File Handling and DB

Date	@October 6, 2024
tag	lecture

https://prod-files-secure.s3.us-west-2.amazonaws.com/5804e32f-5ad4-4d8f-bc8e-f28573f5a1b0/be18530b-b04c-4300-a254-1511a6d060c9/SDA-Lect08-Fall2024-File_Handling_and_DB.pdf

▼ File Handling

`java.io` `.File`

tells whether a file:

exists, is read protected, is write protected, is a directory

```
import java.io.File;
File myobj = new File("filename.txt");
```

File object can also refer to a directory"

`File file2 = new File("C:\sda");`

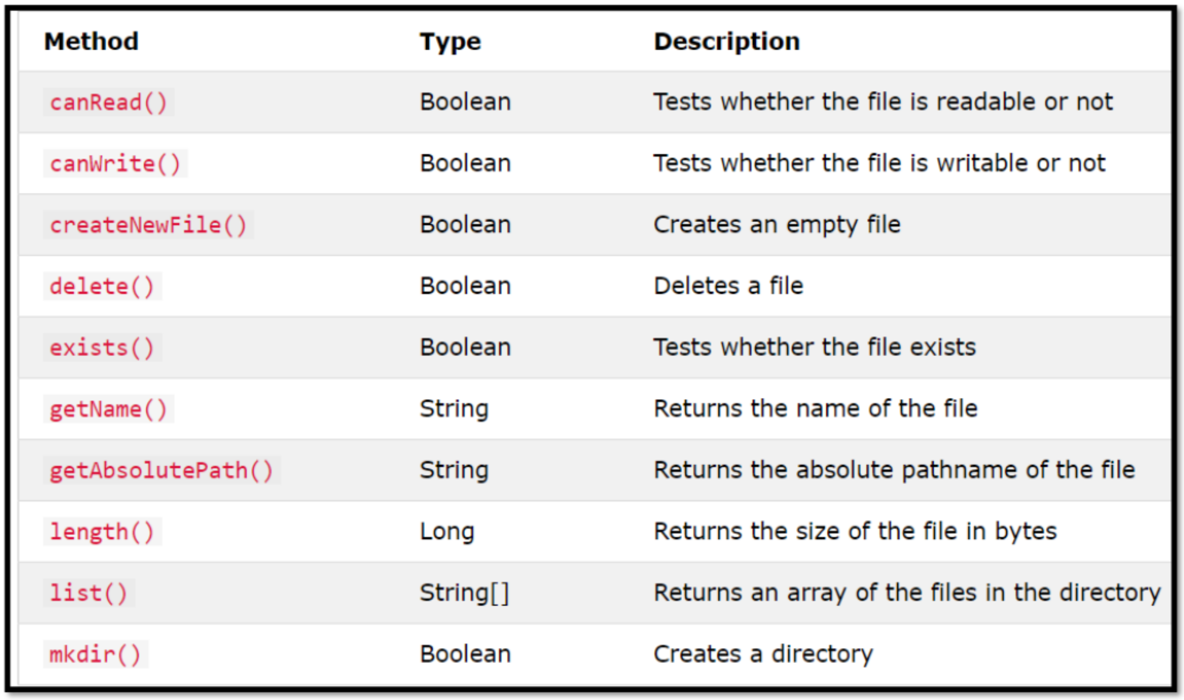
To obtain the path to current working directory:

`System.getProperty("user.dir");`

To obtain the file or path separator use

`System.getProperty ("file.separator");`
 `System.getProperty ("path.separator");`

Useful File Methods:



Method	Type	Description
`canRead()`	Boolean	Tests whether the file is readable or not
`canWrite()`	Boolean	Tests whether the file is writable or not
`createNewFile()`	Boolean	Creates an empty file
`delete()`	Boolean	Deletes a file
`exists()`	Boolean	Tests whether the file exists
`getName()`	String	Returns the name of the file
`getAbsolutePath()`	String	Returns the absolute pathname of the file
`length()`	Long	Returns the size of the file in bytes
`list()`	String[]	Returns an array of the files in the directory
`mkdir()`	Boolean	Creates a directory

