriverManager.getConnection(
   "jdbc:mysql://student.etfbl.net:3306/STUDENT",
   “USERNAME", “PASSWORD");

Connection conn =  DriverManager.getConnection(
   "jdbc:informix-
    sqli://student.etfbl.net:1526/STUDENT:INFORMIXSERVER=ds_server",
   “USERNAME", “PASSWORD");

Connection conn =  DriverManager.getConnection(
   "jdbc:oracle:thin:@student.etfbl.net:1526:STUDENT",
   “USERNAME", “PASSWORD");

Connection conn =  DriverManager.getConnection(
   "jdbc:odbc:student“,  “USERNAME", “PASSWORD");
```

Izuzeci

- **Učitavanje JDBC drajvera**

```
Class.forName("com.mysql.jdbc.Driver"); - ClassNotFoundException
```

- **Uspostavljanje veze**

```
Connection conn =
  DriverManager.getConnection(
    "jdbc:mysql://student.etfbl.net:3306/STUDENT",
    "USERNAME", "PASSWORD"); - SQLException
```

- **Ostale metode** – `SQLException`
- **try-catch** blok
- Završetak komunikacije s bazom podataka - metoda `close()` klase `Connection`

Postavljanje upita

- Statement i ResultSet

```
String query = "SELECT ime, prezime FROM studenti";
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery(query);

while (rset.next()) {
  System.out.println(
    rset.getString(1) + " " + rset.getString(2));
}

rset.close();
stmt.close();
```


DML naredbe

- INSERT, UPDATE, DELETE

```
String query = "INSERT INTO studenti (ime, prezime)
  VALUES ('Nenad', 'Nenadovic') ";

Statement stmt = conn.createStatement();

int n = stmt.executeUpdate(query);

stmt.close();
```


Izvršavanje istih SQL naredbi više puta uzastopno

- PreparedStatement

```
PreparedStatement stmt = conn.prepareStatement(
  "INSERT INTO studenti (student_id, ime, prezime)
   values (?, ?, ?)");

stmt.setInt(1, 1);
stmt.setString(2, new String("Marko"));
stmt.setString(3, new String("Markovic"));
stmt.executeUpdate();

stmt.setInt(1, 2);
stmt.setString(2, new String("Nenad"));
stmt.setString(3, new String("Nenadovic"));
stmt.executeUpdate();

stmt.close();
```

- U slučaju SELECT naredbe poziva se stmt.executeQuery();

Pozivanje uskladištenih procedura i funkcija (*stored procedures*)

- Procedure koje se smeštaju u okviru baze podataka
- Dostupne za pozivanje od strane klijenta
- Proširenja SQL-a (PL/SQL, Transact-SQL,...)

Pozivanje uskladištenih procedura i funkcija

- CallableStatement

```
// pozivamo funkciju uradi
CallableStatement stmt = conn.prepareCall(
  "{? = call uradi (?, ?)}");
```

za funkcije

```
// postavljamo parametre
stmt.setString(2, new String("Sima"));
stmt.setString(3, new String("Simic"));
stmt.registerOutParameter(1, Types.INTEGER);
stmt.executeQuery();

// citamo rezultat funkcije
System.out.println("Status: " + stmt.getInt(1));
```


Primer 2

```
import java.sql.*;
public class JDBCDemoCreateTable {
    public static void main(String[] args) {
// 1. step loading the driver
     try {
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

// 2. step making the connection
       String url = "jdbc:odbc:Primer2" ;
       // the name you use to log in to the DBMS
       String user = "mika" ;
       // your password for the DBMS
       String pass = "mika";
       try {
        Connection con = DriverManager.getConnection(url, user, pass);
        String createTableCoffees = "CREATE TABLE COFFEES " +
                "(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, "
                      +"SALES INTEGER, TOTAL INTEGER)";
//  3. step creating JDBC statement.
        Statement stmt = con.createStatement();
        stmt.executeUpdate(createTableCoffees);
       } catch ( SQLException e ) { System.err.println( e ); }
     }catch ( ClassNotFoundException e) {System.out.println(e); }
   }
}
```


Primer 2

