# Oracle® Visual Information Retrieval Java Classes

User's Guide and Reference

Release 8.1.7

September 2000

Part No. A85333-01

Visual Information Retrieval technology licensed from Virage, Inc.

ORACLE®

Oracle Visual Information Retrieval Java Classes User's Guide and Reference, Release 8.1.7

Part No. A85333-01



Primary Author: Max Chittister

Contributors: Raja Chatterje, Sue Mavris, Dan Mullen, Deb Owens, Brenda Silva, Rod Ward

# Contents

## List of Examples

# Send Us Your Comments

**Oracle Visual Information Retrieval Java Classes User's Guide and Reference, Release 8.1.7**

**Part No. A85333-01**

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc_doc@us.oracle.com
- FAX: 603.897.3316 Attn: Oracle Visual Information Retrieval Java Classes Documentation
- Postal service:
  Oracle Corporation
  Oracle Visual Information Retrieval Documentation
  One Oracle Drive
  Nashua, NH 03062-2698
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

# Preface

This guide describes how to use Oracle Visual Information Retrieval Java Classes release 8.1.7. This product requires Oracle Enterprise Edition.

## Audience

This guide is for anyone who is interested in storing, retrieving, and manipulating image data in an Oracle8*i* database using Oracle Visual Information Retrieval and a Java client interface, including developers of image specialization services. Users of this guide should have experience with Java and Java Database Connectivity (JDBC).

## Organization

This guide contains the following chapters:

| | |
|---|---|
| Chapter 1 | Contains a general introduction. |
| Chapter 2 | Contains information on the examples included with the Java Classes installation. |
| Chapter 3 | Contains reference information on the ORDVir class. |
| Appendix A | Contains information on running the sample files included with the Java Classes. |
| Appendix B | Contains information on possible exceptions and errors. |
| Appendix C | Contains information on methods that have been deprecated since the previous release. |

## Related Documents

This guide is not intended as a standalone document. It is a supplement to *Oracle Visual Information Retreival User's Guide and Reference*. This guide describes the specifics of the Java client interface for Visual Information Retrieval. You need both guides to successfully perform visual information retrieval using the Java interface to this product.

In addition, the ORDVir Java class library described in this guide contains many inherited methods that are documented in *Oracle8i interMedia Audio, Image, and Video Java Classes User's Guide and Reference.* This guide should also be considered essential to understanding and using the Java interface to this product.

For more information about using this product in a development environment, see the following documents in the Oracle8*i* documentation set:

- *Oracle Call Interface Programmer's Guide*
- *Oracle8i Application Developer's Guide - Large Objects (LOBs)*
- *Oracle8i Concepts*
- *Oracle8i interMedia Audio, Image, and Video Java Classes User's Guide and Reference*
- *PL/SQL User's Guide and Reference*

For information about basic image storage and retrieval, see *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*.

For more information on using JDBC, see *Oracle8i JDBC Developer's Guide and Reference*.

Visual Information Retrieval is based on technology licensed from Virage, Inc. Visit the Virage Web site for additional information at

`http://www.virage.com`.

## Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this guide:

| Convention | Meaning |
|---|---|
| .<br>.<br>. | Vertical ellipsis points in an example mean that information not directly related to the example has been omitted. |
| . . . | Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted. |
| **boldface text** | Boldface text indicates a term defined in the text, the glossary, or in both locations. |
| | In code examples, a boldface number surrounded by brackets (for example, **[1]**) indicates that the following code will be described in more detail in the given-numbered step in the subsequent numbered list. |
| *italic text* | Italic text is used for emphasis, for user-supplied variables, and for book titles. |
| < > | Angle brackets enclose user-supplied names. |
| [ ] | Brackets enclose optional clauses from which you can choose one or none. |

# 1
# Introduction

Oracle8*i* Visual Information Retrieval is an extension to Oracle8*i* that provides image storage, content-based retrieval, and format conversion capabilities through an object type. The capabilities of this product encompass the storage, retrieval, and manipulation of image data managed by the Oracle8*i* database server. This product supports image storage using binary large objects (BLOBs) and references to image data residing externally in binary files (BFILEs) or URLs.

Visual Information Retrieval is a building block for various imaging applications, rather than being an end-user application in itself. It consists of an object type (ORDVir) along with related methods for managing and processing image data.

Refer to *Oracle Visual Information Retreival User's Guide and Reference* for information about how visual information retrieval works.

## 1.1 Relationship with Oracle *inter*Media Image

You may already be familiar with Oracle *inter*Media Image, either as a component of Oracle *inter*Media or in a previous release as Oracle8 Image Cartridge. The base *inter*Media Image option and Visual Information Retrieval both let you store an image as an object in the database or as a reference to an external file or URL. Both products let you store and query on the following attributes:

- Image height
- Image width
- Image size
- File type or format (such as TIFF)
- Compression type or format (such as JPEG)
- Image type (such as monochrome)

- MIME type

Understanding the base Image option is important because the Visual Information Retrieval object type is defined as the Image object plus a signature attribute.

The main differences between the Image option and the Visual Information Retrieval product are that Visual Information Retrieval lets you create and use indexes, and perform **content-based retrieval**. Content-based retrieval lets you perform queries based on intrinsic visual attributes of the image (color, structure, texture), rather than being limited to keyword searches in textual annotations or descriptions. The underlying technology was developed by Virage, Inc., a leader in content-based retrieval.

## 1.2 Java Application Support

Oracle Visual Information Retrieval Java Classes lets you write your own Java applications to use, retrieve, and modify multimedia data stored in an Oracle database.

Oracle Visual Information Retrieval Java Classes lets you connect to a database through JDBC calls, select a database object into a Java application object, perform various operations on the application object, and commit your changes to the database object.

## 1.3 Interaction Between the Database and Java Application

Perform the following operations to make a connection between a database object and an application object:

1. Make a connection from the application to the database through JDBC calls.

   Write a method that returns a valid OracleConnection object; see Example 2–3 for an example of a method that makes a connection to the database.

2. Execute a SELECT statement on the database table and store the results in your application.

   To execute the SELECT statement, you must create a Statement object in your application, use the executeQuery( ) method to execute the SELECT statement, and cast the results into an OracleResultSet object. See steps 1 and 2 of Example 2–4 for an example.

3. Move the results into an ORDVir object with the getCustomDatum( ) method.

Create an application ORDVir object, and instantiate it with the results of the getCustomDatum( ) method. See step 5 of Example 2–4 for an example.

You should now have an application ORDVir object that is identical to the database object.

4. Perform operations on the application object.
5. Update the database object to include the results of the operations in step 4.

   Create an OraclePreparedStatement object that contains a SQL statement that updates the database object, and execute the statement. See step 8 of Example 2–4 for an example.
6. Commit your changes.

   Unless you used the setAutoCommit( ) method to true in your connection method, you must perform an explicit commit to update the database object with any changes that you made to the application object. You do this with the JDBC commit( ) method. See step 4 of Example 2–2 for an example of the commit( ) method.
7. Close the connection.

   Close the connection between the application and database with the JDBC close( ) method. See step 5 of Example 2–2 for an example of the close( ) method.

   For more information about using JDBC, see *Oracle8i JDBC Developer's Guide and Reference*.

## 1.4 Compatibility with Previous Versions of Visual Information Retrieval

Oracle Corporation may evolve the Visual Information Retrieval object type by adding new object attributes in a future release. If you want client-side applications to maintain compatibility to release 8.1.7 of the ORDVir object type, even after a server upgrade that evolves the object types, you are advised to make a call to the compatibility initialization function at the beginning of the application.

**Note:** If you do not follow the recommended actions, you may have to upgrade, and perhaps even recompile, your application when you upgrade to a newer release of the server that evolves the ORDVir types.

Client-side applications written in Java using the Visual Information Retrieval Java Classes for release 8.1.7 should call the OrdMediaUtil.virCompatibilityInit( ) function after connecting to Oracle.

```
public static void virCompatibilityInit(OracleConnection con)
    throws Exception
```

This Java function takes an OracleConnection object as an argument. The included Visual Information Retrieval 8.1.7 Java API will ensure that your 8.1.7 application will work (without upgrading) with a potential future version of Visual Information Retrieval with an evolved object type.

