| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BaseRowSet.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PREV CLASS   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/CachedRowSet.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/BaseRowSet.html)    [**NO FRAMES**](http://docs.google.com/BaseRowSet.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#1ksv4uv) | [METHOD](#2jxsxqh) |

## **javax.sql.rowset**

Class BaseRowSet

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.sql.rowset.BaseRowSet**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

public abstract class **BaseRowSet**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

An abstract class providing a RowSet object with its basic functionality. The basic functions include having properties and sending event notifications, which all JavaBeansTM components must implement.

### 1.0 Overview

The BaseRowSet class provides the core functionality for all RowSet implementations, and all standard implementations **may** use this class in combination with one or more RowSet interfaces in order to provide a standard vendor-specific implementation. To clarify, all implementations must implement at least one of the RowSet interfaces (JdbcRowSet, CachedRowSet, JoinRowSet, FilteredRowSet, or WebRowSet). This means that any implementation that extends the BaseRowSet class must also implement one of the RowSet interfaces.

The BaseRowSet class provides the following:

* **Properties**
  + Fields for storing current properties
  + Methods for getting and setting properties
* **Event notification**
* **A complete set of setter methods** for setting the parameters in a RowSet object's command
* **Streams**
  + Fields for storing stream instances
  + Constants for indicating the type of a stream

### 2.0 Setting Properties

All rowsets maintain a set of properties, which will usually be set using a tool. The number and kinds of properties a rowset has will vary, depending on what the RowSet implementation does and how it gets its data. For example, rowsets that get their data from a ResultSet object need to set the properties that are required for making a database connection. If a RowSet object uses the DriverManager facility to make a connection, it needs to set a property for the JDBC URL that identifies the appropriate driver, and it needs to set the properties that give the user name and password. If, on the other hand, the rowset uses a DataSource object to make the connection, which is the preferred method, it does not need to set the property for the JDBC URL. Instead, it needs to set the property for the logical name of the data source along with the properties for the user name and password.

NOTE: In order to use a DataSource object for making a connection, the DataSource object must have been registered with a naming service that uses the Java Naming and Directory InterfaceTM (JNDI) API. This registration is usually done by a person acting in the capacity of a system administrator.

### 3.0 Setting the Command and Its Parameters

When a rowset gets its data from a relational database, it executes a command (a query) that produces a ResultSet object. This query is the command that is set for the RowSet object's command property. The rowset populates itself with data by reading the data from the ResultSet object into itself. If the query contains placeholders for values to be set, the BaseRowSet setter methods are used to set these values. All setter methods allow these values to be set to null if required.

The following code fragment illustrates how the CachedRowSetTM object crs might have its command property set. Note that if a tool is used to set properties, this is the code that the tool would use.

crs.setCommand("SELECT FIRST\_NAME, LAST\_NAME, ADDRESS FROM CUSTOMERS" +  
 "WHERE CREDIT\_LIMIT > ? AND REGION = ?");

In this example, the values for CREDIT\_LIMIT and REGION are placeholder parameters, which are indicated with a question mark (?). The first question mark is placeholder parameter number 1, the second question mark is placeholder parameter number 2, and so on. Any placeholder parameters must be set with values before the query can be executed. To set these placeholder parameters, the BaseRowSet class provides a set of setter methods, similar to those provided by the PreparedStatement interface, for setting values of each data type. A RowSet object stores the parameter values internally, and its execute method uses them internally to set values for the placeholder parameters before it sends the command to the DBMS to be executed.

The following code fragment demonstrates setting the two parameters in the query from the previous example.

crs.setInt(1, 5000);  
 crs.setString(2, "West");

If the execute method is called at this point, the query sent to the DBMS will be:

"SELECT FIRST\_NAME, LAST\_NAME, ADDRESS FROM CUSTOMERS" +  
 "WHERE CREDIT\_LIMIT > 5000 AND REGION = 'West'"

NOTE: Setting Array, Clob, Blob and Ref objects as a command parameter, stores these values as SerialArray, SerialClob, SerialBlob and SerialRef objects respectively.

### 4.0 Handling of Parameters Behind the Scenes

NOTE: The BaseRowSet class provides two kinds of setter methods, those that set properties and those that set placeholder parameters. The setter methods discussed in this section are those that set placeholder parameters.

The placeholder parameters set with the BaseRowSet setter methods are stored as objects in an internal Hashtable object. Primitives are stored as their Object type. For example, byte is stored as Byte object, and int is stored as an Integer object. When the method execute is called, the values in the Hashtable object are substituted for the appropriate placeholder parameters in the command. A call to the method getParams returns the values stored in the Hashtable object as an array of Object instances. An element in this array may be a simple Object instance or an array (which is a type of Object). The particular setter method used determines whether an element in this array is an Object or an array.

The majority of methods for setting placeholder parameters take two parameters, with the first parameter indicating which placeholder parameter is to be set, and the second parameter giving the value to be set. Methods such as getInt, getString, getBoolean, and getLong fall into this category. After these methods have been called, a call to the method getParams will return an array with the values that have been set. Each element in the array is an Object instance representing the values that have been set. The order of these values in the array is determined by the int (the first parameter) passed to the setter method. The values in the array are the values (the second parameter) passed to the setter method. In other words, the first element in the array is the value to be set for the first placeholder parameter in the RowSet object's command. The second element is the value to be set for the second placeholder parameter, and so on.

Several setter methods send the driver and DBMS information beyond the value to be set. When the method getParams is called after one of these setter methods has been used, the elements in the array will themselves be arrays to accommodate the additional information. In this category, the method setNull is a special case because one version takes only two parameters (setNull(int parameterIndex, int SqlType)). Nevertheless, it requires an array to contain the information that will be passed to the driver and DBMS. The first element in this array is the value to be set, which is null, and the second element is the int supplied for *sqlType*, which indicates the type of SQL value that is being set to null. This information is needed by some DBMSs and is therefore required in order to ensure that applications are portable. The other version is intended to be used when the value to be set to null is a user-defined type. It takes three parameters (setNull(int parameterIndex, int sqlType, String typeName)) and also requires an array to contain the information to be passed to the driver and DBMS. The first two elements in this array are the same as for the first version of setNull. The third element, *typeName*, gives the SQL name of the user-defined type. As is true with the other setter methods, the number of the placeholder parameter to be set is indicated by an element's position in the array returned by getParams. So, for example, if the parameter supplied to setNull is 2, the second element in the array returned by getParams will be an array of two or three elements.

Some methods, such as setObject and setDate have versions that take more than two parameters, with the extra parameters giving information to the driver or the DBMS. For example, the methods setDate, setTime, and setTimestamp can take a Calendar object as their third parameter. If the DBMS does not store time zone information, the drivern uses the Calendar object to construct the Date, Time, or Timestamp object being set. As is true with other methods that provide additional information, the element in the array returned by getParams is an array instead of a simple Object instance.

The methods setAsciiStream, setBinaryStream, setCharacterStream, and setUnicodeStream (which is deprecated, so applications should use getCharacterStream instead) take three parameters, so for them, the element in the array returned by getParams is also an array. What is different about these setter methods is that in addition to the information provided by parameters, the array contains one of the BaseRowSet constants indicating the type of stream being set.

NOTE: The method getParams is called internally by RowSet implementations extending this class; it is not normally called by an application programmer directly.

### 5.0 Event Notification

The BaseRowSet class provides the event notification mechanism for rowsets. It contains the field listeners, methods for adding and removing listeners, and methods for notifying listeners of changes.

A listener is an object that has implemented the RowSetListener interface. If it has been added to a RowSet object's list of listeners, it will be notified when an event occurs on that RowSet object. Each listener's implementation of the RowSetListener methods defines what that object will do when it is notified that an event has occurred.

There are three possible events for a RowSet object:

1. the cursor moves
2. an individual row is changed (updated, deleted, or inserted)
3. the contents of the entire RowSet object are changed

The BaseRowSet method used for the notification indicates the type of event that has occurred. For example, the method notifyRowChanged indicates that a row has been updated, deleted, or inserted. Each of the notification methods creates a RowSetEvent object, which is supplied to the listener in order to identify the RowSet object on which the event occurred. What the listener does with this information, which may be nothing, depends on how it was implemented.

### 6.0 Default Behavior

A default BaseRowSet object is initialized with many starting values. The following is true of a default RowSet instance that extends the BaseRowSet class:

* Has a scrollable cursor and does not show changes made by others.
* Is updatable.
* Does not show rows that have been deleted.
* Has no time limit for how long a driver may take to execute the RowSet object's command.
* Has no limit for the number of rows it may contain.
* Has no limit for the number of bytes a column may contain. NOTE: This limit applies only to columns that hold values of the following types: BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, and LONGVARCHAR.
* Will not see uncommitted data (make "dirty" reads).
* Has escape processing turned on.
* Has its connection's type map set to null.
* Has an empty Vector object for storing the values set for the placeholder parameters in the RowSet object's command.

If other values are desired, an application must set the property values explicitly. For example, the following line of code sets the maximum number of rows for the CachedRowSet object *crs* to 500.

crs.setMaxRows(500);

Methods implemented in extensions of this BaseRowSet class **must** throw an SQLException object for any violation of the defined assertions. Also, if the extending class overrides and reimplements any BaseRowSet method and encounters connectivity or underlying data source issues, that method **may** in addition throw an SQLException object for that reason.

**See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#javax.sql.rowset.BaseRowSet)

| **Field Summary** | |
| --- | --- |
| static int | [**ASCII\_STREAM\_PARAM**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#ASCII_STREAM_PARAM)            A constant indicating to a RowSetReaderImpl object that a given parameter is an ASCII stream. |
| protected  [InputStream](http://docs.google.com/java/io/InputStream.html) | [**asciiStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#asciiStream)            The InputStream object that will be returned by the method getAsciiStream, which is specified in the ResultSet interface. |
| static int | [**BINARY\_STREAM\_PARAM**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#BINARY_STREAM_PARAM)            A constant indicating to a RowSetReaderImpl object that a given parameter is a binary stream. |
| protected  [InputStream](http://docs.google.com/java/io/InputStream.html) | [**binaryStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#binaryStream)            The InputStream object that will be returned by the method getBinaryStream, which is specified in the ResultSet interface. |
| protected  [Reader](http://docs.google.com/java/io/Reader.html) | [**charStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#charStream)            The Reader object that will be returned by the method getCharacterStream, which is specified in the ResultSet interface. |
| static int | [**UNICODE\_STREAM\_PARAM**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#UNICODE_STREAM_PARAM)            A constant indicating to a RowSetReaderImpl object that a given parameter is a Unicode stream. |
| protected  [InputStream](http://docs.google.com/java/io/InputStream.html) | [**unicodeStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#unicodeStream)            The InputStream object that will be returned by the method getUnicodeStream, which is specified in the ResultSet interface. |

| **Constructor Summary** | |
| --- | --- |
| [**BaseRowSet**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#BaseRowSet())()            Constructs a new BaseRowSet object initialized with a default Vector object for its listeners field. |

