Die Methode `executeQuery` führt die SQL-Abfrage (bzw. den *SELECT*-Befehl) aus und gibt das Ergebnis als Objekt vom Typ `ResultSet` zurück.

```
            ResultSet ergebnis =
                sqlBefehl.executeQuery("SELECT * FROM Kunden;");
```

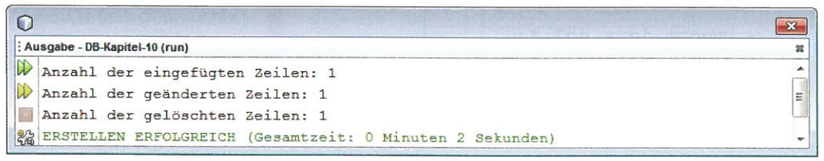
Das Ergebnisobjekt bietet Methoden an, um die Ergebnistabelle abzufragen. Die Methode `next` zeigt an, ob weitere Einträge vorhanden sind und die Methode `getString` liest den nächsten Eintrag aus der gewünschten Spalte (hier *Name*).

```
            while (ergebnis.next() == true) {
                System.out.println("Name: " + ergebnis.
                                   getString("Name"));
            }
            verbindung.close();
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

WICHTIG: Datenbankverbindungen sollten wieder geschlossen werden.

ACHTUNG: Bei der Abfrage von Datenbanken ist es besonders wichtig, dass die Ausnahmebehandlung eingesetzt wird!

Nach dem Starten erscheint dann die folgende Bildschirmausgabe:



Die Datenbank wurde einwandfrei abgefragt und alle Namen der Kunden ausgelesen und angezeigt.

Hinweis:

Der Zugriff auf die Spaltenwerte einer Tabelle mit dem Ergebnisobjekt erfolgt in Abhängigkeit vom jeweiligen Datentyp. Für jeden Datentyp steht eine geeignete Methode zu Verfügung, die entweder den Spaltenindex oder den Spaltennamen übernimmt:

- `getString( int spaltenindex )`
- `getString ( String spaltenname )`
- `getDouble( int spaltenindex )`
- `getDouble ( String spaltenname )`
- `getInt( int spaltenindex )`
- `getInt ( String spaltenname )`
- ... weitere Typen

Beispielsweise könnte die erste Spalte der Kunden-Tabelle mit der Methode `getInt` ausgelesen werden, da es sich um einen ganzzahligen numerischen Typen handelt:

```
while (ergebnis.next() == true){
 System.out.println("ID: " +
        ergebnis.getInt(0));
}
```

9.1.4 Nicht-Select-Befehle absetzen

Das Auslesen einer beliebigen Tabelle kann mithilfe der oben beschriebenen Anweisungen erfolgen. Möchte man hingegen nicht selektieren, sondern einfügen, ändern oder löschen, so kann ein so genannter **executeUpdate**-Befehl abgesetzt werden. Vorher sollte der gewünschte SQL-Befehl in einer Zeichenkette erstellt werden. In dem folgenden Beispiel wird eine neue Zeile in die Kundentabelle eingefügt, eine bestehende Zeile geändert und eine Zeile gelöscht:

```
package db_zugriff_java;
import java.sql.*;

public class DBZugriff {
    public static void main(String[] args) {
      try {
            String datenbank = "jdbc:sqlite:/c:/temp/kunden.sqlite";
            Class.forName("org.sqlite.JDBC");
            Connection verbindung =
                 DriverManager.getConnection(datenbank,"","");
            Statement sqlBefehl = verbindung.createStatement();
```

SQL-Befehl absetzen und die Anzahl der betroffenen Zeilen zurückerhalten.

Der SQL-Befehl, um eine Zeile einzufügen.

```
            int anzahl = sqlBefehl.executeUpdate("INSERT INTO
                                                  Kunden VALUES
                                                  (7,'Koenig');");

            System.out.println("Anzahl der eingefügten Zeilen: "
                               + anzahl);
```

Ein `UPDATE`-Befehl

```
            anzahl = sqlBefehl.executeUpdate("UPDATE Kunden SET
                                              Name = 'Knoblauch'
                                              WHERE Name =
                                              'Knobloch';");

            System.out.println("Anzahl der geänderten Zeilen: "
                               + anzahl);
```

Ein `DELETE`-Befehl

```
            anzahl = sqlBefehl.executeUpdate("DELETE FROM Kunden
                                              WHERE Name =
                                              'Knudsen';");
```