See Example 2–2 for an example of the virCompatibilityInit( ) method.

# 2

# Program Examples Using Java Classes

This chapter provides a full-length example of user-defined classes using Visual Information Retrieval Java Classes. A sample SQL script that demonstrates how to set up a schema on your database server is also included.

This code will not necessarily match the code shipped as VirExample.java with the Java Classes installation. If you want to run an example on your system, use the files provided with the Java Classes installation; do not attempt to compile and run the code presented in this chapter.

> **Note:** This chapter contains examples of Java and SQL code. Some of the code examples display boldface numbers enclosed in brackets; these indicate that further explanation of that code will be in the numbered list immediately following the example.

The Visual Information Retrieval example contains user-defined methods that use SQL, JDBC, and Visual Information Retrieval Java Classes APIs to perform the following operations:

- Create a database server table that contains test content.
- Load data into both application and database ORDVir objects from a local file, and set the local field on both the application and database objects.
- Load data into both application and database ORDVir objects from a local stream, and set the local field on both the application and database objects.
- Load data into both application and database ORDVir objects from a local byte array, and set the local field on both the application and database objects.
- Extract and print properties from the application ORDVir object.

- Show an example of the process( ) and processCopy( ) methods.
- Show an example of the similar( ) method.
- Show an example of the score( ) method.

## 2.1 VirExample.sql

Example 2–1 shows the contents of VirExample.sql.

***Example 2–1 Contents of VirExample.sql***

```
set echo on

-- Please change system password.
connect system/manager
drop user VIRUSER cascade;

[1] grant connect,resource to VIRUSER identified by VIRUSER;

-- Replace 'C:\Oracle\Ora81' with your Oracle home directory
[2] create or replace directory ORDVIRDIR as 'C:\Oracle\Ora81\ord\vir\demo';
grant read on directory ORDVIRDIR to public with grant option;

[3] connect VIRUSER/VIRUSER;

[4] create table ordvirtab(id number, image ORDSYS.ORDVir, image2
ORDSYS.ORDVir);

-- Note - the ORDVir.init method was added in 8.1.7.  If you are running
-- against an older release of Visual Information Retrieval and Oracle,
-- you will have to modify the following INSERT statements to use the
-- OrdVir default constructor.
--
[5] insert into ordvirtab values
(1, ORDSYS.ORDVir.init( ),
    ORDSYS.ORDVir.init( ));

insert into ordvirtab values
(2, ORDSYS.ORDVir.init( ),
    ORDSYS.ORDVir.init( ));

insert into ordvirtab values
(3, ORDSYS.ORDVir.init( ),
    ORDSYS.ORDVir.init( ));
```

```
insert into ordvirtab values
(4, ORDSYS.ORDVir.init( ),
    ORDSYS.ORDVir.init( ));

[5] insert into ordvirtab values
(5, ORDSYS.ORDVir.init('file','ORDVIRDIR','stripes1.bmp'),
    ORDSYS.ORDVir.init('file','ORDVIRDIR','stripes2.bmp'));

insert into ordvirtab values
(6, ORDSYS.ORDVir.init('file','ORDVIRDIR','lcolor1.bmp'),
    ORDSYS.ORDVir.init('file','ORDVIRDIR','lcolor2.bmp'));

commit;

set echo off

exit;
```

The SQL statements in VirExample.sql perform the following operations:

1. Create a user named VIRUSER and grant the appropriate permissions to the user.
2. Create a directory named ORDVIRDIR and set the appropriate permissions. If necessary, edit the name of the directory to match your file system.
3. Connect to the database as VIRUSER.
4. Create a table named ORDVIRTAB, which contains a column of numbers and two columns of ORDVir objects.
5. Using the init method, add four rows with two empty objects in each row.
6. Using the init method, add two rows with two objects in each row with values.

The ORDVir.init method was added in release 8.1.7. If you are running against a previous release of Visual Information Retrieval and Oracle8*i*, you will have to modify the INSERT statements in steps 5 and 6 to use the ORDVir default constructor.

See *Oracle8i Visual Information Retrieval User's Guide and Reference* for more information on the init method.

## 2.2 VirExample.java

Section 2.2.1 through Section 2.2.9 show the methods contained in the VirExample.java sample file.

### 2.2.1 main( ) Method

Example 2–2 shows the main( ) method.

***Example 2–2 main( ) Method***

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try{
          VirExample ve = new VirExample( );
          [1] con = ve.connect( );
          //Only include the following line if you are running
          //an Oracle 8.1.7 database or later.
          //If you are running a database server prior to 8.1.7,
          //the call will fail.
          [2] OrdMediaUtil.virCompatibilityInit(con);
          [3] ve.setPropertiesExample(con);
          ve.displayPropertiesExample(con);
          ve.fileBasedExample(con);
          ve.streamBasedExample(con);
          ve.byteArrayBasedExample(con);
          ve.processExample(con);
          ve.similarExample(con);
          ve.scoreExample(con);
          [4] con.commit( );
          [5] con.close( );
          System.out.println("Done.");
     }
     [6] catch (Exception e){
          try{
               System.out.println("Exception : " + e);
               con.close( );
          }
          catch(Exception ex){
               System.out.println("Close Connection Exception : " + ex);
          }
     }
}
```

The code in the main( ) method performs the following operations:

1. Uses the connect( ) method to make a connection to a database table.
2. Ensures the compatibility of your 8.1.7 application; this will not work if your database server is 8.1.6 or earlier. See Section 1.4 for more information.
3. Calls several methods (also defined in VirExample.java) that manipulate objects on the database server and the local machine.
4. Commits any changes made to the database table.
5. Closes the connection to the database.
6. Handles any errors or exceptions raised by the code.

Section 2.2.2 through Section 2.2.9 will provide information on the methods called from the main( ) method.

## 2.2.2 connect( ) Method

Example 2–3 shows a user-defined method named connect( ), which makes a connection from the application to the database.

***Example 2–3 connect( ) Method***

```
public OracleConnection connect( ) throws Exception{
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)DriverManager.getConnection
          (connectString,"VIRUSER","VIRUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

The connect( ) method performs the following operations:

1. Loads the JDBC drivers directly, because Oracle uses a JDK-compliant Java virtual machine.
2. Defines a string that contains the URL of the database to which you will connect. You may need to change this string to match your database.
3. Sets the connection to the database, using the URL contained in connectString, the user name VIRUSER, and the password VIRUSER. The user name and password were created by VirExample.sql.

4. Disables auto-commit mode. This means that you must commit or roll back manually with the commit( ) or rollback( ) methods, respectively.

## 2.2.3 setPropertiesExample( ) Method

Example 2–4 shows a user-defined method named setPropertiesExample( ) that sets the properties in the application object.

***Example 2–4 setPropertiesExample( ) Method***

```
public void setPropertiesExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from ordvirtab where id = 5 for update");
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdVir imgObj = (OrdVir)rs.getCustomDatum
                    (2, OrdVir.getFactory( ));
               [6] imgObj.setProperties( );
               System.out.println("set Properties called");
               [7] if(imgObj.checkProperties( )){
                    System.out.println("checkProperties called");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("successful");
               }
               else{
                    System.out.println("checkProperties called");
                    System.out.println("setProperties not successful");
                    System.out.println("checkProperties successful");
               }
               [8] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordvirtab set
                    image = ? where id = " + index);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs.close( );
          s.close( );
     }
     [9] catch(Exception e){
          System.out.println("exception raised " + e);
```

```
    }
}
```

The setPropertiesExample( ) method performs the following operations:

1. Creates an OracleStatement object.
2. Executes the given SQL query and casts the results into a local OracleResultSet object.
3. Performs the operations in the loop while there are results in the OracleResultSet that have not been processed. However, in this case, there is only one row included in the OracleResultSet, so the operations in the loop will run once.
4. Sets an index variable to the value of the integer in the first column of the first row in the OracleResultSet (in this case, the value is 5).
5. Creates a local ORDVir object named imgObj. Populates imgObj with the contents of the ORDVir object in the second column of the current row in the OracleResultSet.
6. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDVir object. See "setProperties( )" for a list of the properties values extracted and set.
7. Calls checkProperties( ) to compare the properties values in the application object attributes with the values in the media data. If all values are the same, checkProperties( ) returns TRUE and the appropriate messages are printed to the screen. If any values differ, checkProperties( ) returns FALSE and the appropriate messages are printed to the screen.
8. Creates and executes a SQL statement that will update the database ORDVir object with the contents of imgObj.
9. Handles any errors or exceptions raised by the code.

