| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PreparedStatement.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**](http://docs.google.com/java/sql/ParameterMetaData.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/Ref.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/PreparedStatement.html)    [**NO FRAMES**](http://docs.google.com/PreparedStatement.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#4d34og8) |

## **java.sql**

Interface PreparedStatement

**All Superinterfaces:** [Statement](http://docs.google.com/java/sql/Statement.html), [Wrapper](http://docs.google.com/java/sql/Wrapper.html) **All Known Subinterfaces:** [CallableStatement](http://docs.google.com/java/sql/CallableStatement.html)

public interface **PreparedStatement**extends [Statement](http://docs.google.com/java/sql/Statement.html)

An object that represents a precompiled SQL statement.

A SQL statement is precompiled and stored in a PreparedStatement object. This object can then be used to efficiently execute this statement multiple times.

**Note:** The setter methods (setShort, setString, and so on) for setting IN parameter values must specify types that are compatible with the defined SQL type of the input parameter. For instance, if the IN parameter has SQL type INTEGER, then the method setInt should be used.

If arbitrary parameter type conversions are required, the method setObject should be used with a target SQL type.

In the following example of setting a parameter, con represents an active connection:

PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEES  
 SET SALARY = ? WHERE ID = ?");  
 pstmt.setBigDecimal(1, 153833.00)  
 pstmt.setInt(2, 110592)

**See Also:**[Connection.prepareStatement(java.lang.String)](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String)), [ResultSet](http://docs.google.com/java/sql/ResultSet.html)

| **Field Summary** | |
| --- | --- |

| **Fields inherited from interface java.sql.**[**Statement**](http://docs.google.com/java/sql/Statement.html) |
| --- |
| [CLOSE\_ALL\_RESULTS](http://docs.google.com/java/sql/Statement.html#CLOSE_ALL_RESULTS), [CLOSE\_CURRENT\_RESULT](http://docs.google.com/java/sql/Statement.html#CLOSE_CURRENT_RESULT), [EXECUTE\_FAILED](http://docs.google.com/java/sql/Statement.html#EXECUTE_FAILED), [KEEP\_CURRENT\_RESULT](http://docs.google.com/java/sql/Statement.html#KEEP_CURRENT_RESULT), [NO\_GENERATED\_KEYS](http://docs.google.com/java/sql/Statement.html#NO_GENERATED_KEYS), [RETURN\_GENERATED\_KEYS](http://docs.google.com/java/sql/Statement.html#RETURN_GENERATED_KEYS), [SUCCESS\_NO\_INFO](http://docs.google.com/java/sql/Statement.html#SUCCESS_NO_INFO) |