| **Method Summary** | |
| --- | --- |
| void | [**addRowSetListener**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#addRowSetListener(javax.sql.RowSetListener))([RowSetListener](http://docs.google.com/javax/sql/RowSetListener.html) listener)            The listener will be notified whenever an event occurs on this RowSet object. |
| void | [**clearParameters**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#clearParameters())()            Clears all of the current parameter values in this RowSet object's internal representation of the parameters to be set in this RowSet object's command when it is executed. |
| [String](http://docs.google.com/java/lang/String.html) | [**getCommand**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getCommand())()            Retrieves the SQL query that is the command for this RowSet object. |
| int | [**getConcurrency**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getConcurrency())()            Returns the concurrency for this RowSet object. |
| [String](http://docs.google.com/java/lang/String.html) | [**getDataSourceName**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getDataSourceName())()            Returns the logical name that when supplied to a naming service that uses the Java Naming and Directory Interface (JNDI) API, will retrieve a javax.sql.DataSource object. |
| boolean | [**getEscapeProcessing**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getEscapeProcessing())()            Ascertains whether escape processing is enabled for this RowSet object. |
| int | [**getFetchDirection**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getFetchDirection())()            Retrieves this RowSet object's current setting for the fetch direction. |
| int | [**getFetchSize**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getFetchSize())()            Returns the fetch size for this RowSet object. |
| int | [**getMaxFieldSize**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getMaxFieldSize())()            Retrieves the maximum number of bytes that can be used for a column value in this RowSet object. |
| int | [**getMaxRows**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getMaxRows())()            Retrieves the maximum number of rows that this RowSet object may contain. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**getParams**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())()            Retrieves an array containing the parameter values (both Objects and primitives) that have been set for this RowSet object's command and throws an SQLException object if all parameters have not been set. |
| [String](http://docs.google.com/java/lang/String.html) | [**getPassword**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getPassword())()            Returns the password used to create a database connection for this RowSet object. |
| int | [**getQueryTimeout**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getQueryTimeout())()            Retrieves the maximum number of seconds the driver will wait for a query to execute. |
| boolean | [**getShowDeleted**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getShowDeleted())()            Retrieves a boolean indicating whether rows marked for deletion appear in the set of current rows. |
| int | [**getTransactionIsolation**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getTransactionIsolation())()            Returns the transaction isolation property for this RowSet object's connection. |
| int | [**getType**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getType())()            Returns the type of this RowSet object. |
| [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> | [**getTypeMap**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getTypeMap())()            Retrieves the type map associated with the Connection object for this RowSet object. |
| [String](http://docs.google.com/java/lang/String.html) | [**getUrl**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getUrl())()            Retrieves the JDBC URL that this RowSet object's javax.sql.Reader object uses to make a connection with a relational database using a JDBC technology-enabled driver. |
| [String](http://docs.google.com/java/lang/String.html) | [**getUsername**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getUsername())()            Returns the user name used to create a database connection. |
| protected  void | [**initParams**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#initParams())()            Performs the necessary internal configurations and initializations to allow any JDBC RowSet implementation to start using the standard facilities provided by a BaseRowSet instance. |
| boolean | [**isReadOnly**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#isReadOnly())()            Returns a boolean indicating whether this RowSet object is read-only. |
| protected  void | [**notifyCursorMoved**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#notifyCursorMoved())()            Notifies all of the listeners registered with this RowSet object that its cursor has moved. |
| protected  void | [**notifyRowChanged**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#notifyRowChanged())()            Notifies all of the listeners registered with this RowSet object that one of its rows has changed. |
| protected  void | [**notifyRowSetChanged**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#notifyRowSetChanged())()            Notifies all of the listeners registered with this RowSet object that its entire contents have changed. |
| void | [**removeRowSetListener**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#removeRowSetListener(javax.sql.RowSetListener))([RowSetListener](http://docs.google.com/javax/sql/RowSetListener.html) listener)            Removes the designated object from this RowSet object's list of listeners. |
| void | [**setArray**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setArray(int,%20java.sql.Array))(int parameterIndex, [Array](http://docs.google.com/java/sql/Array.html) array)            Sets the designated parameter to an Array object in the Java programming language. |
| void | [**setAsciiStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setAsciiStream(int,%20java.io.InputStream))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x)            Sets the designated parameter in this RowSet object's command to the given input stream. |
| void | [**setAsciiStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setAsciiStream(int,%20java.io.InputStream,%20int))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x, int length)            Sets the designated parameter to the given java.io.InputStream object, which will have the specified number of bytes. |
| void | [**setAsciiStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setAsciiStream(java.lang.String,%20java.io.InputStream))([String](http://docs.google.com/java/lang/String.html) parameterName, [InputStream](http://docs.google.com/java/io/InputStream.html) x)            Sets the designated parameter to the given input stream. |
| void | [**setAsciiStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setAsciiStream(java.lang.String,%20java.io.InputStream,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, [InputStream](http://docs.google.com/java/io/InputStream.html) x, int length)            Sets the designated parameter to the given input stream, which will have the specified number of bytes. |
| void | [**setBigDecimal**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBigDecimal(int,%20java.math.BigDecimal))(int parameterIndex, [BigDecimal](http://docs.google.com/java/math/BigDecimal.html) x)            Sets the designated parameter to the given java.lang.BigDecimal value. |
| void | [**setBigDecimal**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBigDecimal(java.lang.String,%20java.math.BigDecimal))([String](http://docs.google.com/java/lang/String.html) parameterName, [BigDecimal](http://docs.google.com/java/math/BigDecimal.html) x)            Sets the designated parameter to the given java.math.BigDecimal value. |
| void | [**setBinaryStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBinaryStream(int,%20java.io.InputStream))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x)            Sets the designated parameter in this RowSet object's command to the given input stream. |
| void | [**setBinaryStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBinaryStream(int,%20java.io.InputStream,%20int))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x, int length)            Sets the designated parameter to the given java.io.InputStream object, which will have the specified number of bytes. |
| void | [**setBinaryStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBinaryStream(java.lang.String,%20java.io.InputStream))([String](http://docs.google.com/java/lang/String.html) parameterName, [InputStream](http://docs.google.com/java/io/InputStream.html) x)            Sets the designated parameter to the given input stream. |
| void | [**setBinaryStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBinaryStream(java.lang.String,%20java.io.InputStream,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, [InputStream](http://docs.google.com/java/io/InputStream.html) x, int length)            Sets the designated parameter to the given input stream, which will have the specified number of bytes. |
| void | [**setBlob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBlob(int,%20java.sql.Blob))(int parameterIndex, [Blob](http://docs.google.com/java/sql/Blob.html) x)            Sets the designated parameter to the given Blob object in the Java programming language. |
| void | [**setBlob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBlob(int,%20java.io.InputStream))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream)            Sets the designated parameter to a InputStream object. |
| void | [**setBlob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBlob(int,%20java.io.InputStream,%20long))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream, long length)            Sets the designated parameter to a InputStream object. |
| void | [**setBlob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBlob(java.lang.String,%20java.sql.Blob))([String](http://docs.google.com/java/lang/String.html) parameterName, [Blob](http://docs.google.com/java/sql/Blob.html) x)            Sets the designated parameter to the given java.sql.Blob object. |
| void | [**setBlob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBlob(java.lang.String,%20java.io.InputStream))([String](http://docs.google.com/java/lang/String.html) parameterName, [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream)            Sets the designated parameter to a InputStream object. |
| void | [**setBlob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBlob(java.lang.String,%20java.io.InputStream,%20long))([String](http://docs.google.com/java/lang/String.html) parameterName, [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream, long length)            Sets the designated parameter to a InputStream object. |
| void | [**setBoolean**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBoolean(int,%20boolean))(int parameterIndex, boolean x)            Sets the designated parameter to the given boolean in the Java programming language. |
| void | [**setBoolean**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBoolean(java.lang.String,%20boolean))([String](http://docs.google.com/java/lang/String.html) parameterName, boolean x)            Sets the designated parameter to the given Java boolean value. |
| void | [**setByte**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setByte(int,%20byte))(int parameterIndex, byte x)            Sets the designated parameter to the given byte in the Java programming language. |
| void | [**setByte**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setByte(java.lang.String,%20byte))([String](http://docs.google.com/java/lang/String.html) parameterName, byte x)            Sets the designated parameter to the given Java byte value. |
| void | [**setBytes**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBytes(int,%20byte%5B%5D))(int parameterIndex, byte[] x)            Sets the designated parameter to the given array of bytes. |
| void | [**setBytes**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setBytes(java.lang.String,%20byte%5B%5D))([String](http://docs.google.com/java/lang/String.html) parameterName, byte[] x)            Sets the designated parameter to the given Java array of bytes. |
| void | [**setCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setCharacterStream(int,%20java.io.Reader))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader)            Sets the designated parameter in this RowSet object's command to the given Reader object. |
| void | [**setCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setCharacterStream(int,%20java.io.Reader,%20int))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader, int length)            Sets the designated parameter to the given java.io.Reader object, which will have the specified number of characters. |
| void | [**setCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setCharacterStream(java.lang.String,%20java.io.Reader))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) reader)            Sets the designated parameter to the given Reader object. |
| void | [**setCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setCharacterStream(java.lang.String,%20java.io.Reader,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) reader, int length)            Sets the designated parameter to the given Reader object, which is the given number of characters long. |
| void | [**setClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setClob(int,%20java.sql.Clob))(int parameterIndex, [Clob](http://docs.google.com/java/sql/Clob.html) x)            Sets the designated parameter to the given Clob object in the Java programming language. |
| void | [**setClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setClob(int,%20java.io.Reader))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader)            Sets the designated parameter to a Reader object. |
| void | [**setClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setClob(int,%20java.io.Reader,%20long))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader, long length)            Sets the designated parameter to a Reader object. |
| void | [**setClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setClob(java.lang.String,%20java.sql.Clob))([String](http://docs.google.com/java/lang/String.html) parameterName, [Clob](http://docs.google.com/java/sql/Clob.html) x)            Sets the designated parameter to the given java.sql.Clob object. |
| void | [**setClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setClob(java.lang.String,%20java.io.Reader))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) reader)            Sets the designated parameter to a Reader object. |
| void | [**setClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setClob(java.lang.String,%20java.io.Reader,%20long))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) reader, long length)            Sets the designated parameter to a Reader object. |
| void | [**setCommand**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setCommand(java.lang.String))([String](http://docs.google.com/java/lang/String.html) cmd)            Sets this RowSet object's command property to the given String object and clears the parameters, if any, that were set for the previous command. |
| void | [**setConcurrency**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setConcurrency(int))(int concurrency)            Sets the concurrency for this RowSet object to the specified concurrency. |
| void | [**setDataSourceName**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDataSourceName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Sets the DataSource name property for this RowSet object to the given logical name and sets this RowSet object's Url property to null. |
| void | [**setDate**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDate(int,%20java.sql.Date))(int parameterIndex, [Date](http://docs.google.com/java/sql/Date.html) x)            Sets the designated parameter to the given java.sql.Date value. |
| void | [**setDate**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDate(int,%20java.sql.Date,%20java.util.Calendar))(int parameterIndex, [Date](http://docs.google.com/java/sql/Date.html) x, [Calendar](http://docs.google.com/java/util/Calendar.html) cal)            Sets the designated parameter to the given java.sql.Date object. |
| void | [**setDate**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDate(java.lang.String,%20java.sql.Date))([String](http://docs.google.com/java/lang/String.html) parameterName, [Date](http://docs.google.com/java/sql/Date.html) x)            Sets the designated parameter to the given java.sql.Date value using the default time zone of the virtual machine that is running the application. |
| void | [**setDate**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDate(java.lang.String,%20java.sql.Date,%20java.util.Calendar))([String](http://docs.google.com/java/lang/String.html) parameterName, [Date](http://docs.google.com/java/sql/Date.html) x, [Calendar](http://docs.google.com/java/util/Calendar.html) cal)            Sets the designated parameter to the given java.sql.Date value, using the given Calendar object. |
| void | [**setDouble**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDouble(int,%20double))(int parameterIndex, double x)            Sets the designated parameter to the given double in the Java programming language. |
| void | [**setDouble**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDouble(java.lang.String,%20double))([String](http://docs.google.com/java/lang/String.html) parameterName, double x)            Sets the designated parameter to the given Java double value. |
| void | [**setEscapeProcessing**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setEscapeProcessing(boolean))(boolean enable)            Sets to the given boolean whether or not the driver will scan for escape syntax and do escape substitution before sending SQL statements to the database. |
| void | [**setFetchDirection**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setFetchDirection(int))(int direction)            Gives the driver a performance hint as to the direction in which the rows in this RowSet object will be processed. |
| void | [**setFetchSize**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setFetchSize(int))(int rows)            Sets the fetch size for this RowSet object to the given number of rows. |
| void | [**setFloat**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setFloat(int,%20float))(int parameterIndex, float x)            Sets the designated parameter to the given float in the Java programming language. |
| void | [**setFloat**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setFloat(java.lang.String,%20float))([String](http://docs.google.com/java/lang/String.html) parameterName, float x)            Sets the designated parameter to the given Java float value. |
| void | [**setInt**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setInt(int,%20int))(int parameterIndex, int x)            Sets the designated parameter to an int in the Java programming language. |
| void | [**setInt**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setInt(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, int x)            Sets the designated parameter to the given Java int value. |
| void | [**setLong**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setLong(int,%20long))(int parameterIndex, long x)            Sets the designated parameter to the given long in the Java programming language. |
| void | [**setLong**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setLong(java.lang.String,%20long))([String](http://docs.google.com/java/lang/String.html) parameterName, long x)            Sets the designated parameter to the given Java long value. |
| void | [**setMaxFieldSize**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setMaxFieldSize(int))(int max)            Sets the maximum number of bytes that can be used for a column value in this RowSet object to the given number. |
| void | [**setMaxRows**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setMaxRows(int))(int max)            Sets the maximum number of rows that this RowSet object may contain to the given number. |
| void | [**setNCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNCharacterStream(int,%20java.io.Reader))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) value)            Sets the designated parameter in this RowSet object's command to a Reader object. |
| void | [**setNCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNCharacterStream(int,%20java.io.Reader,%20long))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) value, long length)            Sets the designated parameter to a Reader object. |
| void | [**setNCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNCharacterStream(java.lang.String,%20java.io.Reader))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) value)            Sets the designated parameter to a Reader object. |
| void | [**setNCharacterStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNCharacterStream(java.lang.String,%20java.io.Reader,%20long))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) value, long length)            Sets the designated parameter to a Reader object. |
| void | [**setNClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNClob(int,%20java.sql.NClob))(int parameterIndex, [NClob](http://docs.google.com/java/sql/NClob.html) value)            Sets the designated parameter to a java.sql.NClob object. |
| void | [**setNClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNClob(int,%20java.io.Reader))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader)            Sets the designated parameter to a Reader object. |
| void | [**setNClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNClob(int,%20java.io.Reader,%20long))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader, long length)            Sets the designated parameter to a Reader object. |
| void | [**setNClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNClob(java.lang.String,%20java.sql.NClob))([String](http://docs.google.com/java/lang/String.html) parameterName, [NClob](http://docs.google.com/java/sql/NClob.html) value)            Sets the designated parameter to a java.sql.NClob object. |
| void | [**setNClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNClob(java.lang.String,%20java.io.Reader))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) reader)            Sets the designated parameter to a Reader object. |
| void | [**setNClob**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNClob(java.lang.String,%20java.io.Reader,%20long))([String](http://docs.google.com/java/lang/String.html) parameterName, [Reader](http://docs.google.com/java/io/Reader.html) reader, long length)            Sets the designated parameter to a Reader object. |
| void | [**setNString**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNString(int,%20java.lang.String))(int parameterIndex, [String](http://docs.google.com/java/lang/String.html) value)            Sets the designated paramter to the given String object. |
| void | [**setNString**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNString(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) parameterName, [String](http://docs.google.com/java/lang/String.html) value)            Sets the designated paramter to the given String object. |
| void | [**setNull**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNull(int,%20int))(int parameterIndex, int sqlType)            Sets the designated parameter to SQL NULL. |
| void | [**setNull**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNull(int,%20int,%20java.lang.String))(int parameterIndex, int sqlType, [String](http://docs.google.com/java/lang/String.html) typeName)            Sets the designated parameter to SQL NULL. |
| void | [**setNull**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNull(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, int sqlType)            Sets the designated parameter to SQL NULL. |
| void | [**setNull**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setNull(java.lang.String,%20int,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) parameterName, int sqlType, [String](http://docs.google.com/java/lang/String.html) typeName)            Sets the designated parameter to SQL NULL. |
| void | [**setObject**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setObject(int,%20java.lang.Object))(int parameterIndex, [Object](http://docs.google.com/java/lang/Object.html) x)            Sets the designated parameter to an Object in the Java programming language. |
| void | [**setObject**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setObject(int,%20java.lang.Object,%20int))(int parameterIndex, [Object](http://docs.google.com/java/lang/Object.html) x, int targetSqlType)            Sets the value of the designated parameter with the given Object value. |
| void | [**setObject**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setObject(int,%20java.lang.Object,%20int,%20int))(int parameterIndex, [Object](http://docs.google.com/java/lang/Object.html) x, int targetSqlType, int scale)            Sets the designated parameter to an Object in the Java programming language. |
| void | [**setObject**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setObject(java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) parameterName, [Object](http://docs.google.com/java/lang/Object.html) x)            Sets the value of the designated parameter with the given object. |
| void | [**setObject**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setObject(java.lang.String,%20java.lang.Object,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, [Object](http://docs.google.com/java/lang/Object.html) x, int targetSqlType)            Sets the value of the designated parameter with the given object. |
| void | [**setObject**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setObject(java.lang.String,%20java.lang.Object,%20int,%20int))([String](http://docs.google.com/java/lang/String.html) parameterName, [Object](http://docs.google.com/java/lang/Object.html) x, int targetSqlType, int scale)            Sets the value of the designated parameter with the given object. |
| void | [**setPassword**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setPassword(java.lang.String))([String](http://docs.google.com/java/lang/String.html) pass)            Sets the password used to create a database connection for this RowSet object to the given String object. |
| void | [**setQueryTimeout**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setQueryTimeout(int))(int seconds)            Sets to the given number the maximum number of seconds the driver will wait for a query to execute. |
| void | [**setReadOnly**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setReadOnly(boolean))(boolean value)            Sets this RowSet object's readOnly property to the given boolean. |
| void | [**setRef**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setRef(int,%20java.sql.Ref))(int parameterIndex, [Ref](http://docs.google.com/java/sql/Ref.html) ref)            Sets the designated parameter to the given Ref object in the Java programming language. |
| void | [**setRowId**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setRowId(int,%20java.sql.RowId))(int parameterIndex, [RowId](http://docs.google.com/java/sql/RowId.html) x)            Sets the designated parameter to the given java.sql.RowId object. |
| void | [**setRowId**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setRowId(java.lang.String,%20java.sql.RowId))([String](http://docs.google.com/java/lang/String.html) parameterName, [RowId](http://docs.google.com/java/sql/RowId.html) x)            Sets the designated parameter to the given java.sql.RowId object. |
| void | [**setShort**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setShort(int,%20short))(int parameterIndex, short x)            Sets the designated parameter to the given short in the Java programming language. |
| void | [**setShort**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setShort(java.lang.String,%20short))([String](http://docs.google.com/java/lang/String.html) parameterName, short x)            Sets the designated parameter to the given Java short value. |
| void | [**setShowDeleted**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setShowDeleted(boolean))(boolean value)            Sets the property showDeleted to the given boolean value, which determines whether rows marked for deletion appear in the set of current rows. |
| void | [**setSQLXML**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setSQLXML(int,%20java.sql.SQLXML))(int parameterIndex, [SQLXML](http://docs.google.com/java/sql/SQLXML.html) xmlObject)            Sets the designated parameter to the given java.sql.SQLXML object. |
| void | [**setSQLXML**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setSQLXML(java.lang.String,%20java.sql.SQLXML))([String](http://docs.google.com/java/lang/String.html) parameterName, [SQLXML](http://docs.google.com/java/sql/SQLXML.html) xmlObject)            Sets the designated parameter to the given java.sql.SQLXML object. |
| void | [**setString**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setString(int,%20java.lang.String))(int parameterIndex, [String](http://docs.google.com/java/lang/String.html) x)            Sets the designated parameter to the given String value. |
| void | [**setString**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setString(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) parameterName, [String](http://docs.google.com/java/lang/String.html) x)            Sets the designated parameter to the given Java String value. |
| void | [**setTime**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTime(int,%20java.sql.Time))(int parameterIndex, [Time](http://docs.google.com/java/sql/Time.html) x)            Sets the designated parameter to the given java.sql.Time value. |
| void | [**setTime**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTime(int,%20java.sql.Time,%20java.util.Calendar))(int parameterIndex, [Time](http://docs.google.com/java/sql/Time.html) x, [Calendar](http://docs.google.com/java/util/Calendar.html) cal)            Sets the designated parameter to the given java.sql.Time object. |
| void | [**setTime**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTime(java.lang.String,%20java.sql.Time))([String](http://docs.google.com/java/lang/String.html) parameterName, [Time](http://docs.google.com/java/sql/Time.html) x)            Sets the designated parameter to the given java.sql.Time value. |
| void | [**setTime**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTime(java.lang.String,%20java.sql.Time,%20java.util.Calendar))([String](http://docs.google.com/java/lang/String.html) parameterName, [Time](http://docs.google.com/java/sql/Time.html) x, [Calendar](http://docs.google.com/java/util/Calendar.html) cal)            Sets the designated parameter to the given java.sql.Time value, using the given Calendar object. |
| void | [**setTimestamp**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTimestamp(int,%20java.sql.Timestamp))(int parameterIndex, [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x)            Sets the designated parameter to the given java.sql.Timestamp value. |
| void | [**setTimestamp**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTimestamp(int,%20java.sql.Timestamp,%20java.util.Calendar))(int parameterIndex, [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x, [Calendar](http://docs.google.com/java/util/Calendar.html) cal)            Sets the designated parameter to the given java.sql.Timestamp object. |
| void | [**setTimestamp**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTimestamp(java.lang.String,%20java.sql.Timestamp))([String](http://docs.google.com/java/lang/String.html) parameterName, [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x)            Sets the designated parameter to the given java.sql.Timestamp value. |
| void | [**setTimestamp**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTimestamp(java.lang.String,%20java.sql.Timestamp,%20java.util.Calendar))([String](http://docs.google.com/java/lang/String.html) parameterName, [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x, [Calendar](http://docs.google.com/java/util/Calendar.html) cal)            Sets the designated parameter to the given java.sql.Timestamp value, using the given Calendar object. |
| void | [**setTransactionIsolation**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTransactionIsolation(int))(int level)            Sets the transaction isolation property for this JDBC RowSet object to the given constant. |
| void | [**setType**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setType(int))(int type)            Sets the type for this RowSet object to the specified type. |
| void | [**setTypeMap**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTypeMap(java.util.Map))([Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> map)            Installs the given java.util.Map object as the type map associated with the Connection object for this RowSet object. |
| void | [**setUnicodeStream**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setUnicodeStream(int,%20java.io.InputStream,%20int))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x, int length)  **Deprecated.** *getCharacterStream should be used in its place* |
| void | [**setURL**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setURL(int,%20java.net.URL))(int parameterIndex, [URL](http://docs.google.com/java/net/URL.html) x)            Sets the designated parameter to the given java.net.URL value. |
| void | [**setUrl**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setUrl(java.lang.String))([String](http://docs.google.com/java/lang/String.html) url)            Sets the Url property for this RowSet object to the given String object and sets the dataSource name property to null. |
| void | [**setUsername**](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setUsername(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Sets the username property for this RowSet object to the given user name. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### UNICODE\_STREAM\_PARAM