## 2.2.4 displayPropertiesExample( ) Method

Example 2–5 shows a user-defined method named displayPropertiesExample( ) that prints the attributes of the application object to the screen.

***Example 2–5 displayPropertiesExample( ) Method***

```
public void displayPropertiesExample(OracleConnection con){
    try{
```

```
            int index = 0;
            [1] Statement s = con.createStatement( );
            OracleResultSet rs = (OracleResultSet)s.executeQuery(
                 "select * from ordvirtab where id = 5 ");
            while(rs.next( )){
                 index = rs.getInt(1);
                 OrdVir imgObj = (OrdVir) rs.getCustomDatum(2,
                      OrdVir.getFactory( ));
                 [2] System.out.println("format : " + imgObj.getFormat( ));
                 System.out.println("mimeType: " + imgObj.getMimeType( ));
                 System.out.println("height: " + imgObj.getHeight( ));
                 System.out.println("width: " + imgObj.getWidth( ));
                 System.out.println("contentLength: " +
                      imgObj.getContentLength( ));
                 System.out.println("contentFormat: " +
                      imgObj.getContentFormat( ));
                 System.out.println("compressionFormat: " +
                      imgObj.getCompressionFormat( ));
                 System.out.println("source type: " +
                      imgObj.getSourceType( ));
                 System.out.println("source loc: " +
                      imgObj.getSourceLocation( ));
                 System.out.println("source name: " + imgObj.getSourceName( ));
                 System.out.println("source : " + imgObj.getSource( ));
                 [3] try{
                      String attrString = getAllAttributesAsString(imgObj);
                      System.out.println(attrString);
                 }
                 [4] catch (Exception e){
                      System.out.println("Exception raised in
                           getAllAttributesAsString:");
                 }
                 System.out.println("successful");
            }
       }
       [5] catch(Exception e) {
            System.out.println("exception raised " + e);
       }
}
```

The displayPropertiesExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be

operating on the contents of the fifth row of the database table. This is the same row you operated on in Example 2–4.

2. Gets the values of the properties in imgObj and prints them to the screen.
3. Gets the attributes of imgObj and stores them in a string by using the getAllAttributesAsString( ) method, and prints the contents of the string to the screen. See Section 2.2.5 for more information on getAllAttributesAsString( ).
4. Handles any errors or exceptions raised by the call to getAllAttributesAsString( ).
5. Handles any errors or exceptions raised by the code in general.

## 2.2.5 getAllAttributesAsString( ) Method

Example 2–6 shows a user-defined method named getAllAttributesAsString( ) that creates a String object that contains the values of the application object attributes.

***Example 2–6 getAllAttributesAsString( ) Method***

```
public String getAllAttributesAsString (OrdVir imgObj) throws Exception{
     [1] String attStr = imgObj.getSource( ) + " mimeType = " +
          imgObj.getMimeType( ) + ", fileFormat = " +
          imgObj.getFormat( ) + ", height = " + imgObj.getHeight( )
          + ", width = " + imgObj.getWidth( ) + ", contentLength = "
          + imgObj.getContentLength( ) + ", contentFormat = " +
          imgObj.getContentFormat( ) + ", compressionFormat = " +
          imgObj.getCompressionFormat( );
     [2] return attStr;
}
```

The getAllAttributesAsString( ) method performs the following operations:

1. Creates a String object named attStr. Gets the values of several attributes from the application object and stores their values in attStr.
2. Returns attStr to the method that called this method.

## 2.2.6 fileBasedExample( ) Method

Example 2–7 shows a user-defined method named fileBasedExample( ) that uses the loadDataFromFile( ) method to load media data into the application object.

***Example 2–7 fileBasedExample( ) Method***

```
public void fileBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ordvirtab where id = 2 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVir imgObj = (OrdVir) rs.getCustomDatum(2,
                    OrdVir.getFactory( ));
               [2] imgObj.loadDataFromFile("virdemo1.dat");
               [3] imgObj.setProperties( );
               [4] imgObj.getDataInFile("fileexample.dat");
               [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordvirtab set image =
                    ? where id = " + index);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          System.out.println("successful");
     }
     [6] catch(Exception e){
          System.out.println("exception raised " + e);
     }
}
```

The fileBasedExample( ) method performs the following operation:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the second row of the database table.
2. Uses the loadDataFromFile( ) method to load the media data from the local file virdemo1.dat into the database ORDVir object and into imgObj. This also sets the local field on imgObj, but not the database object.
3. Calls the setProperties( ) method to extract property values from the media data and set them in the application ORDVir object. See "setProperties( )" for a list of the property values extracted and set.

4. Uses the getDataInFile( ) method to get the media data from the application ORDVir object and loads it into a file on the local system named fileexample.dat.
5. Creates and executes a SQL statement that will update the database ORDVir object with the contents of imgObj. This update will set the attributes on the database object, to match the application object.
6. Handles any errors or exceptions raised by the code.

## 2.2.7 streamBasedExample( ) Method

Example 2–8 shows a user-defined method named streamBasedExample( ), which uses the loadDataFromInputStream( ) method to load media data into the application object.

***Example 2–8 streamBasedExample( ) Method***

```
public void streamBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ORDVIRTAB where id = 3 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVir imgObj = (OrdVir) rs.getCustomDatum(2,
                    OrdVir.getFactory( ));
               [2] FileInputStream fStream = new FileInputStream
                    ("virdemo1.dat");
               [3] imgObj.loadDataFromInputStream(fStream);
               [4] fStream.close( );
               [5] imgObj.setProperties( );
               [6] InputStream inpStream = imgObj.getDataInStream( );
               int length = 32300;
               byte[ ] tempBuffer = new byte[length];
               [7] int numRead = inpStream.read(tempBuffer,0,length);
               FileOutputStream outStream=null;
               try{
                    [8] outStream = new FileOutputStream
                         ("streamexample.dat");
                    [9] while(numRead != -1){
                         [10] if (numRead < length){
                              length = numRead;
                              outStream.write(tempBuffer,0,length);
```

```
                              break;
                         }
                         [11] else
                              outStream.write(tempBuffer,0,length);
                         [12] numRead = inpStream.read(tempBuffer,0,
                              length);
                    }
               }
               [13] finally{
                    if (outStream != null)
                         outStream.close( );
                    inpStream.close( );
               }
               [14] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordvirtab set
                    image = ? where id = " + index);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          System.out.println("successful");
     }
     [15] catch(Exception e){
          System.out.println("exception raised " + e);
     }
}
```

The streamBasedExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the third row of the database table.
2. Creates a new FileInputStream object. This input stream contains the contents of a local file named virdemo1.dat.
3. Uses the loadDataFromInputStream( ) method to load the media data in the input stream into the database ORDVir object and into imgObj. This also sets the local field on imgObj, but not the database object.
4. Closes the input stream.
5. Calls the setProperties( ) method to extract property values from the media data and sets them in the application ORDVir object. See "setProperties( )" for a list of the property values extracted and set.