| **Method Summary** | |
| --- | --- |
| void | [**addBatch**](http://docs.google.com/java/sql/PreparedStatement.html#addBatch())()            Adds a set of parameters to this PreparedStatement object's batch of commands. |
| void | [**clearParameters**](http://docs.google.com/java/sql/PreparedStatement.html#clearParameters())()            Clears the current parameter values immediately. |
| boolean | [**execute**](http://docs.google.com/java/sql/PreparedStatement.html#execute())()            Executes the SQL statement in this PreparedStatement object, which may be any kind of SQL statement. |
| [ResultSet](http://docs.google.com/java/sql/ResultSet.html) | [**executeQuery**](http://docs.google.com/java/sql/PreparedStatement.html#executeQuery())()            Executes the SQL query in this PreparedStatement object and returns the ResultSet object generated by the query. |
| int | [**executeUpdate**](http://docs.google.com/java/sql/PreparedStatement.html#executeUpdate())()            Executes the SQL statement in this PreparedStatement object, which must be an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement. |
| [ResultSetMetaData](http://docs.google.com/java/sql/ResultSetMetaData.html) | [**getMetaData**](http://docs.google.com/java/sql/PreparedStatement.html#getMetaData())()            Retrieves a ResultSetMetaData object that contains information about the columns of the ResultSet object that will be returned when this PreparedStatement object is executed. |
| [ParameterMetaData](http://docs.google.com/java/sql/ParameterMetaData.html) | [**getParameterMetaData**](http://docs.google.com/java/sql/PreparedStatement.html#getParameterMetaData())()            Retrieves the number, types and properties of this PreparedStatement object's parameters. |
| void | [**setArray**](http://docs.google.com/java/sql/PreparedStatement.html#setArray(int,%20java.sql.Array))(int parameterIndex, [Array](http://docs.google.com/java/sql/Array.html) x)            Sets the designated parameter to the given java.sql.Array object. |
| void | [**setAsciiStream**](http://docs.google.com/java/sql/PreparedStatement.html#setAsciiStream(int,%20java.io.InputStream))(int parameterIndex, [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/java/sql/PreparedStatement.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 input stream, which will have the specified number of bytes. |
| void | [**setAsciiStream**](http://docs.google.com/java/sql/PreparedStatement.html#setAsciiStream(int,%20java.io.InputStream,%20long))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x, long length)            Sets the designated parameter to the given input stream, which will have the specified number of bytes. |
| void | [**setBigDecimal**](http://docs.google.com/java/sql/PreparedStatement.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.math.BigDecimal value. |
| void | [**setBinaryStream**](http://docs.google.com/java/sql/PreparedStatement.html#setBinaryStream(int,%20java.io.InputStream))(int parameterIndex, [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/java/sql/PreparedStatement.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 input stream, which will have the specified number of bytes. |
| void | [**setBinaryStream**](http://docs.google.com/java/sql/PreparedStatement.html#setBinaryStream(int,%20java.io.InputStream,%20long))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x, long length)            Sets the designated parameter to the given input stream, which will have the specified number of bytes. |
| void | [**setBlob**](http://docs.google.com/java/sql/PreparedStatement.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 java.sql.Blob object. |
| void | [**setBlob**](http://docs.google.com/java/sql/PreparedStatement.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/java/sql/PreparedStatement.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 | [**setBoolean**](http://docs.google.com/java/sql/PreparedStatement.html#setBoolean(int,%20boolean))(int parameterIndex, boolean x)            Sets the designated parameter to the given Java boolean value. |
| void | [**setByte**](http://docs.google.com/java/sql/PreparedStatement.html#setByte(int,%20byte))(int parameterIndex, byte x)            Sets the designated parameter to the given Java byte value. |
| void | [**setBytes**](http://docs.google.com/java/sql/PreparedStatement.html#setBytes(int,%20byte%5B%5D))(int parameterIndex, byte[] x)            Sets the designated parameter to the given Java array of bytes. |
| void | [**setCharacterStream**](http://docs.google.com/java/sql/PreparedStatement.html#setCharacterStream(int,%20java.io.Reader))(int parameterIndex, [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/java/sql/PreparedStatement.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 Reader object, which is the given number of characters long. |
| void | [**setCharacterStream**](http://docs.google.com/java/sql/PreparedStatement.html#setCharacterStream(int,%20java.io.Reader,%20long))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) reader, long length)            Sets the designated parameter to the given Reader object, which is the given number of characters long. |