```
import java.sql.*;
public class JDBCDemoUpdateTable {
  public static void main(String[] args) {
    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      Connection con= DriverManager.getConnection( "jdbc:odbc:Primer2",
         "mika", "mika" );
      Statement stmt = con.createStatement();
      String updateTableCoffees ="INSERT INTO COFFEES VALUES”+
                  “('Col', 101, 7, 0, 0)" ;
      stmt.executeUpdate( updateTableCoffees );
      updateTableCoffees="INSERT INTO COFFEES VALUES('French‘,49,8.99, 0,0)”;
      stmt.executeUpdate( updateTableCoffees );
      updateTableCoffees="INSERT INTO COFFEES VALUES('Espresso‘,80,9.99,0,0)”;
      stmt.executeUpdate( updateTableCoffees );
      updateTableCoffees="INSERT INTO COFFEES VALUES('Colombian‘,98,9.1,0,0)”;
      stmt.executeUpdate( updateTableCoffees );
      updateTableCoffees ="INSERT INTO COFFEES VALUES('French‘,49,9.99,0,0)";
      stmt.executeUpdate( updateTableCoffees );
  }catch ( SQLException e ) { System.err.println( e );
  }catch ( ClassNotFoundException e) {System.out.println(e); }
 }
}
```


Primer 2

```
import java.sql.*;
public class JDBCDemoSelectTable {
  public static void main(String[] args) {
    try{
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     Connection con = DriverManager.getConnection("jdbc:odbc:Primer2",
                "mika", "mika" );
     Statement stmt = con.createStatement();
     String queryTableCoffees ="SELECT COF_NAME, SUP_ID, “
                          +”PRICE, SALES, TOTAL FROM COFFEES" ;
     ResultSet rs = stmt.executeQuery( queryTableCoffees );
     while (rs.next()) {
            String coffeName = rs.getString("COF_NAME");
            int supplierId = rs.getInt("SUP_ID");
            float price = rs.getFloat("PRICE");
            int sales = rs.getInt( 4 );
            int total = rs.getInt( 5 );
           System.out.println( coffeName + " " + supplierId + " " +
                        price + "   “ +sales + "    " + total );
     }
   } catch ( SQLException e ) { System.err.println(e);
   } catch ( ClassNotFoundException e) { System.out.println(e ); }
   }
}
```


Primer 2

```
import java.sql.*;
public class JDBCDemoPreparedStatement {
  public static void main(String[] args) {
    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      try {
      Connection con = DriverManager.getConnection(
                "jdbc:odbc:Primer2", "mika", "mika" );
      PreparedStatement updateSalesPrepared = con.prepareStatement(
          "UPDATE COFFEES SET SALES = ? WHERE COF_NAME LIKE ?");
      int [] salesForWeek = {175, 150, 60, 155, 90};
      String [] coffees = {"Colombian", "French_Roast", "Espresso",
                    "Colombian_Decaf", "French_Roast_Decaf"};
      int len = coffees.length;
      for(int i = 0; i < len; i++) {
        // setting up first parameter
        updateSalesPrepared.setInt(1, salesForWeek[i]);
        // setting up second parameter
        updateSalesPrepared.setString(2, coffees[i]);
        updateSalesPrepared.executeUpdate();  // executing update
      }
```


Primer 2