6. Creates a new InputStream object named inpStream. Calls the getDataInStream( ) method to get the media data from the application ORDVir object and stores it in inpStream.
7. Reads 32300 bytes from the beginning (that is, at an offset of 0) of inpStream into the byte array tempBuffer. The integer numRead will be set to the total number of bytes read, or -1 if the end of the input stream has been reached. In this case, if loading is successful, numRead should be equal to 32300.
8. Creates a new FileOutputStream object named outStream. This output stream will write data to a local file named streamexample.dat.
9. Runs the operations in the while loop if numRead is not equal to -1. The program should enter this loop.
10. Writes the number of bytes read into tempBuffer into outStream, and then break out of the loop, if numRead is less than 32300 (that is, if not all of the data was read).
11. Writes 32300 bytes into outStream if numRead is 32300.
12. Attempts to read more data from the input stream into the byte array. If you have already successfully read 32300 bytes of data, then numRead will be set to -1, and the program will exit the loop. If there is still unread data in the input stream, then it will be read into the byte array and repeats steps 10 and 11.
13. Closes both the input stream and the output stream after exiting the while loop.
14. Creates and executes a SQL statement that will update the database ORDVir object with the contents of imgObj. This update will set the attributes on the database object to match the application object.
15. Handles any errors or exceptions raised by the code.

## 2.2.8 byteArrayBasedExample( ) Method

Example 2–9 shows a user-defined method named byteArrayBasedExample( ) that uses the loadDataFromByteArray( ) method to load media data into the application object.

***Example 2–9 byteArrayBasedExample( ) Method***

```
public void byteArrayBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
```

```
            OracleResultSet rs = (OracleResultSet)s.executeQuery
                 ("select * from ORDVIRTAB where id = 4 for update ");
            while(rs.next( )){
                 index = rs.getInt(1);
                 OrdVir imgObj = (OrdVir) rs.getCustomDatum(2,
                      OrdVir.getFactory( ));
                 [2] File ff = new File("virdemo1.dat");
                 int fileLength = (int) ff.length( );
                 byte[ ] data = new byte[fileLength];
                 [3] FileInputStream fStream = new
                      FileInputStream("virdemo1.dat");
                 [4] fStream.read(data,0,fileLength);
                 [5] imgObj.loadDataFromByteArray(data);
                 [6] fStream.close( );
                 [7] imgObj.setProperties( );
                 [8] byte[ ] resArr = imgObj.getDataInByteArray( );
                 [9] System.out.println("byte array length : " +
                      resArr.length);
                 [10] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                      con.prepareCall("update ordvirtab set image =
                      ? where id = " + index);
                 stmt1.setCustomDatum(1,imgObj);
                 stmt1.execute( );
                 stmt1.close( ) ;
            }
            System.out.println("successful");
       }
       [11] catch(Exception e){
            System.out.println("exception raised " + e);
       }
}
```

The byteArrayBasedExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the fourth row of the database table.
2. Determines the size (in bytes) of the local file named virdemo1.dat and creates a byte array of the same size.
3. Creates a new FileInputStream object. This input stream contains the contents of virdemo1.dat.
4. Reads the contents of the input stream into the byte array.

5. Uses the loadDataFromByteArray( ) method to load the media data in the byte array into the database ORDVir object and into imgObj. This also sets the local field on imgObj, but not the database object.
6. Closes the input stream.
7. Calls the setProperties( ) method to extract property values from the media data and set them in the application ORDVir object. See "setProperties( )" for a list of the property values extracted and set.
8. Uses the getDataInByteArray( ) method to get the media data from the application ORDVir object and load it into a local byte array named resArr.
9. Get the length of resArr and print it to the screen to verify the success of the loading.
10. Creates and executes a SQL statement that will update the database ORDVir object with the contents of imgObj. This update will set the attributes on the database object to match the application object.
11. Handles any errors or exceptions raised by the code.

## 2.2.9 processExample( ) Method

Example 2–10 shows a user-defined method named processExample( ) that uses the process( ) and processCopy( ) methods to manipulate the media data in the application object.

***Example 2–10 processExample( ) Method***

```
public void processExample(OracleConnection con){
     try{
          int index1 = 0;
          [1] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)s1.executeQuery
               ("select * from ORDVIRTAB where id = 2 for update ");
          while(rs1.next( )){
               index1 = rs1.getInt(1);
               OrdVir imgObj = (OrdVir) rs1.getCustomDatum(2,
                    OrdVir.getFactory( ));
               [2] OrdVir imgObj2 = (OrdVir) rs1.getCustomDatum(3,
                    OrdVir.getFactory( ));
               try{
                    [3] imgObj.processCopy("maxScale=32 32, fileFormat=
                         GIFF", imgObj2);
                    [4] imgObj.process("fileFormat=JFIF");
```

```
                    [5] System.out.println(getAllAttributesAsString
                         (imgObj));
                    [6] System.out.println(getAllAttributesAsString(imgObj2));
               }
               [7] catch (Exception e){
                    System.out.println("Exception raised in process"
                         + e );
               }
               [8] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordvirtab set image =
                    ?, image2 = ? where id = " + index1);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.setCustomDatum(2,imgObj2);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs1.close( );
          s1.close( );
     }
     [9] catch(Exception e){
          System.out.println("Exception raised: " + e);
     }
     System.out.println("successful");
}
```

The processExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the second row of the database table. The database ORDVir object is named image.
2. Creates a local ORDVir object named imgObj2. Populates imgObj2 with the contents of the ORDVir object in the third column of the current row in the OracleResultSet. This database ORDVir column is named image2.
3. Populates the image data in imgObj2 with a 32 x 32 GIF thumbnail image generated from the image data in imgObj. imgObj is unchanged by this operation.
4. Uses the process( ) method to convert the image in imgObj to a JPEG (JFIF) image.

5. Gets the attributes of imgObj by using the getAllAttributesAsString( ) method, and prints the attributes to the screen. See Section 2.2.5 for more information on the getAllAttributesAsString( ) method.
6. Gets the attributes of imgObj2 by using the getAllAttributesAsString( ) method, and prints the attributes to the screen. See Section 2.2.5 for more information on the getAllAttributesAsString( ) method.
7. Handles any errors or exceptions raised by the code in steps 3 through 6.
8. Creates and executes a SQL statement that will update the appropriate database ORDVir objects with the contents of imgObj and imgObj2.
9. Handles any errors or exceptions raised by the code.

## 2.2.10 similarExample( ) Method

Example 2–11 shows a user-defined method named similarExample( ) that uses the analyze( ) and similar( ) methods to compare two application objects.

***Example 2–11 similarExample( ) Method***

```
public void similarExample(OracleConnection con){
     try{
          int index1 = 0;
          [1] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)s1.executeQuery
               ("select * from ordvirtab where id = 5 for update ");
          while(rs1.next( )){
               index1 = rs1.getInt(1);
               OrdVir imgObj1 = (OrdVir) rs1.getCustomDatum
                    (2, OrdVir.getFactory( ));
               [2] OrdVir imgObj2 = (OrdVir) rs1.getCustomDatum
                    (3, OrdVir.getFactory( ));
               [3] try{
                    imgObj1.analyze( );
                    imgObj2.analyze( );
               }
               [4] catch (Exception e){
                    System.out.println("Exception raised in analyze" + e );
               }
               [5] if (1 == imgObj1.similar(imgObj2.getSignature( ),
                    "texture=50,structure=50", 5)){
                    System.out.println("imgObj1 is similar to imgObj2");
               }
```

```
                else{
                     System.out.println("imgObj1 is not similar to imgObj2");
                }
                [6] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update ordvirtab set image = ?,
                     image2 = ? where id = " + index1);
                stmt1.setCustomDatum(1,imgObj1);
                stmt1.setCustomDatum(2,imgObj2);
                stmt1.execute( );
                stmt1.close( ) ;
           }
           rs1.close( );
           s1.close( );
      }
      [7] catch(Exception e){
           System.out.println("Exception raised: " + e);
      }
      System.out.println("successful");
}
```

The similarExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the fifth row of the database table. The database ORDVir object is named image.
2. Creates a local ORDVir object named imgObj2. Populates imgObj2 with the contents of the ORDVir object in the third column of the current row in the OracleResultSet. This database ORDVir column is named image2.
3. Uses the analyze( ) method to generate signatures for image and image2.
4. Handles any errors or exceptions raised by the analyze( ) method.
5. Compares the two ORDVir objects with the similar( ) method, with equal weight being placed on the texture and structure attributes. If the score is lower than 5, the images match and the method will return 1. If similar( ) returns 1, a success message is printed to the screen; if it returns 0, a failure method is printed to the screen.
6. Creates and executes a SQL statement that will update the appropriate database ORDVir objects with the contents of imgObj and imgObj2.
7. Handles any errors or exceptions raised by the code.