| void | [**setClob**](http://docs.google.com/java/sql/PreparedStatement.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 java.sql.Clob object. |
| void | [**setClob**](http://docs.google.com/java/sql/PreparedStatement.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/java/sql/PreparedStatement.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 | [**setDate**](http://docs.google.com/java/sql/PreparedStatement.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 using the default time zone of the virtual machine that is running the application. |
| void | [**setDate**](http://docs.google.com/java/sql/PreparedStatement.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 value, using the given Calendar object. |
| void | [**setDouble**](http://docs.google.com/java/sql/PreparedStatement.html#setDouble(int,%20double))(int parameterIndex, double x)            Sets the designated parameter to the given Java double value. |
| void | [**setFloat**](http://docs.google.com/java/sql/PreparedStatement.html#setFloat(int,%20float))(int parameterIndex, float x)            Sets the designated parameter to the given Java float value. |
| void | [**setInt**](http://docs.google.com/java/sql/PreparedStatement.html#setInt(int,%20int))(int parameterIndex, int x)            Sets the designated parameter to the given Java int value. |
| void | [**setLong**](http://docs.google.com/java/sql/PreparedStatement.html#setLong(int,%20long))(int parameterIndex, long x)            Sets the designated parameter to the given Java long value. |
| void | [**setNCharacterStream**](http://docs.google.com/java/sql/PreparedStatement.html#setNCharacterStream(int,%20java.io.Reader))(int parameterIndex, [Reader](http://docs.google.com/java/io/Reader.html) value)            Sets the designated parameter to a Reader object. |
| void | [**setNCharacterStream**](http://docs.google.com/java/sql/PreparedStatement.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 | [**setNClob**](http://docs.google.com/java/sql/PreparedStatement.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/java/sql/PreparedStatement.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/java/sql/PreparedStatement.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 | [**setNString**](http://docs.google.com/java/sql/PreparedStatement.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 | [**setNull**](http://docs.google.com/java/sql/PreparedStatement.html#setNull(int,%20int))(int parameterIndex, int sqlType)            Sets the designated parameter to SQL NULL. |
| void | [**setNull**](http://docs.google.com/java/sql/PreparedStatement.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 | [**setObject**](http://docs.google.com/java/sql/PreparedStatement.html#setObject(int,%20java.lang.Object))(int parameterIndex, [Object](http://docs.google.com/java/lang/Object.html) x)            Sets the value of the designated parameter using the given object. |
| void | [**setObject**](http://docs.google.com/java/sql/PreparedStatement.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. |
| void | [**setObject**](http://docs.google.com/java/sql/PreparedStatement.html#setObject(int,%20java.lang.Object,%20int,%20int))(int parameterIndex, [Object](http://docs.google.com/java/lang/Object.html) x, int targetSqlType, int scaleOrLength)            Sets the value of the designated parameter with the given object. |
| void | [**setRef**](http://docs.google.com/java/sql/PreparedStatement.html#setRef(int,%20java.sql.Ref))(int parameterIndex, [Ref](http://docs.google.com/java/sql/Ref.html) x)            Sets the designated parameter to the given REF(<structured-type>) value. |
| void | [**setRowId**](http://docs.google.com/java/sql/PreparedStatement.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 | [**setShort**](http://docs.google.com/java/sql/PreparedStatement.html#setShort(int,%20short))(int parameterIndex, short x)            Sets the designated parameter to the given Java short value. |
| void | [**setSQLXML**](http://docs.google.com/java/sql/PreparedStatement.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 | [**setString**](http://docs.google.com/java/sql/PreparedStatement.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 Java String value. |
| void | [**setTime**](http://docs.google.com/java/sql/PreparedStatement.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/java/sql/PreparedStatement.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 value, using the given Calendar object. |
| void | [**setTimestamp**](http://docs.google.com/java/sql/PreparedStatement.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/java/sql/PreparedStatement.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 value, using the given Calendar object. |
| void | [**setUnicodeStream**](http://docs.google.com/java/sql/PreparedStatement.html#setUnicodeStream(int,%20java.io.InputStream,%20int))(int parameterIndex, [InputStream](http://docs.google.com/java/io/InputStream.html) x, int length)  **Deprecated.** |
| void | [**setURL**](http://docs.google.com/java/sql/PreparedStatement.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. |