public static final int **UNICODE\_STREAM\_PARAM**

A constant indicating to a RowSetReaderImpl object that a given parameter is a Unicode stream. This RowSetReaderImpl object is provided as an extension of the SyncProvider abstract class defined in the SyncFactory static factory SPI mechanism.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.BaseRowSet.UNICODE_STREAM_PARAM)

### BINARY\_STREAM\_PARAM

public static final int **BINARY\_STREAM\_PARAM**

A constant indicating to a RowSetReaderImpl object that a given parameter is a binary stream. A RowSetReaderImpl object is provided as an extension of the SyncProvider abstract class defined in the SyncFactory static factory SPI mechanism.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.BaseRowSet.BINARY_STREAM_PARAM)

### ASCII\_STREAM\_PARAM

public static final int **ASCII\_STREAM\_PARAM**

A constant indicating to a RowSetReaderImpl object that a given parameter is an ASCII stream. A RowSetReaderImpl object is provided as an extension of the SyncProvider abstract class defined in the SyncFactory static factory SPI mechanism.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.BaseRowSet.ASCII_STREAM_PARAM)

### binaryStream

protected [InputStream](http://docs.google.com/java/io/InputStream.html) **binaryStream**

The InputStream object that will be returned by the method getBinaryStream, which is specified in the ResultSet interface.

### unicodeStream

protected [InputStream](http://docs.google.com/java/io/InputStream.html) **unicodeStream**

The InputStream object that will be returned by the method getUnicodeStream, which is specified in the ResultSet interface.

### asciiStream

protected [InputStream](http://docs.google.com/java/io/InputStream.html) **asciiStream**

The InputStream object that will be returned by the method getAsciiStream, which is specified in the ResultSet interface.

### charStream

protected [Reader](http://docs.google.com/java/io/Reader.html) **charStream**

The Reader object that will be returned by the method getCharacterStream, which is specified in the ResultSet interface.

| **Constructor Detail** |
| --- |

### BaseRowSet

public **BaseRowSet**()

Constructs a new BaseRowSet object initialized with a default Vector object for its listeners field. The other default values with which it is initialized are listed in Section 6.0 of the class comment for this class.

| **Method Detail** |
| --- |

### initParams

protected void **initParams**()

Performs the necessary internal configurations and initializations to allow any JDBC RowSet implementation to start using the standard facilities provided by a BaseRowSet instance. This method **should** be called after the RowSet object has been instantiated to correctly initialize all parameters. This method **should** never be called by an application, but is called from with a RowSet implementation extending this class.