## 2.2.11 scoreExample( ) Method

Example 2–12 shows a user-defined method named scoreExample( ) that uses the analyze( ) and score( ) methods to create a score for an application object.

***Example 2–12 scoreExample( ) Method***

```
public void scoreExample(OracleConnection con){
     try{
          int index1 = 0;
          [1] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)s1.executeQuery
               ("select * from ordvirtab where id = 6 for update ");
          while(rs1.next( )){
               index1 = rs1.getInt(1);
               OrdVir imgObj1 = (OrdVir) rs1.getCustomDatum
                    (2, OrdVir.getFactory( ));
               [2] OrdVir imgObj2 = (OrdVir) rs1.getCustomDatum
                    (3, OrdVir.getFactory( ));
               [3] try{
                    imgObj1.analyze( );
                    imgObj2.analyze( );
               }
               [4] catch (Exception e){
                    System.out.println("Exception raised in analyze" + e );
               }
               [5] System.out.println("Score = " + imgObj1.score
                    (imgObj2.getSignature( ),"globalColor=100"));
               [6] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordvirtab set image = ?,
                    image2 = ? where id = " + index1);
               stmt1.setCustomDatum(1,imgObj1);
               stmt1.setCustomDatum(2,imgObj2);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs1.close( );
          s1.close( );
     }
     [7] catch(Exception e){
          System.out.println("Exception raised: " + e);
     }
     System.out.println("successful");
}
```

The similarExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVir object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the sixth row of the database table. The database ORDVir object is named image.
2. Creates a local ORDVir object named imgObj2. Populates imgObj2 with the contents of the ORDVir object in the third column of the current row in the OracleResultSet. This database ORDVir column is named image2.
3. Uses the analyze( ) method to generate signatures for image and image2.
4. Handles any errors or exceptions raised by the analyze( ) method.
5. Generates the score of the comparison of the signatures of image and image2 with the score( ) method, and prints the score to the screen. The only attribute being compared is globalColor.
6. Creates and executes a SQL statement that will update the appropriate database ORDVir objects with the contents of imgObj and imgObj2.
7. Handles any errors or exceptions raised by the code.

# 3

# ORDVir Reference Information

Oracle Visual Information Retrieval Java Classes describes the ORDVir object type.

Methods invoked at the ORDVir level that are handed off for processing to the database source plug-in have `byte[ ] ctx[ ]` as a context parameter. In cases where a client system is connecting to a database server, the space for the parameter is created by the client (in the reference examples, 4000 bytes of space), but the content of the context parameter is generated by the server. The context parameter is passed from the client to the server for the processing of context information.

> **Note:** In the current release, not all source plug-ins will use or generate the context parameter, but if you include the parameter as previously described, your application should work with any current or future source plug-in.

See *Oracle Visual Information Retreival User's Guide and Reference* for more information.

## 3.1 Prerequisites

You will need to include the following import statements in your Java file to run Visual Information Retrieval methods:

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

The examples in this reference chapter are based on the assumption that the following operations have already been performed:

- A connection has been made to a table that contains a column of type ORDVir.
- A local ORDVir object named virObj has been created and populated with data.

For examples of making a connection and populating a local object, see Section 2.2.2.

## 3.2 Reference Information

This section presents reference information on the methods that operate on ORDVir objects.

# analyze( )

## Format

public void analyze( )

## Description

Sets the value for the application ORDVir object signature attribute based on the image data.

## Parameters

None.

## Return Value

None.

## Exceptions

SQLException

## Example

```
virObj.analyze( );
```

# checkProperties( )

## Format

public boolean checkProperties( )

## Description

Checks whether or not the properties stored in the media data of the local object are consistent with the attributes of the local object.

## Parameters

None.

## Return Value

This method returns TRUE if the attribute values stored in the object attributes are the same as the properties stored in the image data; otherwise, this method returns FALSE.

## Exceptions

java.sql.SQLException

## Example

```
if(virObj.checkProperties( ))
     System.out.println("checkProperties successful");
```

# clearLocal( )

## Format

public void clearLocal( )

## Description

Clears the source local field of the application ORDVir object.

## Parameters

None.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.clearLocal( );
```

# convert( )

## Format

public int convert(String operations)

## Description

Converts the application ORDVir object signature data to a format usable on the database server (host) machine.

## Parameters

**operations**
The type of processing done to the image signature. The following operations are available:

| Operation Keyword | Description |
|---|---|
| BYTEORDER | Converts the signature to the natural order of the host machine, regardless of its initial state. |

## Return Value

This method returns 0 if successful; otherwise, this method returns a non-zero integer.

## Exceptions

SQLException

## Example

```
int i = virObj.convert("BYTEORDER");
if(i == 0)
     System.out.println("convert successful");
```

where:

- BYTEORDER: is the processing done to the image signature.

# convert( ) (static)

## Format

public static int convert(byte[ ] signature, String operations, OracleConnection connection)

## Description

Converts the application ORDVir object signature data to a format usable on the database server (host) machine.

Unlike the non-static version of this function, the static function can directly operate on signature data without an instance of an ORDVir object.

## Parameters

**signature**
The signature of the image, as created by the analyze( ) method.

**operations**
The type of processing done to the image signature. The following operations are available:

| Operation Keyword | Description |
|---|---|
| BYTEORDER | Converts the signature to the natural order of the host machine, regardless of its initial state. |

**connection**
An object that represents the connection to the database.

## Return Value

This method returns 0 if successful; otherwise, this method returns a non-zero integer.

## Exceptions

SQLException

## Example

```
int i = virObj.convert(signature,"BYTEORDER",connection);
if(i == 0)
     System.out.println("convert successful");
```

where:

- signature: is the signature of the image.
- BYTEORDER: is the processing done to the image signature.
- connection: is the connection to the database.

# copy( )

## Format

public void copy(OrdVir dest)

## Description

Copies image data from the application ORDVir object source to a destination ORDVir object on the server.

## Parameters

**dest**
The ORDVir object to which the data will be copied.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
//create and populate an object named virObj2
virObj.copy(virObj2);
```

where:

- virObj2: is an ORDVir object that will receive the image data from virObj.

# deleteContent( )

## Format

public void deleteContent( )

## Description

Deletes the media data in the BLOB in the application ORDVir object.

## Parameters

None.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
audObj.deleteContent( );
```

# export( )

## Format

public void export (byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

## Description

Exports the data from the application ORDVir object to the location specified in the parameters. The location is of the form `sourceType://sourceLocation/sourceName`.

This method will only work if you are running Oracle database release 8.1.7 or higher.

See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

## Parameters

**ctx[ ]**
The source plug-in context information. It is set to NULL if there is no context information.

**sourceType**
The source type to which the content will be exported. Only "FILE" is natively supported.

**sourceLocation**
The location on the database server to which the content will be exported.

**sourceName**
The source name to which the content will be exported.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
byte[ ] ctx = new byte[4000][1];
virObj.export(ctx,"FILE","IMAGEDIR","image.gif");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the source type to which the content will be.
- IMAGEDIR: is the location on the server to which the content will be exported.
- image.gif: is the file to which the content will be exported.

# getBFILE( )

## Format

public oracle.sql.BFILE getBFILE( )

## Description

Gets the BFILE attribute of the application ORDVir object.

## Parameters

None.

## Return Value

This method returns the BFILE.

## Exceptions

java.sql.SQLException

## Example

```
BFILE imageBFILE = virObj.getBFILE( );
```

# getCompressionFormat( )

## Format

public String getCompressionFormat( )

## Description

Gets the compression format attribute of the application ORDVir object as a String.

## Parameters

None.

## Return Value

This method returns the compression format attribute, as a String.

## Exceptions

java.sql.SQLException

## Example

```
String compression = virObj.getCompressionFormat( );
```

# getContent( )

## Format

public oracle.sql.BLOB getContent( )

## Description

Gets the LOB locator from the application ORDVir object.

## Parameters

None.

## Return Value

This method returns the LOB locator of the application object.

## Exceptions

java.sql.SQLException

## Example