| **Methods inherited from interface java.sql.**[**Statement**](http://docs.google.com/java/sql/Statement.html) |
| --- |
| [addBatch](http://docs.google.com/java/sql/Statement.html#addBatch(java.lang.String)), [cancel](http://docs.google.com/java/sql/Statement.html#cancel()), [clearBatch](http://docs.google.com/java/sql/Statement.html#clearBatch()), [clearWarnings](http://docs.google.com/java/sql/Statement.html#clearWarnings()), [close](http://docs.google.com/java/sql/Statement.html#close()), [execute](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String)), [execute](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String,%20int)), [execute](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String,%20int%5B%5D)), [execute](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String,%20java.lang.String%5B%5D)), [executeBatch](http://docs.google.com/java/sql/Statement.html#executeBatch()), [executeQuery](http://docs.google.com/java/sql/Statement.html#executeQuery(java.lang.String)), [executeUpdate](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String)), [executeUpdate](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String,%20int)), [executeUpdate](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String,%20int%5B%5D)), [executeUpdate](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String,%20java.lang.String%5B%5D)), [getConnection](http://docs.google.com/java/sql/Statement.html#getConnection()), [getFetchDirection](http://docs.google.com/java/sql/Statement.html#getFetchDirection()), [getFetchSize](http://docs.google.com/java/sql/Statement.html#getFetchSize()), [getGeneratedKeys](http://docs.google.com/java/sql/Statement.html#getGeneratedKeys()), [getMaxFieldSize](http://docs.google.com/java/sql/Statement.html#getMaxFieldSize()), [getMaxRows](http://docs.google.com/java/sql/Statement.html#getMaxRows()), [getMoreResults](http://docs.google.com/java/sql/Statement.html#getMoreResults()), [getMoreResults](http://docs.google.com/java/sql/Statement.html#getMoreResults(int)), [getQueryTimeout](http://docs.google.com/java/sql/Statement.html#getQueryTimeout()), [getResultSet](http://docs.google.com/java/sql/Statement.html#getResultSet()), [getResultSetConcurrency](http://docs.google.com/java/sql/Statement.html#getResultSetConcurrency()), [getResultSetHoldability](http://docs.google.com/java/sql/Statement.html#getResultSetHoldability()), [getResultSetType](http://docs.google.com/java/sql/Statement.html#getResultSetType()), [getUpdateCount](http://docs.google.com/java/sql/Statement.html#getUpdateCount()), [getWarnings](http://docs.google.com/java/sql/Statement.html#getWarnings()), [isClosed](http://docs.google.com/java/sql/Statement.html#isClosed()), [isPoolable](http://docs.google.com/java/sql/Statement.html#isPoolable()), [setCursorName](http://docs.google.com/java/sql/Statement.html#setCursorName(java.lang.String)), [setEscapeProcessing](http://docs.google.com/java/sql/Statement.html#setEscapeProcessing(boolean)), [setFetchDirection](http://docs.google.com/java/sql/Statement.html#setFetchDirection(int)), [setFetchSize](http://docs.google.com/java/sql/Statement.html#setFetchSize(int)), [setMaxFieldSize](http://docs.google.com/java/sql/Statement.html#setMaxFieldSize(int)), [setMaxRows](http://docs.google.com/java/sql/Statement.html#setMaxRows(int)), [setPoolable](http://docs.google.com/java/sql/Statement.html#setPoolable(boolean)), [setQueryTimeout](http://docs.google.com/java/sql/Statement.html#setQueryTimeout(int)) |