### addRowSetListener

public void **addRowSetListener**([RowSetListener](http://docs.google.com/javax/sql/RowSetListener.html) listener)

The listener will be notified whenever an event occurs on this RowSet object.

A listener might, for example, be a table or graph that needs to be updated in order to accurately reflect the current state of the RowSet object.

**Note**: if the RowSetListener object is null, this method silently discards the null value and does not add a null reference to the set of listeners.

**Note**: if the listener is already set, and the new RowSetListerner instance is added to the set of listeners already registered to receive event notifications from this RowSet.

**Parameters:**listener - an object that has implemented the javax.sql.RowSetListener interface and wants to be notified of any events that occur on this RowSet object; May be null.**See Also:**[removeRowSetListener(javax.sql.RowSetListener)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#removeRowSetListener(javax.sql.RowSetListener))

### removeRowSetListener

public void **removeRowSetListener**([RowSetListener](http://docs.google.com/javax/sql/RowSetListener.html) listener)

Removes the designated object from this RowSet object's list of listeners. If the given argument is not a registered listener, this method does nothing. **Note**: if the RowSetListener object is null, this method silently discards the null value.

**Parameters:**listener - a RowSetListener object that is on the list of listeners for this RowSet object**See Also:**[addRowSetListener(javax.sql.RowSetListener)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#addRowSetListener(javax.sql.RowSetListener))

### notifyCursorMoved

protected void **notifyCursorMoved**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Notifies all of the listeners registered with this RowSet object that its cursor has moved.

When an application calls a method to move the cursor, that method moves the cursor and then calls this method internally. An application **should** never invoke this method directly.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the class extending the BaseRowSet abstract class does not implement the RowSet interface or one of it's sub-interfaces.

### notifyRowChanged

protected void **notifyRowChanged**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Notifies all of the listeners registered with this RowSet object that one of its rows has changed.

When an application calls a method that changes a row, such as the CachedRowSet methods insertRow, updateRow, or deleteRow, that method calls notifyRowChanged internally. An application **should** never invoke this method directly.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the class extending the BaseRowSet abstract class does not implement the RowSet interface or one of it's sub-interfaces.

### notifyRowSetChanged

protected void **notifyRowSetChanged**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Notifies all of the listeners registered with this RowSet object that its entire contents have changed.

When an application calls methods that change the entire contents of the RowSet object, such as the CachedRowSet methods execute, populate, restoreOriginal, or release, that method calls notifyRowSetChanged internally (either directly or indirectly). An application **should** never invoke this method directly.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the class extending the BaseRowSet abstract class does not implement the RowSet interface or one of it's sub-interfaces.

### getCommand

public [String](http://docs.google.com/java/lang/String.html) **getCommand**()

Retrieves the SQL query that is the command for this RowSet object. The command property contains the query that will be executed to populate this RowSet object.

The SQL query returned by this method is used by RowSet methods such as execute and populate, which may be implemented by any class that extends the BaseRowSet abstract class and implements one or more of the standard JSR-114 RowSet interfaces.

The command is used by the RowSet object's reader to obtain a ResultSet object. The reader then reads the data from the ResultSet object and uses it to to populate this RowSet object.

The default value for the command property is null.

**Returns:**the String that is the value for this RowSet object's command property; may be null**See Also:**[setCommand(java.lang.String)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setCommand(java.lang.String))

### setCommand

public void **setCommand**([String](http://docs.google.com/java/lang/String.html) cmd)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets this RowSet object's command property to the given String object and clears the parameters, if any, that were set for the previous command.

The command property may not be needed if the RowSet object gets its data from a source that does not support commands, such as a spreadsheet or other tabular file. Thus, this property is optional and may be null.

**Parameters:**cmd - a String object containing an SQL query that will be set as this RowSet object's command property; may be null but may not be an empty string **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an empty string is provided as the command value**See Also:**[getCommand()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getCommand())

### getUrl

public [String](http://docs.google.com/java/lang/String.html) **getUrl**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the JDBC URL that this RowSet object's javax.sql.Reader object uses to make a connection with a relational database using a JDBC technology-enabled driver.

The Url property will be null if the underlying data source is a non-SQL data source, such as a spreadsheet or an XML data source.

**Returns:**a String object that contains the JDBC URL used to establish the connection for this RowSet object; may be null (default value) if not set **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs retrieving the URL value**See Also:**[setUrl(java.lang.String)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setUrl(java.lang.String))

### setUrl

public void **setUrl**([String](http://docs.google.com/java/lang/String.html) url)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the Url property for this RowSet object to the given String object and sets the dataSource name property to null. The Url property is a JDBC URL that is used when the connection is created using a JDBC technology-enabled driver ("JDBC driver") and the DriverManager. The correct JDBC URL for the specific driver to be used can be found in the driver documentation. Although there are guidelines for for how a JDBC URL is formed, a driver vendor can specify any String object except one with a length of 0 (an empty string).

Setting the Url property is optional if connections are established using a DataSource object instead of the DriverManager. The driver will use either the URL property or the dataSourceName property to create a connection, whichever was specified most recently. If an application uses a JDBC URL, it must load a JDBC driver that accepts the JDBC URL before it uses the RowSet object to connect to a database. The RowSet object will use the URL internally to create a database connection in order to read or write data.

**Parameters:**url - a String object that contains the JDBC URL that will be used to establish the connection to a database for this RowSet object; may be null but must not be an empty string **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs setting the Url property or the parameter supplied is a string with a length of 0 (an empty string)**See Also:**[getUrl()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getUrl())

### getDataSourceName

public [String](http://docs.google.com/java/lang/String.html) **getDataSourceName**()

Returns the logical name that when supplied to a naming service that uses the Java Naming and Directory Interface (JNDI) API, will retrieve a javax.sql.DataSource object. This DataSource object can be used to establish a connection to the data source that it represents.

Users should set either the url or the data source name property. The driver will use the property set most recently to establish a connection.

**Returns:**a String object that identifies the DataSource object to be used for making a connection; if no logical name has been set, null is returned.**See Also:**[setDataSourceName(java.lang.String)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setDataSourceName(java.lang.String))

### setDataSourceName

public void **setDataSourceName**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the DataSource name property for this RowSet object to the given logical name and sets this RowSet object's Url property to null. The name must have been bound to a DataSource object in a JNDI naming service so that an application can do a lookup using that name to retrieve the DataSource object bound to it. The DataSource object can then be used to establish a connection to the data source it represents.

Users should set either the Url property or the dataSourceName property. If both properties are set, the driver will use the property set most recently.

**Parameters:**name - a String object with the name that can be supplied to a naming service based on JNDI technology to retrieve the DataSource object that can be used to get a connection; may be null but must not be an empty string **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an empty string is provided as the DataSource name**See Also:**[getDataSourceName()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getDataSourceName())

### getUsername

public [String](http://docs.google.com/java/lang/String.html) **getUsername**()

Returns the user name used to create a database connection. Because it is not serialized, the username property is set at runtime before calling the method execute.

**Returns:**the String object containing the user name that is supplied to the data source to create a connection; may be null (default value) if not set**See Also:**[setUsername(java.lang.String)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setUsername(java.lang.String))

### setUsername

public void **setUsername**([String](http://docs.google.com/java/lang/String.html) name)

Sets the username property for this RowSet object to the given user name. Because it is not serialized, the username property is set at run time before calling the method execute.

**Parameters:**name - the String object containing the user name that is supplied to the data source to create a connection. It may be null.**See Also:**[getUsername()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getUsername())

### getPassword

public [String](http://docs.google.com/java/lang/String.html) **getPassword**()

Returns the password used to create a database connection for this RowSet object. Because the password property is not serialized, it is set at run time before calling the method execute. The default value is null

**Returns:**the String object that represents the password that must be supplied to the database to create a connection**See Also:**[setPassword(java.lang.String)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setPassword(java.lang.String))

### setPassword

public void **setPassword**([String](http://docs.google.com/java/lang/String.html) pass)

Sets the password used to create a database connection for this RowSet object to the given String object. Because the password property is not serialized, it is set at run time before calling the method execute.

**Parameters:**pass - the String object that represents the password that is supplied to the database to create a connection. It may be null.**See Also:**[getPassword()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getPassword())

### setType

public void **setType**(int type)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the type for this RowSet object to the specified type. The default type is ResultSet.TYPE\_SCROLL\_INSENSITIVE.

**Parameters:**type - one of the following constants: ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the parameter supplied is not one of the following constants: ResultSet.TYPE\_FORWARD\_ONLY or ResultSet.TYPE\_SCROLL\_INSENSITIVE ResultSet.TYPE\_SCROLL\_SENSITIVE**See Also:**[getConcurrency()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getConcurrency()), [getType()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getType())

### getType

public int **getType**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns the type of this RowSet object. The type is initially determined by the statement that created the RowSet object. The RowSet object can call the method setType at any time to change its type. The default is TYPE\_SCROLL\_INSENSITIVE.

**Returns:**the type of this JDBC RowSet object, which must be one of the following: ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs getting the type of of this RowSet object**See Also:**[setType(int)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setType(int))

### setConcurrency

public void **setConcurrency**(int concurrency)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the concurrency for this RowSet object to the specified concurrency. The default concurrency for any RowSet object (connected or disconnected) is ResultSet.CONCUR\_UPDATABLE, but this method may be called at any time to change the concurrency.

**Parameters:**concurrency - one of the following constants: ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the parameter supplied is not one of the following constants: ResultSet.CONCUR\_UPDATABLE or ResultSet.CONCUR\_READ\_ONLY**See Also:**[getConcurrency()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getConcurrency()), [isReadOnly()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#isReadOnly())

### isReadOnly

public boolean **isReadOnly**()

Returns a boolean indicating whether this RowSet object is read-only. Any attempts to update a read-only RowSet object will result in an SQLException being thrown. By default, rowsets are updatable if updates are possible.

**Returns:**true if this RowSet object cannot be updated; false otherwise**See Also:**[setConcurrency(int)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setConcurrency(int)), [setReadOnly(boolean)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setReadOnly(boolean))

### setReadOnly

public void **setReadOnly**(boolean value)

Sets this RowSet object's readOnly property to the given boolean.

**Parameters:**value - true to indicate that this RowSet object is read-only; false to indicate that it is updatable

### getTransactionIsolation

public int **getTransactionIsolation**()

Returns the transaction isolation property for this RowSet object's connection. This property represents the transaction isolation level requested for use in transactions.

For RowSet implementations such as the CachedRowSet that operate in a disconnected environment, the SyncProvider object offers complementary locking and data integrity options. The options described below are pertinent only to connected RowSet objects (JdbcRowSet objects).

**Returns:**one of the following constants: Connection.TRANSACTION\_NONE, Connection.TRANSACTION\_READ\_UNCOMMITTED, Connection.TRANSACTION\_READ\_COMMITTED, Connection.TRANSACTION\_REPEATABLE\_READ, or Connection.TRANSACTION\_SERIALIZABLE**See Also:**[SyncFactory](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html), [SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html), [setTransactionIsolation(int)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setTransactionIsolation(int))

### setTransactionIsolation

public void **setTransactionIsolation**(int level)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the transaction isolation property for this JDBC RowSet object to the given constant. The DBMS will use this transaction isolation level for transactions if it can.

For RowSet implementations such as the CachedRowSet that operate in a disconnected environment, the SyncProvider object being used offers complementary locking and data integrity options. The options described below are pertinent only to connected RowSet objects (JdbcRowSet objects).

**Parameters:**level - one of the following constants, listed in ascending order: Connection.TRANSACTION\_NONE, Connection.TRANSACTION\_READ\_UNCOMMITTED, Connection.TRANSACTION\_READ\_COMMITTED, Connection.TRANSACTION\_REPEATABLE\_READ, or Connection.TRANSACTION\_SERIALIZABLE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the given parameter is not one of the Connection constants**See Also:**[SyncFactory](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html), [SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html), [getTransactionIsolation()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getTransactionIsolation())

### getTypeMap

public [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> **getTypeMap**()

Retrieves the type map associated with the Connection object for this RowSet object.

Drivers that support the JDBC 3.0 API will create Connection objects with an associated type map. This type map, which is initially empty, can contain one or more fully-qualified SQL names and Class objects indicating the class to which the named SQL value will be mapped. The type mapping specified in the connection's type map is used for custom type mapping when no other type map supersedes it.

If a type map is explicitly supplied to a method that can perform custom mapping, that type map supersedes the connection's type map.