```
BLOB localContent = virObj.getContent( );
```

# getContentFormat( )

## Format

public String getContentFormat( )

## Description

Gets the content format attribute of the application ORDVir object as a String.

## Parameters

None.

## Return Value

This method returns the content format attribute, as a String.

## Exceptions

java.sql.SQLException

## Example

```
String format = virObj.getContentFormat( );
```

# getContentLength( )

## Format

public int getContentLength( )

## Description

Gets the content length of the media data in the application ORDVir object.

## Parameters

None.

## Return Value

This method returns the content length of the media data, in bytes.

## Exceptions

java.sql.SQLException

## Example

```
int length = virObj.getContentLength( );
```

# getDataInByteArray( )

## Format

public byte[ ] getDataInByteArray( )

## Description

Gets data from the LOB locator of the application ORDVir object and puts it in a local byte array.

## Parameters

None.

## Return Value

This method returns the byte array from which the data will be read.

## Exceptions

java.sql.SQLException

java.io.IOException

java.lang.OutOfMemoryError

## Example

```
byte[ ] byteArr = virObj.getDataInByteArray( );
```

# getDataInFile( )

## Format

public boolean getDataInFile(String filename)

## Description

Gets data from the LOB locator of the application ORDVir object and puts it in a local file.

## Parameters

**filename**
The name of the file that the data will be downloaded into.

## Return Value

This method returns TRUE if loading is successful; otherwise, this method returns FALSE.

## Exceptions

java.sql.SQLException

java.io.IOException

## Example

```
boolean load = virObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

where:

- output1.dat: is the file into which the data will be loaded.

# getDataInStream( )

## Format

public InputStream getDataInStream( )

## Description

Gets data from the LOB locator of the application ORDVir object and puts it in a local input stream.

## Parameters

None.

## Return Value

This method returns the input stream from which the data will be read.

## Exceptions

java.sql.SQLException

## Example

```
InputStream inpStream = virObj.getDataInStream( );
```

# getFormat( )

## Format

public String getFormat( )

## Description

Gets the format attribute of the application ORDVir object as a String.

## Parameters

None.

## Return Value

This method returns the format attribute as a String.

## Exceptions

java.sql.SQLException

## Example

```
String format = virObj.getFormat( );
```

# getHeight( )

## Format

public int getHeight( )

## Description

Gets the height of the application ORDVir object.

## Parameters

None.

## Return Value

This method returns the height of the ORDVir object, in pixels.

## Exceptions

java.sql.SQLException

## Example

```
int height = virObj.getHeight( );
```

# getImage( )

## Format

public OrdImage getImage( )

## Description

Gets the application ORDVir object ORDImage object information.

## Parameters

None.

## Return Value

This method returns the ORDImage object.

## Exceptions

java.sql.SQLException

## Example

```
ORDImage imgObj = virObj.getImage( );
```

# getMimeType( )

## Format

public String getMimeType( )

## Description

Gets the MIME type of the application ORDVir object as a String.

## Parameters

None.

## Return Value

This method returns the MIME type of the ORDVir object, as a String.

## Exceptions

java.sql.SQLException

## Example

```
String mime = virObj.getMimeType( );
```

# getSignature( )

## Format

public byte[ ] getSignature( )

## Description

Gets the signature of the application ORDVir object.

## Parameters

None.

## Return Value

This method returns the signature of the application ORDVir object.

## Exceptions

java.sql.SQLException

## Example

```
byte[ ] sig = virObj.getSignature( );
```

# getSource( )

## Format

public String getSource( )

## Description

Gets the application ORDVir object source information, including the source location, name, and type.

## Parameters

None.

## Return Value

This method returns a String containing the object source information.

## Exceptions

java.sql.SQLException

## Example

```
String sourceName = virObj.getSource( );
```

# getSourceLocation( )

## Format

public String getSourceLocation( )

## Description

Gets the application ORDVir object source location as a String.

## Parameters

None.

## Return Value

This method returns a String containing the object source location.

## Exceptions

java.sql.SQLException

## Example

```
String location = virObj.getSourceLocation( );
```

# getSourceName( )

## Format

public String getSourceName( )

## Description

Gets the application ORDVir object source name as a String.

## Parameters

None.

## Return Value

This method returns a String containing the object source name.

## Exceptions

java.sql.SQLException

## Example

```
String name = virObj.getSourceName( );
```

# getSourceType( )

## Format

public String getSourceType( )

## Description

Gets the application ORDVir object source location as a String.

## Parameters

None.

## Return Value

This method returns a String containing the object source type.

## Exceptions

java.sql.SQLException

## Example

```
String type = virObj.getSourceType( );
```

# getUpdateTime( )

## Format

public java.sql.Timestamp getUpdateTime( )

## Description

Gets a Timestamp object that contains information on when the application ORDVir object was most recently updated.

## Parameters

None.

## Return Value

This method returns a Timestamp object that contains the time of the most recent update.

## Exceptions

java.sql.SQLException

## Example

```
Timestamp time = virObj.getUpdateTime( );
```

# getWidth( )

## Format

public int getWidth( )

## Description

Gets the width of the application ORDVir object.

## Parameters

None.

## Return Value

This method returns the width of the ORDVir object, in pixels.

## Exceptions

java.sql.SQLException

## Example

```
int width = virObj.getWidth( );
```

# importData( )

## Format

public void importData(byte[ ] ctx[ ])

## Description

Imports data from an external source into the application ORDImage object.

The srcType, srcLocation, and srcName attributes must all be defined for this method to work.

## Parameters

**ctx[ ]**
The source plug-in context information. It is set to NULL if there is no context information.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
byte[ ] ctx = new byte[4000][1];
virObj.importData(ctx)
```

where:

- ctx: contains the source plug-in context information.

# importFrom( )

## Format

public void importFrom(byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

## Description

Imports data from an external source into the application ORDVir object. The location of the external source is of the form `sourceType://sourceLocation/sourceName`.

## Parameters

**ctx[ ]**
The source plug-in context information. See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

**sourceType**
The source type from which the data will be imported.

**sourceLocation**
The source location from which the data will be imported.

**sourceName**
The source name from which the data will be imported.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
byte[ ] ctx = new byte[4000][1];
virObj.importFrom("FILE","IMAGEDIR","testimg.dat");
```

where:

- ctx: contains the source plug-in context information.

- FILE: is the type of source from which the data will be imported.
- IMAGEDIR: is the location of the file from which the data will be imported.
- testimg.dat: is the file from which the data will be imported.

# isLocal( )

## Format

public boolean isLocal( )

## Description

Checks if the application ORDVir object local attribute is set.

## Parameters

None.

## Return Value

This method returns TRUE if the ORDVir object local attribute is set; otherwise, this method returns FALSE.

## Exceptions

java.sql.SQLException

## Example

```
if(virObj.isLocal( ))
     System.out.println("local attribute is true");
else
     System.out.println("local attribuite is false");
```

# loadDataFromByteArray( )

## Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

## Description

Loads data from the local byte buffer into the database ORDVir object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDVir object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

## Parameters

**byteArr**
The name of the local byte array from which to load data.

## Return Value

This method returns TRUE if loading is successful; otherwise, this method returns FALSE.