| **Methods inherited from interface java.sql.**[**Wrapper**](http://docs.google.com/java/sql/Wrapper.html) |
| --- |
| [isWrapperFor](http://docs.google.com/java/sql/Wrapper.html#isWrapperFor(java.lang.Class)), [unwrap](http://docs.google.com/java/sql/Wrapper.html#unwrap(java.lang.Class)) |

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

### executeQuery

[ResultSet](http://docs.google.com/java/sql/ResultSet.html) **executeQuery**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the SQL query in this PreparedStatement object and returns the ResultSet object generated by the query.

**Returns:**a ResultSet object that contains the data produced by the query; never null **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 the SQL statement does not return a ResultSet object

### executeUpdate

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

Executes the SQL statement in this PreparedStatement object, which must be an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement.

**Returns:**either (1) the row count for SQL Data Manipulation Language (DML) statements or (2) 0 for SQL statements that return nothing **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 the SQL statement returns a ResultSet object

### setNull

void **setNull**(int parameterIndex,  
 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:**parameterIndex - the first parameter is 1, the second is 2, ...sqlType - the SQL type code defined in java.sql.Types **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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 sqlType 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

### setBoolean

void **setBoolean**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setByte

void **setByte**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setShort

void **setShort**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setInt

void **setInt**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setLong

void **setLong**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setFloat

void **setFloat**(int parameterIndex,  
 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 REAL 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setDouble

void **setDouble**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setBigDecimal

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.math.BigDecimal value. The driver converts this to an SQL NUMERIC 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setString

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 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setBytes

void **setBytes**(int parameterIndex,  
 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setDate

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 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:**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setTime

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.

**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setTimestamp

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.

**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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setAsciiStream

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 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### setUnicodeStream

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.**

Sets the designated parameter to the given input stream, which will have the specified number of bytes. 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. 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. The byte format of the Unicode stream must be a Java UTF-8, as defined in the Java Virtual Machine Specification.

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

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - a java.io.InputStream object that contains the Unicode parameter valuelength - the number of bytes in the stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### setBinaryStream

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 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

### clearParameters

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

Clears the current parameter values immediately.

In general, parameter values remain in force for repeated use of a statement. Setting a parameter value automatically clears its previous value. However, in some cases it is useful to immediately release the resources used by the current parameter values; this can be done by calling the method clearParameters.

**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

### setObject

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. This method is like the method setObject above, except that it assumes a scale of zero.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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 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**See Also:**[Types](http://docs.google.com/java/sql/Types.html)

### setObject

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 value of the designated parameter using 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, RowId, SQLXML or Array, the driver should pass it to the database as a value of the corresponding SQL type.

**Note:** Not all databases allow for a non-typed Null to be sent to the backend. For maximum portability, the setNull or the setObject(int parameterIndex, Object x, int sqlType) method should be used instead of setObject(int parameterIndex, Object x).

**Note:** 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - the object containing the input parameter value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs; this method is called on a closed PreparedStatement or the type of the given object is ambiguous

### execute

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

Executes the SQL statement in this PreparedStatement object, which may be any kind of SQL statement. Some prepared statements return multiple results; the execute method handles these complex statements as well as the simpler form of statements handled by the methods executeQuery and executeUpdate.

The execute method returns a boolean to indicate the form of the first result. You must call either the method getResultSet or getUpdateCount to retrieve the result; you must call getMoreResults to move to any subsequent result(s).

**Returns:**true if the first result is a ResultSet object; false if the first result is an update count or there is no result **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 an argument is supplied to this method**See Also:**[Statement.execute(java.lang.String)](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String)), [Statement.getResultSet()](http://docs.google.com/java/sql/Statement.html#getResultSet()), [Statement.getUpdateCount()](http://docs.google.com/java/sql/Statement.html#getUpdateCount()), [Statement.getMoreResults()](http://docs.google.com/java/sql/Statement.html#getMoreResults())

### addBatch

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

Adds a set of parameters to this PreparedStatement object's batch of commands.

**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**Since:** 1.2 **See Also:**[Statement.addBatch(java.lang.String)](http://docs.google.com/java/sql/Statement.html#addBatch(java.lang.String))

### setCharacterStream

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 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:**parameterIndex - the first parameter is 1, the second is 2, ...reader - the java.io.Reader object that contains the Unicode datalength - the number of characters in the stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.2

### setRef

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

Sets the designated parameter to the given REF(<structured-type>) value. The driver converts this to an SQL REF value when it sends it to the database.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - an SQL REF value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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:** 1.2

### setBlob

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

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - a Blob object that maps an SQL BLOB value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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:** 1.2

### setClob

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

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - a Clob object that maps an SQL CLOB value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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:** 1.2

### setArray

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

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

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - an Array object that maps an SQL ARRAY value **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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:** 1.2

### getMetaData

[ResultSetMetaData](http://docs.google.com/java/sql/ResultSetMetaData.html) **getMetaData**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves a ResultSetMetaData object that contains information about the columns of the ResultSet object that will be returned when this PreparedStatement object is executed.