**Returns:**the java.util.Map object that is the type map for this RowSet object's connection

### setTypeMap

public void **setTypeMap**([Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> map)

Installs the given java.util.Map object as the type map associated with the Connection object for this RowSet object. The custom mapping indicated in this type map will be used unless a different type map is explicitly supplied to a method, in which case the type map supplied will be used.

**Parameters:**map - a java.util.Map object that contains the mapping from SQL type names for user defined types (UDT) to classes in the Java programming language. Each entry in the Map object consists of the fully qualified SQL name of a UDT and the Class object for the SQLData implementation of that UDT. May be null.

### getMaxFieldSize

public int **getMaxFieldSize**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the maximum number of bytes that can be used for a column value in this RowSet object. This limit applies only to columns that hold values of the following types: BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, and LONGVARCHAR. If the limit is exceeded, the excess data is silently discarded.

**Returns:**an int indicating the current maximum column size limit; zero means that there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs internally determining the maximum limit of the column size

### setMaxFieldSize

public void **setMaxFieldSize**(int max)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the maximum number of bytes that can be used for a column value in this RowSet object to the given number. This limit applies only to columns that hold values of the following types: BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, and LONGVARCHAR. If the limit is exceeded, the excess data is silently discarded. For maximum portability, it is advisable to use values greater than 256.

**Parameters:**max - an int indicating the new maximum column size limit; zero means that there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if (1) an error occurs internally setting the maximum limit of the column size or (2) a size of less than 0 is set

### getMaxRows

public int **getMaxRows**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the maximum number of rows that this RowSet object may contain. If this limit is exceeded, the excess rows are silently dropped.

**Returns:**an int indicating the current maximum number of rows; zero means that there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs internally determining the maximum limit of rows that a Rowset object can contain

### setMaxRows

public void **setMaxRows**(int max)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the maximum number of rows that this RowSet object may contain to the given number. If this limit is exceeded, the excess rows are silently dropped.

**Parameters:**max - an int indicating the current maximum number of rows; zero means that there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs internally setting the maximum limit on the number of rows that a JDBC RowSet object can contain; or if *max* is less than 0; or if *max* is less than the fetchSize of the RowSet

### setEscapeProcessing

public void **setEscapeProcessing**(boolean enable)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets to the given boolean whether or not the driver will scan for escape syntax and do escape substitution before sending SQL statements to the database. The default is for the driver to do escape processing.

Note: Since PreparedStatement objects have usually been parsed prior to making this call, disabling escape processing for prepared statements will likely have no effect.

**Parameters:**enable - true to enable escape processing; false to disable it **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs setting the underlying JDBC technology-enabled driver to process the escape syntax

### getQueryTimeout

public int **getQueryTimeout**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the maximum number of seconds the driver will wait for a query to execute. If the limit is exceeded, an SQLException is thrown.

**Returns:**the current query timeout limit in seconds; zero means that there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs in determining the query time-out value

### setQueryTimeout

public void **setQueryTimeout**(int seconds)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets to the given number the maximum number of seconds the driver will wait for a query to execute. If the limit is exceeded, an SQLException is thrown.

**Parameters:**seconds - the new query time-out limit in seconds; zero means that there is no limit; must not be less than zero **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs setting the query time-out or if the query time-out value is less than 0

### getShowDeleted

public boolean **getShowDeleted**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves a boolean indicating whether rows marked for deletion appear in the set of current rows. The default value is false.

Note: Allowing deleted rows to remain visible complicates the behavior of some of the methods. However, most RowSet object users can simply ignore this extra detail because only sophisticated applications will likely want to take advantage of this feature.

**Returns:**true if deleted rows are visible; false otherwise **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs determining if deleted rows are visible or not**See Also:**[setShowDeleted(boolean)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setShowDeleted(boolean))

### setShowDeleted

public void **setShowDeleted**(boolean value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the property showDeleted to the given boolean value, which determines whether rows marked for deletion appear in the set of current rows.

**Parameters:**value - true if deleted rows should be shown; false otherwise **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs setting whether deleted rows are visible or not**See Also:**[getShowDeleted()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getShowDeleted())

### getEscapeProcessing

public boolean **getEscapeProcessing**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Ascertains whether escape processing is enabled for this RowSet object.

**Returns:**true if escape processing is turned on; false otherwise **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs determining if escape processing is enabled or not or if the internal escape processing trigger has not been enabled

### setFetchDirection

public void **setFetchDirection**(int direction)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Gives the driver a performance hint as to the direction in which the rows in this RowSet object will be processed. The driver may ignore this hint.

A RowSet object inherits the default properties of the ResultSet object from which it got its data. That ResultSet object's default fetch direction is set by the Statement object that created it.

This method applies to a RowSet object only while it is connected to a database using a JDBC driver.

A RowSet object may use this method at any time to change its setting for the fetch direction.

**Parameters:**direction - one of ResultSet.FETCH\_FORWARD, ResultSet.FETCH\_REVERSE, or ResultSet.FETCH\_UNKNOWN **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if (1) the RowSet type is TYPE\_FORWARD\_ONLY and the given fetch direction is not FETCH\_FORWARD or (2) the given fetch direction is not one of the following: ResultSet.FETCH\_FORWARD, ResultSet.FETCH\_REVERSE, or ResultSet.FETCH\_UNKNOWN**See Also:**[getFetchDirection()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getFetchDirection())

### getFetchDirection

public int **getFetchDirection**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves this RowSet object's current setting for the fetch direction. The default type is ResultSet.FETCH\_FORWARD

**Returns:**one of ResultSet.FETCH\_FORWARD, ResultSet.FETCH\_REVERSE, or ResultSet.FETCH\_UNKNOWN **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs in determining the current fetch direction for fetching rows**See Also:**[setFetchDirection(int)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setFetchDirection(int))

### setFetchSize

public void **setFetchSize**(int rows)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the fetch size for this RowSet object to the given number of rows. The fetch size gives a JDBC technology-enabled driver ("JDBC driver") a hint as to the number of rows that should be fetched from the database when more rows are needed for this RowSet object. If the fetch size specified is zero, the driver ignores the value and is free to make its own best guess as to what the fetch size should be.

A RowSet object inherits the default properties of the ResultSet object from which it got its data. That ResultSet object's default fetch size is set by the Statement object that created it.

This method applies to a RowSet object only while it is connected to a database using a JDBC driver. For connected RowSet implementations such as JdbcRowSet, this method has a direct and immediate effect on the underlying JDBC driver.

A RowSet object may use this method at any time to change its setting for the fetch size.

For RowSet implementations such as CachedRowSet, which operate in a disconnected environment, the SyncProvider object being used may leverage the fetch size to poll the data source and retrieve a number of rows that do not exceed the fetch size and that may form a subset of the actual rows returned by the original query. This is an implementation variance determined by the specific SyncProvider object employed by the disconnected RowSet object.

**Parameters:**rows - the number of rows to fetch; 0 to let the driver decide what the best fetch size is; must not be less than 0 or more than the maximum number of rows allowed for this RowSet object (the number returned by a call to the method [getMaxRows()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getMaxRows())) **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the specified fetch size is less than 0 or more than the limit for the maximum number of rows**See Also:**[getFetchSize()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getFetchSize())

### getFetchSize

public int **getFetchSize**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns the fetch size for this RowSet object. The default value is zero.

**Returns:**the number of rows suggested as the fetch size when this RowSet object needs more rows from the database **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs determining the number of rows in the current fetch size**See Also:**[setFetchSize(int)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setFetchSize(int))

### getConcurrency

public int **getConcurrency**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns the concurrency for this RowSet object. The default is CONCUR\_UPDATABLE for both connected and disconnected RowSet objects.

An application can call the method setConcurrency at any time to change a RowSet object's concurrency.

**Returns:**the concurrency type for this RowSet object, which must be one of the following: ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs getting the concurrency of this RowSet object**See Also:**[setConcurrency(int)](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#setConcurrency(int)), [isReadOnly()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#isReadOnly())

### setNull

public void **setNull**(int parameterIndex,  
 int sqlType)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to SQL NULL. Note that the parameter's SQL type must be specified using one of the type codes defined in java.sql.Types. This SQL type is specified in the second parameter.

Note that the second parameter tells the DBMS the data type of the value being set to NULL. Some DBMSs require this information, so it is required in order to make code more portable.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setNull has been called will return an Object array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is null. The second element is the value set for *sqlType*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the second placeholder parameter is being set to null, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greatersqlType - an int that is one of the SQL type codes defined in the class [Types](http://docs.google.com/java/sql/Types.html). If a non-standard *sqlType* is supplied, this method will not throw a SQLException. This allows implicit support for non-standard SQL types. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or the given parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setNull

public void **setNull**(int parameterIndex,  
 int sqlType,  
 [String](http://docs.google.com/java/lang/String.html) typeName)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to SQL NULL. Although this version of the method setNull is intended for user-defined and REF parameters, this method may be used to set a null parameter for any JDBC type. The following are user-defined types: STRUCT, DISTINCT, and JAVA\_OBJECT, and named array types.

**Note:** To be portable, applications must give the SQL type code and the fully qualified SQL type name when specifying a NULL user-defined or REF parameter. In the case of a user-defined type, the name is the type name of the parameter itself. For a REF parameter, the name is the type name of the referenced type. If a JDBC technology-enabled driver does not need the type code or type name information, it may ignore it.

If the parameter does not have a user-defined or REF type, the given typeName parameter is ignored.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setNull has been called will return an Object array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is null. The second element is the value set for *sqlType*, and the third element is the value set for *typeName*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the second placeholder parameter is being set to null, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greatersqlType - a value from java.sql.TypestypeName - the fully qualified name of an SQL user-defined type, which is ignored if the parameter is not a user-defined type or REF value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the given parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setBoolean

public void **setBoolean**(int parameterIndex,  
 boolean x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given boolean in the Java programming language. The driver converts this to an SQL BIT value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute, populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setByte

public void **setByte**(int parameterIndex,  
 byte x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given byte in the Java programming language. The driver converts this to an SQL TINYINT value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setShort

public void **setShort**(int parameterIndex,  
 short x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given short in the Java programming language. The driver converts this to an SQL SMALLINT value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setInt

public void **setInt**(int parameterIndex,  
 int x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to an int in the Java programming language. The driver converts this to an SQL INTEGER value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setLong

public void **setLong**(int parameterIndex,  
 long x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given long in the Java programming language. The driver converts this to an SQL BIGINT value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setFloat

public void **setFloat**(int parameterIndex,  
 float x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given float in the Java programming language. The driver converts this to an SQL FLOAT value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setDouble

public void **setDouble**(int parameterIndex,  
 double x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given double in the Java programming language. The driver converts this to an SQL DOUBLE value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class. S

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setBigDecimal

public void **setBigDecimal**(int parameterIndex,  
 [BigDecimal](http://docs.google.com/java/math/BigDecimal.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.lang.BigDecimal value. The driver converts this to an SQL NUMERIC value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

Note: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setString

public void **setString**(int parameterIndex,  
 [String](http://docs.google.com/java/lang/String.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given String value. The driver converts this to an SQL VARCHAR or LONGVARCHAR value (depending on the argument's size relative to the driver's limits on VARCHAR values) when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setBytes

public void **setBytes**(int parameterIndex,  
 byte[] x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given array of bytes. The driver converts this to an SQL VARBINARY or LONGVARBINARY value (depending on the argument's size relative to the driver's limits on VARBINARY values) when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setDate

public void **setDate**(int parameterIndex,  
 [Date](http://docs.google.com/java/sql/Date.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Date value. The driver converts this to an SQL DATE value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setDate has been called will return an array with the value to be set for placeholder parameter number *parameterIndex* being the Date object supplied as the second parameter. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setTime

public void **setTime**(int parameterIndex,  
 [Time](http://docs.google.com/java/sql/Time.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Time value. The driver converts this to an SQL TIME value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of the method setTime has been called will return an array of the parameters that have been set. The parameter to be set for parameter placeholder number *parameterIndex* will be the Time object that was set as the second parameter to this method.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a java.sql.Time object, which is to be set as the value for placeholder parameter *parameterIndex* **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setTimestamp

public void **setTimestamp**(int parameterIndex,  
 [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Timestamp value. The driver converts this to an SQL TIMESTAMP value when it sends it to the database.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setTimestamp has been called will return an array with the value for parameter placeholder number *parameterIndex* being the Timestamp object that was supplied as the second parameter to this method. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a java.sql.Timestamp object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setAsciiStream

public void **setAsciiStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.io.InputStream object, which will have the specified number of bytes. The contents of the stream will be read and sent to the database. This method throws an SQLException object if the number of bytes read and sent to the database is not equal to *length*.