## Exceptions

java.sql.SQLException

java.io.IOException

## Example

```
byte[ ] data = new byte[32000];
FileInputStream fStream = new FileInputStream("testimg.dat");
fStream.read(data,0,32300);
boolean success = virObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

where:

- data: is the local byte array from which the data will be loaded.
- testimg.dat: is a local file that contains 32,300 bytes of data.

# loadDataFromFile( )

## Format

public boolean loadDataFromFile(String filename)

## Description

Loads data from the local file into the database ORDVir object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDVir object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

## Parameters

**filename**
The name of the local file from which to load data.

## Return Value

This method returns TRUE if loading is successful; otherwise, this method returns FALSE.

## Exceptions

java.sql.SQLException

java.io.IOException

## Example

```
virObj.loadDataFromFile("testimg.dat");
```

where:

- testimg.dat: is a local file that contains media data.

# loadDataFromInputStream( )

## Format

public boolean loadDataFromInputStream(InputStream inpStream)

## Description

Loads data from the local input stream into the database ORDVir object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDVir object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

## Parameters

**inpStream**
The name of the local input stream from which to load data.

## Return Value

This method returns TRUE if loading is successful; otherwise, this method returns FALSE.

## Exceptions

java.sql.SQLException

java.io.IOException

## Example

```
FileInputStream fStream = new FileInputStream("testimg.dat");
virObj.loadDataFromInputStream(fStream);
```

where:

- testimg.dat: is a local file that contains media data.
- fStream: is the local input stream that will load data into the ORDVir object.

# OrdVir( )

## Format

public OrdVir( )

## Description

Creates an ORDVir object.

This method is the default ORDVir constructor.

## Parameters

None.

## Return Value

None.

## Exceptions

None.

## Example

None.

# process( )

## Format

public void process(String command)

## Description

Executes a given command on the application ORDVir object.

For more information on the commands that can be processed, see *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*.

## Parameters

**command**
The command to be executed.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.process("fileFormat=JFIF");
```

where:

- fileFormat=JFIF: is the command to be executed.

# processCopy( )

## Format

public void processCopy(String command, OrdVir dest)

## Description

Copies data from the application ORDVir object to a destination ORDVir object and modifies the copy according to the specified command.

## Parameters

**command**
The command to be executed.

**dest**
The object that will receive the results of the command.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
//create and populate an OrdVir object named virObj2
virObj.processCopy("maxScale=32 32, fileFormat= GIFF", virObj2);
```

where:

- maxScale=32 32, fileFormat= GIFF: is the command to be executed.
- virObj2: is the object that will receive the results of the command.

# score( )

## Format

public float score(byte[ ] signature2, String attrWeights)

## Description

Compares the image signature of the application ORDVir object to the signature stored in signature2 using the weights stored in attrWeights.

## Parameters

**signature2**
The signature to be compared with the signature of the application ORDVir object.

**attrWeights**
A list of weights to apply to each visual attribute. The following attributes can be specified. You must specify a value greater than 0.0 for at least one attribute. There is no mandatory value that the weights must add to; however, it is recommended that you ensure that you are consistent with the total used in your application. The visual attributes are the following:

| Attribute | Description |
|---|---|
| globalcolor | The weight value assigned to the global color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| localcolor | The weight value assigned to the local color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| texture | The weight value assigned to the texture visual attribute.<br>Data type is String.<br>Default is 0.0. |
| structure | The weight value assigned to the structure visual attribute.<br>Data type is String.<br>Default is 0.0. |

## Return Value

This method returns the score, which is a value between 0.0 and 100.0.

## Exceptions

java.sql.SQLException

## Example

```
float score = virObj.score(signature2, "localcolor=1.0");
```

where:

- signature2: is the signature to be compared with the signature of the application ORDVir object.
- localcolor=1.0: is the weight to apply to the visual attributes.

# score( ) (static)

## Format

```
public static float score(byte[ ] signature1, byte[ ] signature2, String attrWeights,
                          OracleConnection connection)
```

## Description

Compares the image signature stored in signature1 to the signature stored in signature2 using the weights stored in attrWeights.

Unlike the non-static version of this function, the static function can directly operate on signature data without an instance of an ORDVir object.

## Parameters

**signature1**
The first signature.

**signature2**
The second signature, which will be compared to signature1.

**attrWeights**
A list of weights to apply to each visual attribute. The following attributes can be specified. You must specify a value greater than 0.0 for at least one attribute. There is no mandatory value that the weights must add to; however, it is recommended that you ensure that you are consistent with the total used in your application. The visual attributes are the following:

| Attribute | Description |
|---|---|
| globalcolor | The weight value assigned to the global color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| localcolor | The weight value assigned to the local color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| texture | The weight value assigned to the texture visual attribute.<br>Data type is String.<br>Default is 0.0. |

| Attribute | Description |
|---|---|
| structure | The weight value assigned to the structure visual attribute.<br>Data type is String.<br>Default is 0.0. |

**connection**
An object that represents the connection to the database.

## Return Value

This method returns the score, which is a value between 0.0 and 100.0.

## Exceptions

java.sql.SQLException

## Example

```
float score = virObj.score(signature1, signature2, "localcolor=1.0",
     connection);
```

where:

- signature1: is the first signature.
- signature2: is the second signature, which will be compared to signature1.
- localcolor=1.0: is the weight to apply to the visual attributes.
- connection: is the connection to the database.

# setCompressionFormat( )

## Format

public void setCompressionFormat(String CompressionFormat)

## Description

Sets the compression format attribute of the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**CompressionFormat**
The compression format to be set.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

None.

# setContentFormat( )

## Format

public void setContentFormat(String ContentFormat)

## Description

Sets the content format attribute of the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**ContentFormat**
The content format to be set.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

None.

# setContentLength( )

## Format

public void setContentLength(int newContentLength)

## Description

Sets the content length of the media data in the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**newContentLength**
The new content length to be set, in bytes.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

None.

# setFormat( )

## Format

public void setFormat(String Format)

## Description

Sets the format attribute of the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**Format**
The format of the contents of the ORDVir object, as a String object.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

None.

# setHeight( )

## Format

public void setHeight(int newHeight)

## Description

Sets the height of the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**newHeight**
The new height to be set, in pixels.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setHeight(24);
```

where:

- 24: is the height to be set, in pixels.

# setImage( )

## Format

public void setImage(OrdImage image)

## Description

Sets the ORDImage object in the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**image**
The ORDImage object to be set.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
//create and populate an OrdImage object named imgObj
virObj.setImage(imgObj);
```

where:

- imgObj: is the ORDImage object to be set.

# setLocal( )

## Format

public void setLocal( )

## Description

Sets the source local field of the application ORDVir object.

## Parameters

None

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setLocal( );
```

# setMimeType( )

## Format

public void setMimeType(String MimeType)

## Description

Sets the MIME type of the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**MimeType**
The MIME type of the contents of the ORDVir object, as a String.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setMimeType("image/bmp");
```

where:

- image/bmp: is the MIME type to be set.

# setProperties( )

## Format

public void setProperties( )

## Description

Reads the image data, extracts attributes, and sets the application ORDVir object attributes according to the values stored in the content data header.

The properties to be set include height, width, data size of the on-disk image, file type, image type, compression type, and MIME type.

## Parameters

None.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setProperties( );
```

# setProperties(String)

## Format

public void setProperties(String command)

## Description

Sets the application ORDVir object attributes according to the values stored in the given String. This method is used for foreign image files. The properties that can be set are listed in *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*.

## Parameters

**command**
The object attribute values to be set.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
String properties = "width=123 height=321 compressionformat=NONE
     userString=DJM dataOffset=128 scanlineorder=INVERSE
     pixelorder=REVERSE interleaving=BIL numberOfBands=1
     defaultChannelSelection=1";
virObj.setProperties(properties);
```

where:

- properties: is a String that contains the properties to be set.

# setSignature( )

## Format

public void setSignature(byte[ ] signature)

## Description

Sets the signature of the application ORDVir object. This signature must be generated by the analyze( ) method.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**signature**
The signature to be set.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setSignature(imgSignature);
```

where:

- imgSignature: is a signature that is generated by the analyze( ) method.

# setSource( )

## Format

public void setSource(String sourceType, String sourceLocation, String sourceName)

## Description

Sets the application ORDVir object source information.

## Parameters

**sourceType**
The type of the source.

**sourceLocation**
The location of the source.

**sourceName**
The name of the source.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setSource("FILE","IMAGEDIR","jdoe.gif");
```

where:

- FILE: is the source type.
- IMAGEDIR: is the source location.
- jdoe.gif: is the source name.

# setUpdateTime( )

## Format

public void setUpdateTime(java.sql.Timestamp currentTime)

## Description

Sets the update time in the application ORDVir object to the current time.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**currentTime**
The current time, which will be set in the ORDVir object. This value should be set to null; the method will then use the system date (SYSDATE) of the database server.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setUpdateTime(null);
```

# setWidth( )

## Format

public void setWidth(int newWidth)

## Description

Sets the width of the application ORDVir object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). This method will set only the attribute value; it does not change the media file itself.