```
    Statement stmt = con.createStatement();
    String queryTableCoffees = "SELECT COF_NAME, SALES"
      + "FROM COFFEES" ;
    ResultSet rs = stmt.executeQuery( queryTableCoffees );
    while (rs.next()) {
      String coffeName = rs.getString("COF_NAME");
      int sales = rs.getInt(2);
      System.out.println( coffeName + "    " + sales  );
    }
  } catch ( SQLException e ) { System.err.println( e );
  } catch ( ClassNotFoundException e) {
    System.out.println("Error : " + e ); }
  }
}
```


Primer 2

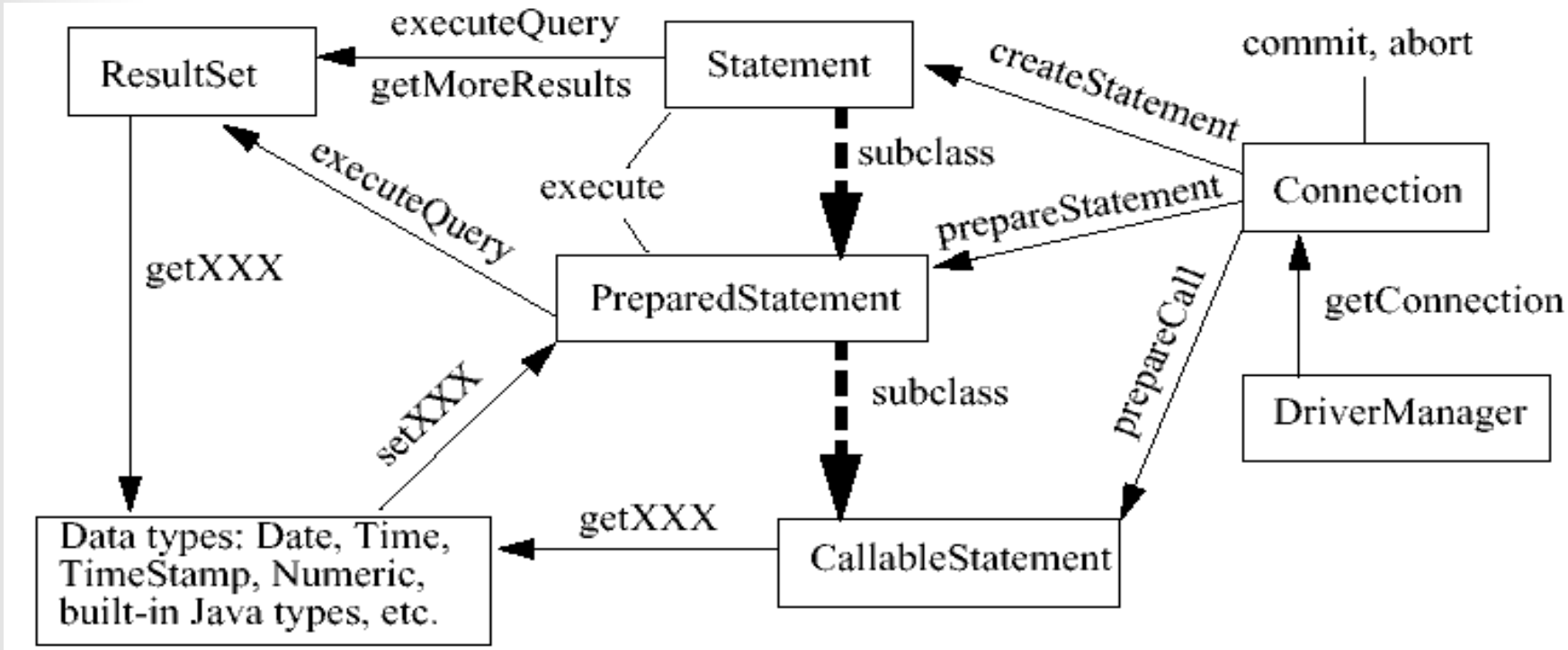
```
import java.sql.*;
public class JDBCDemoCallableStatement {
  public static void main(String[] args) {
    try{
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      String url = "jdbc:odbc:Primer2" ;
      String user = "mika”; //the name you use to log in to the DBMS
      String pass = "mika”; // your password for the DBMS
      try {
        Connection con = DriverManager.getConnection(url,user, pass);
       // stored procedura je grupa SQL naredbi koja formira logičku
       // celinu i izvršava određeni zadatak
       String createProcedure ="create procedure SHOW_SUPPLIERS “
           + "as select SUPPLIERS.SUP_NAME, COFFEES.COF_NAME “
           + "from SUPPLIERS, COFFEES “
           + "where SUPPLIERS.SUP_ID = COFFEES.SUP_ID “
           + "order by SUP_NAME";
```


Primer 2

```
        Statement stmt = con.createStatement();
        stmt.executeUpdate(createProcedure);
         //  Calling a Stored Procedure from JDBC
        CallableStatement cs =
                con.prepareCall("{call SHOW_SUPPLIERS}");
        ResultSet rs = cs.executeQuery();

        while (rs.next()) {
            String supName = rs.getString("SUP_NAME");
            String coffyName = rs.getString("COF_NAME");
            System.out.println( supName + "   " + coffyName );
        }
      }catch ( SQLException e ) { System.err.println( e );
      }catch ( ClassNotFoundException e) {
         System.out.println(e );
      }
   }
}
```


Dijagram izvršavanja



Upravljanje transakcijama

- `conn.commit();`
- `conn.rollback();`
- `conn.setAutoCommit(false);`