When a very large ASCII value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream object. A JDBC technology-enabled driver will read the data from the stream as needed until it reaches end-of-file. The driver will do any necessary conversion from ASCII to the database CHAR format.

**Note:** This stream object can be either a standard Java stream object or your own subclass that implements the standard interface.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

Note: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after setAsciiStream has been called will return an array containing the parameter values that have been set. The element in the array that represents the values set with this method will itself be an array. The first element of that array is the given java.io.InputStream object. The second element is the value set for *length*. The third element is an internal BaseRowSet constant specifying that the stream passed to this method is an ASCII stream. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the input stream being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the Java input stream that contains the ASCII parameter valuelength - the number of bytes in the stream. This is the number of bytes the driver will send to the DBMS; lengths of 0 or less are are undefined but will cause an invalid length exception to be thrown in the underlying JDBC driver. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs, the parameter index is out of bounds, or when connected to a data source, the number of bytes the driver reads and sends to the database is not equal to the number of bytes specified in *length***See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setAsciiStream

public void **setAsciiStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter in this RowSet object's command to the given input stream. When a very large ASCII value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream. Data will be read from the stream as needed until end-of-file is reached. The JDBC driver will do any necessary conversion from ASCII to the database char format.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setAsciiStream which takes a length parameter.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - the Java input stream that contains the ASCII parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setBinaryStream

public void **setBinaryStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.io.InputStream object, which will have the specified number of bytes. The contents of the stream will be read and sent to the database. This method throws an SQLException object if the number of bytes read and sent to the database is not equal to *length*.

When a very large binary value is input to a LONGVARBINARY parameter, it may be more practical to send it via a java.io.InputStream object. A JDBC technology-enabled driver will read the data from the stream as needed until it reaches end-of-file.

**Note:** This stream object can be either a standard Java stream object or your own subclass that implements the standard interface.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after setBinaryStream has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given java.io.InputStream object. The second element is the value set for *length*. The third element is an internal BaseRowSet constant specifying that the stream passed to this method is a binary stream. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the input stream being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the input stream that contains the binary value to be setlength - the number of bytes in the stream; lengths of 0 or less are are undefined but will cause an invalid length exception to be thrown in the underlying JDBC driver. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs, the parameter index is out of bounds, or when connected to a data source, the number of bytes the driver reads and sends to the database is not equal to the number of bytes specified in *length***See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setBinaryStream

public void **setBinaryStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter in this RowSet object's command to the given input stream. When a very large binary value is input to a LONGVARBINARY parameter, it may be more practical to send it via a java.io.InputStream object. The data will be read from the stream as needed until end-of-file is reached.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setBinaryStream which takes a length parameter.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - the java input stream which contains the binary parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setUnicodeStream

public void **setUnicodeStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

**Deprecated.** *getCharacterStream should be used in its place*

Sets the designated parameter to the given java.io.InputStream object, which will have the specified number of bytes. The contents of the stream will be read and sent to the database. This method throws an SQLException if the number of bytes read and sent to the database is not equal to *length*.

When a very large Unicode value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream object. A JDBC technology-enabled driver will read the data from the stream as needed, until it reaches end-of-file. The driver will do any necessary conversion from Unicode to the database CHAR format. The byte format of the Unicode stream must be Java UTF-8, as defined in the Java Virtual Machine Specification.

**Note:** This stream object can be either a standard Java stream object or your own subclass that implements the standard interface.

This method is deprecated; the method getCharacterStream should be used in its place.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Calls made to the method getParams after setUnicodeStream has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given java.io.InputStream object. The second element is the value set for *length*. The third element is an internal BaseRowSet constant specifying that the stream passed to this method is a Unicode stream. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the input stream being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the java.io.InputStream object that contains the UNICODE parameter valuelength - the number of bytes in the input stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs, the parameter index is out of bounds, or the number of bytes the driver reads and sends to the database is not equal to the number of bytes specified in *length***See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setCharacterStream

public void **setCharacterStream**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.io.Reader object, which will have the specified number of characters. The contents of the reader will be read and sent to the database. This method throws an SQLException if the number of bytes read and sent to the database is not equal to *length*.

When a very large Unicode value is input to a LONGVARCHAR parameter, it may be more practical to send it via a Reader object. A JDBC technology-enabled driver will read the data from the stream as needed until it reaches end-of-file. The driver will do any necessary conversion from Unicode to the database CHAR format. The byte format of the Unicode stream must be Java UTF-8, as defined in the Java Virtual Machine Specification.

**Note:** This stream object can be either a standard Java stream object or your own subclass that implements the standard interface.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after setCharacterStream has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given java.io.Reader object. The second element is the value set for *length*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the reader being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterreader - the Reader object that contains the Unicode datalength - the number of characters in the stream; lengths of 0 or less are undefined but will cause an invalid length exception to be thrown in the underlying JDBC driver. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs, the parameter index is out of bounds, or when connected to a data source, the number of bytes the driver reads and sends to the database is not equal to the number of bytes specified in *length***See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setCharacterStream

public void **setCharacterStream**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter in this RowSet object's command to the given Reader object. When a very large UNICODE value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.Reader object. The data will be read from the stream as needed until end-of-file is reached. The JDBC driver will do any necessary conversion from UNICODE to the database char format.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setCharacterStream which takes a length parameter.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...reader - the java.io.Reader object that contains the Unicode data **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setObject

public void **setObject**(int parameterIndex,  
 [Object](http://docs.google.com/java/lang/Object.html) x,  
 int targetSqlType,  
 int scale)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to an Object in the Java programming language. The second parameter must be an Object type. For integral values, the java.lang equivalent objects should be used. For example, use the class Integer for an int.

The driver converts this object to the specified target SQL type before sending it to the database. If the object has a custom mapping (is of a class implementing SQLData), the driver should call the method SQLData.writeSQL to write the object to the SQL data stream. If, on the other hand, the object is of a class implementing Ref, Blob, Clob, Struct, or Array, the driver should pass it to the database as a value of the corresponding SQL type.

Note that this method may be used to pass database- specific abstract data types.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSetexecute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setObject has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given Object instance, and the second element is the value set for *targetSqlType*. The third element is the value set for *scale*, which the driver will ignore if the type of the object being set is not java.sql.Types.NUMERIC or java.sql.Types.DECIMAL. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the object being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the Object containing the input parameter value; must be an Object typetargetSqlType - the SQL type (as defined in java.sql.Types) to be sent to the database. The scale argument may further qualify this type. If a non-standard *targetSqlType* is supplied, this method will not throw a SQLException. This allows implicit support for non-standard SQL types.scale - for the types java.sql.Types.DECIMAL and java.sql.Types.NUMERIC, this is the number of digits after the decimal point. For all other types, this value will be ignored. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setObject

public void **setObject**(int parameterIndex,  
 [Object](http://docs.google.com/java/lang/Object.html) x,  
 int targetSqlType)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the value of the designated parameter with the given Object value. This method is like setObject(int parameterIndex, Object x, int targetSqlType, int scale) except that it assumes a scale of zero.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setObject has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given Object instance. The second element is the value set for *targetSqlType*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the object being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the Object containing the input parameter value; must be an Object typetargetSqlType - the SQL type (as defined in java.sql.Types) to be sent to the database. If a non-standard *targetSqlType* is supplied, this method will not throw a SQLException. This allows implicit support for non-standard SQL types. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setObject

public void **setObject**(int parameterIndex,  
 [Object](http://docs.google.com/java/lang/Object.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to an Object in the Java programming language. The second parameter must be an Object type. For integral values, the java.lang equivalent objects should be used. For example, use the class Integer for an int.

The JDBC specification defines a standard mapping from Java Object types to SQL types. The driver will use this standard mapping to convert the given object to its corresponding SQL type before sending it to the database. If the object has a custom mapping (is of a class implementing SQLData), the driver should call the method SQLData.writeSQL to write the object to the SQL data stream.

If, on the other hand, the object is of a class implementing Ref, Blob, Clob, Struct, or Array, the driver should pass it to the database as a value of the corresponding SQL type.

This method throws an exception if there is an ambiguity, for example, if the object is of a class implementing more than one interface.

Note that this method may be used to pass database-specific abstract data types.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

After this method has been called, a call to the method getParams will return an object array of the current command parameters, which will include the Object set for placeholder parameter number parameterIndex. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - the object containing the input parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs the parameter index is out of bounds, or there is ambiguity in the implementation of the object being set**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setRef

public void **setRef**(int parameterIndex,  
 [Ref](http://docs.google.com/java/sql/Ref.html) ref)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Ref object in the Java programming language. The driver converts this to an SQL REF value when it sends it to the database. Internally, the Ref is represented as a SerialRef to ensure serializability.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

After this method has been called, a call to the method getParams will return an object array of the current command parameters, which will include the Ref object set for placeholder parameter number parameterIndex. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterref - a Ref object representing an SQL REF value; cannot be null **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs; the parameter index is out of bounds or the Ref object is null; or the Ref object returns a null base type name.**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams()), [SerialRef](http://docs.google.com/javax/sql/rowset/serial/SerialRef.html)

### setBlob

public void **setBlob**(int parameterIndex,  
 [Blob](http://docs.google.com/java/sql/Blob.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Blob object in the Java programming language. The driver converts this to an SQL BLOB value when it sends it to the database. Internally, the Blob is represented as a SerialBlob to ensure serializability.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces. NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

After this method has been called, a call to the method getParams will return an object array of the current command parameters, which will include the Blob object set for placeholder parameter number parameterIndex. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a Blob object representing an SQL BLOB value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams()), [SerialBlob](http://docs.google.com/javax/sql/rowset/serial/SerialBlob.html)

### setClob

public void **setClob**(int parameterIndex,  
 [Clob](http://docs.google.com/java/sql/Clob.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Clob object in the Java programming language. The driver converts this to an SQL CLOB value when it sends it to the database. Internally, the Clob is represented as a SerialClob to ensure serializability.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

After this method has been called, a call to the method getParams will return an object array of the current command parameters, which will include the Clob object set for placeholder parameter number parameterIndex. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a Clob object representing an SQL CLOB value; cannot be null **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs; the parameter index is out of bounds or the Clob is null**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams()), [SerialBlob](http://docs.google.com/javax/sql/rowset/serial/SerialBlob.html)

### setArray

public void **setArray**(int parameterIndex,  
 [Array](http://docs.google.com/java/sql/Array.html) array)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to an Array object in the Java programming language. The driver converts this to an SQL ARRAY value when it sends it to the database. Internally, the Array is represented as a SerialArray to ensure serializability.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

Note: JdbcRowSet does not require the populate method as it is undefined in this class.

After this method has been called, a call to the method getParams will return an object array of the current command parameters, which will include the Array object set for placeholder parameter number parameterIndex. Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is element number *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterarray - an Array object representing an SQL ARRAY value; cannot be null. The Array object passed to this method must return a non-null Object for all getArray() method calls. A null value will cause a SQLException to be thrown. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs; the parameter index is out of bounds or the ARRAY is null**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams()), [SerialArray](http://docs.google.com/javax/sql/rowset/serial/SerialArray.html)

### setDate

public void **setDate**(int parameterIndex,  
 [Date](http://docs.google.com/java/sql/Date.html) x,  
 [Calendar](http://docs.google.com/java/util/Calendar.html) cal)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Date object. When the DBMS does not store time zone information, the driver will use the given Calendar object to construct the SQL DATE value to send to the database. With a Calendar object, the driver can calculate the date taking into account a custom time zone. If no Calendar object is specified, the driver uses the time zone of the Virtual Machine that is running the application.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setDate has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given java.sql.Date object. The second element is the value set for *cal*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the date being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a java.sql.Date object representing an SQL DATE valuecal - a java.util.Calendar object to use when when constructing the date **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setTime

public void **setTime**(int parameterIndex,  
 [Time](http://docs.google.com/java/sql/Time.html) x,  
 [Calendar](http://docs.google.com/java/util/Calendar.html) cal)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Time object. The driver converts this to an SQL TIME value when it sends it to the database.

When the DBMS does not store time zone information, the driver will use the given Calendar object to construct the SQL TIME value to send to the database. With a Calendar object, the driver can calculate the date taking into account a custom time zone. If no Calendar object is specified, the driver uses the time zone of the Virtual Machine that is running the application.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setTime has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given java.sql.Time object. The second element is the value set for *cal*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the time being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a java.sql.Time objectcal - the java.util.Calendar object the driver can use to construct the time **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### setTimestamp

public void **setTimestamp**(int parameterIndex,  
 [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x,  
 [Calendar](http://docs.google.com/java/util/Calendar.html) cal)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Timestamp object. The driver converts this to an SQL TIMESTAMP value when it sends it to the database.