## Parameters

**newWidth**
The width to be set, in pixels.

## Return Value

None.

## Exceptions

java.sql.SQLException

## Example

```
virObj.setWidth(24);
```

where:

- 24: is the width to be set, in pixels.

# similar( )

## Format

public int similar(byte[ ] signature2, String attrWeights, float threshold)

## Description

Compares the image signatures in the application ORDVir object and signature2, using the weights provided. If the result of the comparison is less than or equal to the provided threshold, the images are considered a match.

## Parameters

**signature2**
The signature to be compared to the signature of the application ORDVir object.

**attrWeights**
A list of weights to apply to each visual attribute. The following attributes can be specified. You must specify a value greater than 0.0 for at least one attribute. There is no mandatory value that the weights must add to; however, it is recommended that you ensure that you are consistent with the total used in your application. The visual attributes are the following:

| Attribute | Description |
|---|---|
| globalcolor | The weight value assigned to the global color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| localcolor | The weight value assigned to the local color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| texture | The weight value assigned to the texture visual attribute.<br>Data type is String.<br>Default is 0.0. |
| structure | The weight value assigned to the structure visual attribute.<br>Data type is String.<br>Default is 0.0. |

**threshold**
The value that must be attained to be considered a match.

## Return Value

This method returns 1 if the images match; otherwise, this method returns 0.

## Exceptions

java.sql.SQLException

## Example

```
int i = virObj.similar(signature2, "localcolor=1.0", 10);
```

where:

- signature2: is the signature to be compared with the signature of the application ORDVir object.
- localcolor=1.0: is the weight to apply to the visual attributes.
- 10: is the value that must be attained to be considered a match.

# similar( ) (static)

## Format

public static int similar(byte[ ] signature1, byte[ ] signature2, String attrWeights, float threshold,
OracleConnection connection)

## Description

Compares the image signatures in signature1 and signature2, using the weights provided. If the result of the comparison is less than or equal to the provided threshold, the images are considered a match.

Unlike the non-static version of this function, the static function can directly operate on signature data without an instance of an ORDVir object.

## Parameters

**signature1**
The first signature.

**signature2**
The second signature, which will be compared to signature1.

**attrWeights**
A list of weights to apply to each visual attribute. The following attributes can be specified. You must specify a value greater than 0.0 for at least one attribute. There is no mandatory value that the weights must add to; however, it is recommended that you ensure that you are consistent with the total used in your application. The visual attributes are the following:

| Attribute | Description |
|---|---|
| globalcolor | The weight value assigned to the global color visual attribute.<br>Data type is String.<br>Default is 0.0. |
| localcolor | The weight value assigned to the local color visual attribute.<br>Data type is String.<br>Default is 0.0. |

| Attribute | Description |
|---|---|
| texture | The weight value assigned to the texture visual attribute.<br>Data type is String.<br>Default is 0.0. |
| structure | The weight value assigned to the structure visual attribute.<br>Data type is String.<br>Default is 0.0. |

**threshold**
The value that must be attained to be considered a match.

**connection**
An object that represents the connection to the database.

## Return Value

This method returns 1 if the images match; otherwise, this method returns 0.

## Exceptions

java.sql.SQLException

## Example

```
int i = virObj.similar(signature1, signature2, "localcolor=1.0", 10,
     connection);
```

where:

- signature1: is the first signature.
- signature2: is the second signature, which will be compared to signature1.
- localcolor=1.0: is the weight to apply to the visual attributes.
- 10: is the value that must be attained to be considered a match.
- connection: is the connection to the database.

# A

# Running Java Classes Examples

A sample file (program written in Java) is provided in the installation of Oracle Visual Information Retrieval Java Classes. This file provides examples of how to build Java applications. It demonstrates loading data from various sources into database objects, downloading data from database objects to the file system, extracting and displaying metadata from the media content, and comparing images for similarity.

The name of the Java sample file is VirExample.java.

To run the Java sample file included with Visual Information Retrieval Java Classes, you must perform the following operations:

1. Install Oracle8*i* with Visual Information Retrieval.

   You must have the Oracle8*i* database server with Visual Information Retrieval installed on a server machine.

2. Check the values of local variables.

   All users must make sure that classes111.zip, ordvir817.zip, the JDK classes, and the SQLJ runtime.zip files are included in the CLASSPATH variable on the local machine, and the local directory containing the javac and java commands is included in the PATH variable on the local machine.

   Solaris users must make sure the directory that contains the file named ocijdbc8.so is included in the LD_LIBRARY_PATH variable.

   Windows NT users must make sure that the directory that contains the file named ocijdbc8.dll is included in the PATH variable.

3. Compile the Java file on your local machine.

   Using version 1.1.6 of the JDK, compile the sample programs using the following command:

```
javac VirExample.java
```

4. Connect to your database and run the SQL script that corresponds to your Java file.

   For the sample program to run, your database must include tables that contain a column of the appropriate object type. The Visual Information Retrieval Java Classes installation includes a SQL file that contains commands to create a new user, create a table, and add some sample data to the table.

   The name of the SQL script is VirExample.sql.

   ---

   **Note:** The SQL script connects to the database as the user *system*, with a password of *manager*. Edit the SQL files to change the password or remove the CONNECT statement before running the script.

   Also, edit the directory path to reflect your schema.

   ---

5. Run the compiled Java program using the following command:

   ```
   java VirExample
   ```

For more information on the sample file, see the readme file.

# B

# Exceptions and Errors

This appendix contains information on the exceptions and errors that can be raised by Oracle Visual Information Retrieval Java Classes.

## B.1 IOException

The IOException class signals that an I/O exception of some sort has occurred.

```
public class java.io.IOException extends java.lang.Exception {
     //Constructs a FileNotFoundException with no detailed message
     public IOException();

     //Constructs a FileNotFoundException with the specified detailed message
     public IOException(String s);
}
```

## B.2 OutOfMemoryError

The OutOfMemoryError class signals that the Java Virtual Machine cannot allocate an object because it is both out of memory and unable to make more memory available through garbage collecting (that is, through deleting objects that are no longer being used).

```
public class java.lang.OutOfMemoryError extends java.lang.VirtualMachineError {
     //Constructs an OutOfMemoryError with no detailed message
     public OutOfMemoryError();

     //Constructs an OutOfMemoryError with a detailed message
     public OutOfMemoryError(string s);
}
```

## B.3 SQLException

The SQLException class provides information on a database access error.

```
public class java.sql.SQLException extends java.lang.Exception {
     //The following four methods are public constructors:

     //Constructs a fully specified SQLException
     public SQLException(String reason, String SQLState, int vendorCode);

     //Construct a SQLException with venderCode value of 0
     public SQLException(String reason, String SQLState);

     //Construct a SQLException with vendorCode value of 0 and a null SQLState
     public SQLException(String reason);

     //Construct a SQLException with vendorCode of 0, a null SQLState. and a
     //null reason
     public SQLException();

     //The following four methods are public instance methods:

     //Get the vendor-specific exception code
     public int getErrorCode();

     //Get the exception connected to this one
     public SQLException getNextException();

     //Get the SQL state
     public String getSQLState();

     //Add a SQLException to the end
     public synchronized void setNextException(SQLException ex);
}
```

# C

# Deprecated Methods

The following list shows the methods that have been deprecated since release 8.1.5 of Visual Information Retrieval Java Client:

- protected int bindInSQLParams( )
- protected int declareSQLResults( )
- protected String defineSQLResults( )
- public void flush( )
- public String getAllAttributesAsString( )
- protected String getContentLengthAPI( )
- protected String getFormatStr( )
- protected String getMediaType( )
- protected String getSourceStr( )
- protected int getSQLResults( )
- protected String getUpdStr( )
- public void refresh( )
- protected void setLock( )
- protected String setSQLParams( )

# Index

## I



## L



## M



## O



## P



## R



## S

## V