Because a PreparedStatement object is precompiled, it is possible to know about the ResultSet object that it will return without having to execute it. Consequently, it is possible to invoke the method getMetaData on a PreparedStatement object rather than waiting to execute it and then invoking the ResultSet.getMetaData method on the ResultSet object that is returned.

**NOTE:** Using this method may be expensive for some drivers due to the lack of underlying DBMS support.

**Returns:**the description of a ResultSet object's columns or null if the driver cannot return a ResultSetMetaData object **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:** 1.2

### setDate

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 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 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.2

### setTime

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 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 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.2

### setTimestamp

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 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 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.2

### setNull

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. 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:**parameterIndex - the first parameter is 1, the second is 2, ...sqlType - 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 REF **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; 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 sqlType 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 or if the JDBC driver does not support this method**Since:** 1.2

### setURL

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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### getParameterMetaData

[ParameterMetaData](http://docs.google.com/java/sql/ParameterMetaData.html) **getParameterMetaData**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the number, types and properties of this PreparedStatement object's parameters.

**Returns:**a ParameterMetaData object that contains information about the number, types and properties for each parameter marker of this PreparedStatement object **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**Since:** 1.4 **See Also:**[ParameterMetaData](http://docs.google.com/java/sql/ParameterMetaData.html)

### setRowId

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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### setNString

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 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

### setNCharacterStream

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 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

### setNClob

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 to a 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 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

### setClob

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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs; this method is called on a closed PreparedStatement 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

### setBlob

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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs; this method is called on a closed PreparedStatement; 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

### setNClob

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

### setSQLXML

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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs; this method is called on a closed PreparedStatement or the java.xml.transform.Result, Writer or OutputStream has not been closed for the SQLXML object [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.6

### setObject

void **setObject**(int parameterIndex,  
 [Object](http://docs.google.com/java/lang/Object.html) x,  
 int targetSqlType,  
 int scaleOrLength)  
 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. If the second argument is an InputStream then the stream must contain the number of bytes specified by scaleOrLength. If the second argument is a Reader then the reader must contain the number of characters specified by scaleOrLength. If these conditions are not true the driver will generate a SQLException when the prepared statement is executed.

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 database-specific abstract data types.

**Parameters:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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.scaleOrLength - for java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types, this is the number of digits after the decimal point. For Java Object types InputStream and Reader, this is the length of the data in the stream or reader. For all other types, this value will be ignored. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs; this method is called on a closed PreparedStatement or if the Java Object specified by x is an InputStream or Reader object and the value of the scale parameter is less than zero [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:** 1.6 **See Also:**[Types](http://docs.google.com/java/sql/Types.html)

### setAsciiStream

void **setAsciiStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 long 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.6

### setBinaryStream

void **setBinaryStream**(int parameterIndex,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) x,  
 long 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:**parameterIndex - the first parameter is 1, the second is 2, ...x - 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 parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.6

### setCharacterStream

void **setCharacterStream**(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 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:**parameterIndex - the first parameter is 1, the second is 2, ...reader - the java.io.Reader object that contains the Unicode datalength - the number of characters in the stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement**Since:** 1.6

### setAsciiStream

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 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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

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 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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### setCharacterStream

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 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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### setNCharacterStream

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 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 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

### setClob

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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### setBlob

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 parameterIndex does not correspond to a parameter marker in the SQL statement; 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

### setNClob

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

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PreparedStatement.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**](http://docs.google.com/java/sql/ParameterMetaData.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/Ref.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/PreparedStatement.html)    [**NO FRAMES**](http://docs.google.com/PreparedStatement.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#4d34og8) |

[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).