When the DBMS does not store time zone information, the driver will use the given Calendar object to construct the SQL TIMESTAMP value to send to the database. With a Calendar object, the driver can calculate the timestamp taking into account a custom time zone. If no Calendar object is specified, the driver uses the time zone of the Virtual Machine that is running the application.

The parameter value set by this method is stored internally and will be supplied as the appropriate parameter in this RowSet object's command when the method execute is called. Methods such as execute and populate must be provided in any class that extends this class and implements one or more of the standard JSR-114 RowSet interfaces.

NOTE: JdbcRowSet does not require the populate method as it is undefined in this class.

Calls made to the method getParams after this version of setTimestamp has been called will return an array containing the parameter values that have been set. In that array, the element that represents the values set with this method will itself be an array. The first element of that array is the given java.sql.Timestamp object. The second element is the value set for *cal*. The parameter number is indicated by an element's position in the array returned by the method getParams, with the first element being the value for the first placeholder parameter, the second element being the value for the second placeholder parameter, and so on. In other words, if the timestamp being set is the value for the second placeholder parameter, the array containing it will be the second element in the array returned by getParams.

Note that because the numbering of elements in an array starts at zero, the array element that corresponds to placeholder parameter number *parameterIndex* is *parameterIndex* -1.

**Parameters:**parameterIndex - the ordinal number of the placeholder parameter in this RowSet object's command that is to be set. The first parameter is 1, the second is 2, and so on; must be 1 or greaterx - a java.sql.Timestamp objectcal - the java.util.Calendar object the driver can use to construct the timestamp **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs or the parameter index is out of bounds**See Also:**[getParams()](http://docs.google.com/javax/sql/rowset/BaseRowSet.html#getParams())

### clearParameters

public void **clearParameters**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Clears all of the current parameter values in this RowSet object's internal representation of the parameters to be set in this RowSet object's command when it is executed.

In general, parameter values remain in force for repeated use in this RowSet object's command. Setting a parameter value with the setter methods automatically clears the value of the designated parameter and replaces it with the new specified value.

This method is called internally by the setCommand method to clear all of the parameters set for the previous command.

Furthermore, this method differs from the initParams method in that it maintains the schema of the RowSet object.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs clearing the parameters

### getParams

public [Object](http://docs.google.com/java/lang/Object.html)[] **getParams**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves an array containing the parameter values (both Objects and primitives) that have been set for this RowSet object's command and throws an SQLException object if all parameters have not been set. Before the command is sent to the DBMS to be executed, these parameters will be substituted for placeholder parameters in the PreparedStatement object that is the command for a RowSet implementation extending the BaseRowSet class.

Each element in the array that is returned is an Object instance that contains the values of the parameters supplied to a setter method. The order of the elements is determined by the value supplied for *parameterIndex*. If the setter method takes only the parameter index and the value to be set (possibly null), the array element will contain the value to be set (which will be expressed as an Object). If there are additional parameters, the array element will itself be an array containing the value to be set plus any additional parameter values supplied to the setter method. If the method sets a stream, the array element includes the type of stream being supplied to the method. These additional parameters are for the use of the driver or the DBMS and may or may not be used.

NOTE: Stored parameter values of types Array, Blob, Clob and Ref are returned as SerialArray, SerialBlob, SerialClob and SerialRef respectively.

**Returns:**an array of Object instances that includes the parameter values that may be set in this RowSet object's command; an empty array if no parameters have been set **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an error occurs retrieveing the object array of parameters of this RowSet object or if not all parameters have been set

### setNull

public void **setNull**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 int sqlType)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to SQL NULL.

**Note:** You must specify the parameter's SQL type.

**Parameters:**parameterName - the name of the parametersqlType - the SQL type code defined in java.sql.Types **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### setNull

public void **setNull**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 int sqlType,  
 [String](http://docs.google.com/java/lang/String.html) typeName)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to SQL NULL. This version of the method setNull should be used for user-defined types and REF type parameters. Examples of user-defined types include: STRUCT, DISTINCT, JAVA\_OBJECT, and named array types.

**Note:** To be portable, applications must give the SQL type code and the fully-qualified SQL type name when specifying a NULL user-defined or REF parameter. In the case of a user-defined type the name is the type name of the parameter itself. For a REF parameter, the name is the type name of the referenced type. If a JDBC driver does not need the type code or type name information, it may ignore it. Although it is intended for user-defined and Ref parameters, this method may be used to set a null parameter of any JDBC type. If the parameter does not have a user-defined or REF type, the given typeName is ignored.

**Parameters:**parameterName - the name of the parametersqlType - a value from java.sql.TypestypeName - the fully-qualified name of an SQL user-defined type; ignored if the parameter is not a user-defined type or SQL REF value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### setBoolean

public void **setBoolean**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 boolean x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java boolean value. The driver converts this to an SQL BIT or BOOLEAN value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getBoolean

### setByte

public void **setByte**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 byte x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java byte value. The driver converts this to an SQL TINYINT value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getByte

### setShort

public void **setShort**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 short x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java short value. The driver converts this to an SQL SMALLINT value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getShort

### setInt

public void **setInt**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 int x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java int value. The driver converts this to an SQL INTEGER value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getInt

### setLong

public void **setLong**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 long x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java long value. The driver converts this to an SQL BIGINT value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getLong

### setFloat

public void **setFloat**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 float x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java float value. The driver converts this to an SQL FLOAT value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getFloat

### setDouble

public void **setDouble**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 double x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java double value. The driver converts this to an SQL DOUBLE value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getDouble

### setBigDecimal

public void **setBigDecimal**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [BigDecimal](http://docs.google.com/java/math/BigDecimal.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.math.BigDecimal value. The driver converts this to an SQL NUMERIC value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getBigDecimal

### setString

public void **setString**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [String](http://docs.google.com/java/lang/String.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java String value. The driver converts this to an SQL VARCHAR or LONGVARCHAR value (depending on the argument's size relative to the driver's limits on VARCHAR values) when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getString

### setBytes

public void **setBytes**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 byte[] x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Java array of bytes. The driver converts this to an SQL VARBINARY or LONGVARBINARY (depending on the argument's size relative to the driver's limits on VARBINARY values) when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getBytes

### setTimestamp

public void **setTimestamp**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Timestamp value. The driver converts this to an SQL TIMESTAMP value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getTimestamp

### setAsciiStream

public void **setAsciiStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given input stream, which will have the specified number of bytes. When a very large ASCII value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream. Data will be read from the stream as needed until end-of-file is reached. The JDBC driver will do any necessary conversion from ASCII to the database char format.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Parameters:**parameterName - the name of the parameterx - the Java input stream that contains the ASCII parameter valuelength - the number of bytes in the stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### setBinaryStream

public void **setBinaryStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given input stream, which will have the specified number of bytes. When a very large binary value is input to a LONGVARBINARY parameter, it may be more practical to send it via a java.io.InputStream object. The data will be read from the stream as needed until end-of-file is reached.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Parameters:**parameterName - the name of the parameterx - the java input stream which contains the binary parameter valuelength - the number of bytes in the stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### setCharacterStream

public void **setCharacterStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader,  
 int length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Reader object, which is the given number of characters long. When a very large UNICODE value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.Reader object. The data will be read from the stream as needed until end-of-file is reached. The JDBC driver will do any necessary conversion from UNICODE to the database char format.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Parameters:**parameterName - the name of the parameterreader - the java.io.Reader object that contains the UNICODE data used as the designated parameterlength - the number of characters in the stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### setAsciiStream

public void **setAsciiStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given input stream. When a very large ASCII value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream. Data will be read from the stream as needed until end-of-file is reached. The JDBC driver will do any necessary conversion from ASCII to the database char format.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setAsciiStream which takes a length parameter.

**Parameters:**parameterName - the name of the parameterx - the Java input stream that contains the ASCII parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setBinaryStream

public void **setBinaryStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given input stream. When a very large binary value is input to a LONGVARBINARY parameter, it may be more practical to send it via a java.io.InputStream object. The data will be read from the stream as needed until end-of-file is reached.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setBinaryStream which takes a length parameter.

**Parameters:**parameterName - the name of the parameterx - the java input stream which contains the binary parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setCharacterStream

public void **setCharacterStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given Reader object. When a very large UNICODE value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.Reader object. The data will be read from the stream as needed until end-of-file is reached. The JDBC driver will do any necessary conversion from UNICODE to the database char format.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setCharacterStream which takes a length parameter.

**Parameters:**parameterName - the name of the parameterreader - the java.io.Reader object that contains the Unicode data **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setNCharacterStream

public void **setNCharacterStream**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter in this RowSet object's command to a Reader object. The Reader reads the data till end-of-file is reached. The driver does the necessary conversion from Java character format to the national character set in the database.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setNCharacterStream which takes a length parameter.

**Parameters:**parameterIndex - of the first parameter is 1, the second is 2, ...value - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur ; if a database access error occurs; or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setObject

public void **setObject**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Object](http://docs.google.com/java/lang/Object.html) x,  
 int targetSqlType,  
 int scale)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the value of the designated parameter with the given object. The second argument must be an object type; for integral values, the java.lang equivalent objects should be used.

The given Java object will be converted to the given targetSqlType before being sent to the database. If the object has a custom mapping (is of a class implementing the interface SQLData), the JDBC driver should call the method SQLData.writeSQL to write it to the SQL data stream. If, on the other hand, the object is of a class implementing Ref, Blob, Clob, NClob, Struct, java.net.URL, or Array, the driver should pass it to the database as a value of the corresponding SQL type.

Note that this method may be used to pass datatabase- specific abstract data types.

**Parameters:**parameterName - the name of the parameterx - the object containing the input parameter valuetargetSqlType - the SQL type (as defined in java.sql.Types) to be sent to the database. The scale argument may further qualify this type.scale - for java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types, this is the number of digits after the decimal point. For all other types, this value will be ignored. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if targetSqlType is a ARRAY, BLOB, CLOB, DATALINK, JAVA\_OBJECT, NCHAR, NCLOB, NVARCHAR, LONGNVARCHAR, REF, ROWID, SQLXML or STRUCT data type and the JDBC driver does not support this data type**Since:** 1.4 **See Also:**[Types](http://docs.google.com/java/sql/Types.html), #getObject

### setObject

public void **setObject**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Object](http://docs.google.com/java/lang/Object.html) x,  
 int targetSqlType)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the value of the designated parameter with the given object. This method is like the method setObject above, except that it assumes a scale of zero.

**Parameters:**parameterName - the name of the parameterx - the object containing the input parameter valuetargetSqlType - the SQL type (as defined in java.sql.Types) to be sent to the database **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if targetSqlType is a ARRAY, BLOB, CLOB, DATALINK, JAVA\_OBJECT, NCHAR, NCLOB, NVARCHAR, LONGNVARCHAR, REF, ROWID, SQLXML or STRUCT data type and the JDBC driver does not support this data type**Since:** 1.4 **See Also:**#getObject

### setObject

public void **setObject**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Object](http://docs.google.com/java/lang/Object.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the value of the designated parameter with the given object. The second parameter must be of type Object; therefore, the java.lang equivalent objects should be used for built-in types.

The JDBC specification specifies a standard mapping from Java Object types to SQL types. The given argument will be converted to the corresponding SQL type before being sent to the database.

Note that this method may be used to pass datatabase- specific abstract data types, by using a driver-specific Java type. If the object is of a class implementing the interface SQLData, the JDBC driver should call the method SQLData.writeSQL to write it to the SQL data stream. If, on the other hand, the object is of a class implementing Ref, Blob, Clob, NClob, Struct, java.net.URL, or Array, the driver should pass it to the database as a value of the corresponding SQL type.

This method throws an exception if there is an ambiguity, for example, if the object is of a class implementing more than one of the interfaces named above.

**Parameters:**parameterName - the name of the parameterx - the object containing the input parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed CallableStatement or if the given Object parameter is ambiguous [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getObject

### setBlob

public void **setBlob**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a InputStream object. The inputstream must contain the number of characters specified by length otherwise a SQLException will be generated when the PreparedStatement is executed. This method differs from the setBinaryStream (int, InputStream, int) method because it informs the driver that the parameter value should be sent to the server as a BLOB. When the setBinaryStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGVARBINARY or a BLOB

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...inputStream - An object that contains the data to set the parameter value to.length - the number of bytes in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed PreparedStatement, if parameterIndex does not correspond to a parameter marker in the SQL statement, if the length specified is less than zero or if the number of bytes in the inputstream does not match the specfied length. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setBlob

public void **setBlob**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a InputStream object. This method differs from the setBinaryStream (int, InputStream) method because it informs the driver that the parameter value should be sent to the server as a BLOB. When the setBinaryStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGVARBINARY or a BLOB

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setBlob which takes a length parameter.

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...inputStream - An object that contains the data to set the parameter value to. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed PreparedStatement or if parameterIndex does not correspond to a parameter marker in the SQL statement, [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setBlob

public void **setBlob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a InputStream object. The inputstream must contain the number of characters specified by length, otherwise a SQLException will be generated when the CallableStatement is executed. This method differs from the setBinaryStream (int, InputStream, int) method because it informs the driver that the parameter value should be sent to the server as a BLOB. When the setBinaryStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGVARBINARY or a BLOB

**Parameters:**parameterName - the name of the parameter to be set the second is 2, ...inputStream - An object that contains the data to set the parameter value to.length - the number of bytes in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement, or if the length specified is less than zero; if the number of bytes in the inputstream does not match the specfied length; if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setBlob

public void **setBlob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Blob](http://docs.google.com/java/sql/Blob.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Blob object. The driver converts this to an SQL BLOB value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - a Blob object that maps an SQL BLOB value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setBlob

public void **setBlob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) inputStream)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a InputStream object. This method differs from the setBinaryStream (int, InputStream) method because it informs the driver that the parameter value should be sent to the server as a BLOB. When the setBinaryStream method is used, the driver may have to do extra work to determine whether the parameter data should be send to the server as a LONGVARBINARY or a BLOB

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setBlob which takes a length parameter.

**Parameters:**parameterName - the name of the parameterinputStream - An object that contains the data to set the parameter value to. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setClob

public void **setClob**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The reader must contain the number of characters specified by length otherwise a SQLException will be generated when the PreparedStatement is executed. This method differs from the setCharacterStream (int, Reader, int) method because it informs the driver that the parameter value should be sent to the server as a CLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGVARCHAR or a CLOB

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...reader - An object that contains the data to set the parameter value to.length - the number of characters in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed PreparedStatement, if parameterIndex does not correspond to a parameter marker in the SQL statement, or if the length specified is less than zero. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setClob

public void **setClob**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. This method differs from the setCharacterStream (int, Reader) method because it informs the driver that the parameter value should be sent to the server as a CLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGVARCHAR or a CLOB

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setClob which takes a length parameter.

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...reader - An object that contains the data to set the parameter value to. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed PreparedStatementor if parameterIndex does not correspond to a parameter marker in the SQL statement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setClob

public void **setClob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The reader must contain the number of characters specified by length otherwise a SQLException will be generated when the CallableStatement is executed. This method differs from the setCharacterStream (int, Reader, int) method because it informs the driver that the parameter value should be sent to the server as a CLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be send to the server as a LONGVARCHAR or a CLOB

**Parameters:**parameterName - the name of the parameter to be setreader - An object that contains the data to set the parameter value to.length - the number of characters in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if the length specified is less than zero; a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setClob

public void **setClob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Clob](http://docs.google.com/java/sql/Clob.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Clob object. The driver converts this to an SQL CLOB value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - a Clob object that maps an SQL CLOB value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setClob

public void **setClob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. This method differs from the setCharacterStream (int, Reader) method because it informs the driver that the parameter value should be sent to the server as a CLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be send to the server as a LONGVARCHAR or a CLOB

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setClob which takes a length parameter.

**Parameters:**parameterName - the name of the parameterreader - An object that contains the data to set the parameter value to. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setDate

public void **setDate**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Date](http://docs.google.com/java/sql/Date.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Date value using the default time zone of the virtual machine that is running the application. The driver converts this to an SQL DATE value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getDate

### setDate

public void **setDate**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Date](http://docs.google.com/java/sql/Date.html) x,  
 [Calendar](http://docs.google.com/java/util/Calendar.html) cal)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Date value, using the given Calendar object. The driver uses the Calendar object to construct an SQL DATE value, which the driver then sends to the database. With a a Calendar object, the driver can calculate the date taking into account a custom timezone. If no Calendar object is specified, the driver uses the default timezone, which is that of the virtual machine running the application.

**Parameters:**parameterName - the name of the parameterx - the parameter valuecal - the Calendar object the driver will use to construct the date **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getDate

### setTime

public void **setTime**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Time](http://docs.google.com/java/sql/Time.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Time value. The driver converts this to an SQL TIME value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getTime

### setTime

public void **setTime**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Time](http://docs.google.com/java/sql/Time.html) x,  
 [Calendar](http://docs.google.com/java/util/Calendar.html) cal)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Time value, using the given Calendar object. The driver uses the Calendar object to construct an SQL TIME value, which the driver then sends to the database. With a a Calendar object, the driver can calculate the time taking into account a custom timezone. If no Calendar object is specified, the driver uses the default timezone, which is that of the virtual machine running the application.

**Parameters:**parameterName - the name of the parameterx - the parameter valuecal - the Calendar object the driver will use to construct the time **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getTime

### setTimestamp

public void **setTimestamp**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Timestamp](http://docs.google.com/java/sql/Timestamp.html) x,  
 [Calendar](http://docs.google.com/java/util/Calendar.html) cal)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.Timestamp value, using the given Calendar object. The driver uses the Calendar object to construct an SQL TIMESTAMP value, which the driver then sends to the database. With a a Calendar object, the driver can calculate the timestamp taking into account a custom timezone. If no Calendar object is specified, the driver uses the default timezone, which is that of the virtual machine running the application.

**Parameters:**parameterName - the name of the parameterx - the parameter valuecal - the Calendar object the driver will use to construct the timestamp **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**#getTimestamp

### setSQLXML

public void **setSQLXML**(int parameterIndex,  
 [SQLXML](http://docs.google.com/java/sql/SQLXML.html) xmlObject)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.SQLXML object. The driver converts this to an SQL XML value when it sends it to the database.

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...xmlObject - a SQLXML object that maps an SQL XML value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed result set, the java.xml.transform.Result, Writer or OutputStream has not been closed for the SQLXML object or if there is an error processing the XML value. The getCause method of the exception may provide a more detailed exception, for example, if the stream does not contain valid XML.**Since:** 1.6

### setSQLXML

public void **setSQLXML**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [SQLXML](http://docs.google.com/java/sql/SQLXML.html) xmlObject)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.SQLXML object. The driver converts this to an SQL XML value when it sends it to the database.

**Parameters:**parameterName - the name of the parameterxmlObject - a SQLXML object that maps an SQL XML value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed result set, the java.xml.transform.Result, Writer or OutputStream has not been closed for the SQLXML object or if there is an error processing the XML value. The getCause method of the exception may provide a more detailed exception, for example, if the stream does not contain valid XML.**Since:** 1.6

### setRowId

public void **setRowId**(int parameterIndex,  
 [RowId](http://docs.google.com/java/sql/RowId.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.RowId object. The driver converts this to a SQL ROWID value when it sends it to the database

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs**Since:** 1.6

### setRowId

public void **setRowId**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [RowId](http://docs.google.com/java/sql/RowId.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.sql.RowId object. The driver converts this to a SQL ROWID when it sends it to the database.

**Parameters:**parameterName - the name of the parameterx - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs**Since:** 1.6

### setNString

public void **setNString**(int parameterIndex,  
 [String](http://docs.google.com/java/lang/String.html) value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated paramter to the given String object. The driver converts this to a SQL NCHAR or NVARCHAR or LONGNVARCHAR value (depending on the argument's size relative to the driver's limits on NVARCHAR values) when it sends it to the database.

**Parameters:**parameterIndex - of the first parameter is 1, the second is 2, ...value - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur ; or if a database access error occurs**Since:** 1.6

### setNString

public void **setNString**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [String](http://docs.google.com/java/lang/String.html) value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated paramter to the given String object. The driver converts this to a SQL NCHAR or NVARCHAR or LONGNVARCHAR

**Parameters:**parameterName - the name of the column to be setvalue - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; or if a database access error occurs**Since:** 1.6

### setNCharacterStream

public void **setNCharacterStream**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) value,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The Reader reads the data till end-of-file is reached. The driver does the necessary conversion from Java character format to the national character set in the database.

**Parameters:**parameterIndex - of the first parameter is 1, the second is 2, ...value - the parameter valuelength - the number of characters in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur ; or if a database access error occurs**Since:** 1.6

### setNCharacterStream

public void **setNCharacterStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) value,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The Reader reads the data till end-of-file is reached. The driver does the necessary conversion from Java character format to the national character set in the database.

**Parameters:**parameterName - the name of the column to be setvalue - the parameter valuelength - the number of characters in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; or if a database access error occurs**Since:** 1.6

### setNCharacterStream

public void **setNCharacterStream**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The Reader reads the data till end-of-file is reached. The driver does the necessary conversion from Java character format to the national character set in the database.

**Note:** This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setNCharacterStream which takes a length parameter.

**Parameters:**parameterName - the name of the parametervalue - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur ; if a database access error occurs; or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setNClob

public void **setNClob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [NClob](http://docs.google.com/java/sql/NClob.html) value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a java.sql.NClob object. The object implements the java.sql.NClob interface. This NClob object maps to a SQL NCLOB.

**Parameters:**parameterName - the name of the column to be setvalue - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; or if a database access error occurs**Since:** 1.6

### setNClob

public void **setNClob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The reader must contain \* the number of characters specified by length otherwise a SQLException will be generated when the CallableStatement is executed. This method differs from the setCharacterStream (int, Reader, int) method because it informs the driver that the parameter value should be sent to the server as a NCLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be send to the server as a LONGNVARCHAR or a NCLOB

**Parameters:**parameterName - the name of the parameter to be setreader - An object that contains the data to set the parameter value to.length - the number of characters in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if the length specified is less than zero; if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setNClob

public void **setNClob**([String](http://docs.google.com/java/lang/String.html) parameterName,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. This method differs from the setCharacterStream (int, Reader) method because it informs the driver that the parameter value should be sent to the server as a NCLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be send to the server as a LONGNVARCHAR or a NCLOB

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setNClob which takes a length parameter.

**Parameters:**parameterName - the name of the parameterreader - An object that contains the data to set the parameter value to. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; if a database access error occurs or this method is called on a closed CallableStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setNClob

public void **setNClob**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader,  
 long length)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. The reader must contain the number of characters specified by length otherwise a SQLException will be generated when the PreparedStatement is executed. This method differs from the setCharacterStream (int, Reader, int) method because it informs the driver that the parameter value should be sent to the server as a NCLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGNVARCHAR or a NCLOB

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...reader - An object that contains the data to set the parameter value to.length - the number of characters in the parameter data. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if the length specified is less than zero; if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; if a database access error occurs or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setNClob

public void **setNClob**(int parameterIndex,  
 [NClob](http://docs.google.com/java/sql/NClob.html) value)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a java.sql.NClob object. The driver converts this oa SQL NCLOB value when it sends it to the database.

**Parameters:**parameterIndex - of the first parameter is 1, the second is 2, ...value - the parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the driver does not support national character sets; if the driver can detect that a data conversion error could occur ; or if a database access error occurs**Since:** 1.6

### setNClob

public void **setNClob**(int parameterIndex,  
 [Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to a Reader object. This method differs from the setCharacterStream (int, Reader) method because it informs the driver that the parameter value should be sent to the server as a NCLOB. When the setCharacterStream method is used, the driver may have to do extra work to determine whether the parameter data should be sent to the server as a LONGNVARCHAR or a NCLOB

**Note:** Consult your JDBC driver documentation to determine if it might be more efficient to use a version of setNClob which takes a length parameter.

**Parameters:**parameterIndex - index of the first parameter is 1, the second is 2, ...reader - An object that contains the data to set the parameter value to. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if the driver does not support national character sets; if the driver can detect that a data conversion error could occur; if a database access error occurs or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setURL

public void **setURL**(int parameterIndex,  
 [URL](http://docs.google.com/java/net/URL.html) x)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the designated parameter to the given java.net.URL value. The driver converts this to an SQL DATALINK value when it sends it to the database.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - the java.net.URL object to be set **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed PreparedStatement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BaseRowSet.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PREV CLASS   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/CachedRowSet.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/BaseRowSet.html)    [**NO FRAMES**](http://docs.google.com/BaseRowSet.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#1ksv4uv) | [METHOD](#2jxsxqh) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).