Create a File:

```
import java.io.File;
import java.io.IOException;
public class CreateFile{
        public static void main(String[] args){
                try
                {
                    File myObj = new File("C:\\sda\\newfil
e.txt");
                    if(myObj.createNewFile()){
                        System.out.println("File created" +
myObj.getName());
                    }
                    else{
                        System.out.println("File already ex
ists");
                    }
                }
                catch(IOException e)
                {
                    System.out.println("An error occured");
                    e.printStackTrace();
                }
        }
}
```

Get File Information:

```
import java.io.File;
import java.io.IOException;
public class CreateFile{
        public static void main(String[] args){
                try
                {
                    File myObj = new File("C:\\sda\\newfil
e.txt");
                    if(myObj.exists()){
                        System.out.println("File created" +
myObj.getName());
                        System.out.println("Absolute Path:
" + myObj.getAbsolutePath());
                        System.out.println("Writeable: " +
myObj.canWrite());
                        System.out.println("Readable: " + m
yObj.canRead());
                        System.out.println("File Size in by
tes: " + myObj.length());
                    }
                    else{
                        System.out.println("File already ex
ists");
                    }
                }
                catch(IOException e)
                {
                    System.out.println("An error occured");
                    e.printStackTrace();
                }
        }
}
```

Directory Listing:

```
import java.io.*;
public class DirListing {
    public static void main(String[] args) {
        File dir = new File(System.getProperty(“user.di
r”));
        if (dir.isDirectory())
        {
            System.out.println(“Directory of ” + dir);
            String[] listing = dir.list();
            for (int i=0; i < listing.length; i++) {
                System.out.println(“\t” + listing[i]);
            }
        }
    }
}
```

jaca io package provides over 60 io classes

Streams

Byte Stream: 8 bits (chars, videos, audios, images)

Character Stream: 16 bits (text data)

File Output Stream:

```
import java.io.FileOutputStream;
public class Main{
    public static void main(String[] args){
        String data = "This is a line of text inside the fi
le.";
        try
        {
            FileOutputStream output = new FileOutputStream
("ouutput.txt");
            byte[] array = data.getBytes();
            output.write(array);
            output.close();
        }
        catch(Exception e){
        e.getStackTrace();
        }
    }
}
```

File Input Stream:

```
import java.io.FileOutputStream;
public class Main{
    public static void main(String[] args){
        try
        {
            FileOutputStream input = new FileInputStream("i
nput.txt");
            System.out.println("Data in the file: ");
            int i = input.read();
            while(i!=-1)
            {
                System.out.println((char)i);
                i=input.read();
            }
            input.close();
        }
        catch(Exception e){
        e.getStackTrace();
        }
    }
}
```

Write to a File:

```
import java.io.FileWriter;
public class WriteToFile{
    public static void main(String[] args){
        try
        {
            FileWriter myWriter = new FileWriter("newfile.t
xt");
            myWriter.write("Maryam");
            myWriter.close();
        }
        catch(IOException e)
        {
            //catch exception
        }
    }
}
```

Read a File:

```
import java.io.File;
public class ReadFromFile{
    public static void main(String[] args){
        try
        {
            FileReader fr = new FileReader("newfile.txt");
            Scanner myReader = new Scanner(fr);
            while(myReader.hasNextLine())
            {
                String data = myReader.nextLine();
                System.out.println(data);
            }
            myReader.close();
        }
        catch(IOException e)
        {
            //catch exception
        }
    }
}
```

Buffer Streams:

```
import java.io.*;
public class Copy {
    public static void main(String[] args) throws IOExcepti
on {
    // opening the streams
        FileReader in = new FileReader (“infile.txt");
        BufferedReader br = new BufferedReader(in);
        FileWriter out = new FileWriter ("outfile.txt");
        BufferedWriter bw = new BufferedWriter(out);
        // processing the streams
        String aLine = null;
        while ((aLine = br.readLine()) != null) {
            bw.write(aLine, 0, aLine.length());
        }
        // closing the streams
        in.close(); out.close();
    }
}
```

▼ Java Database Connectivity (JDBC)

its a Java API

java application → jdbc api → jdbc driver→ db

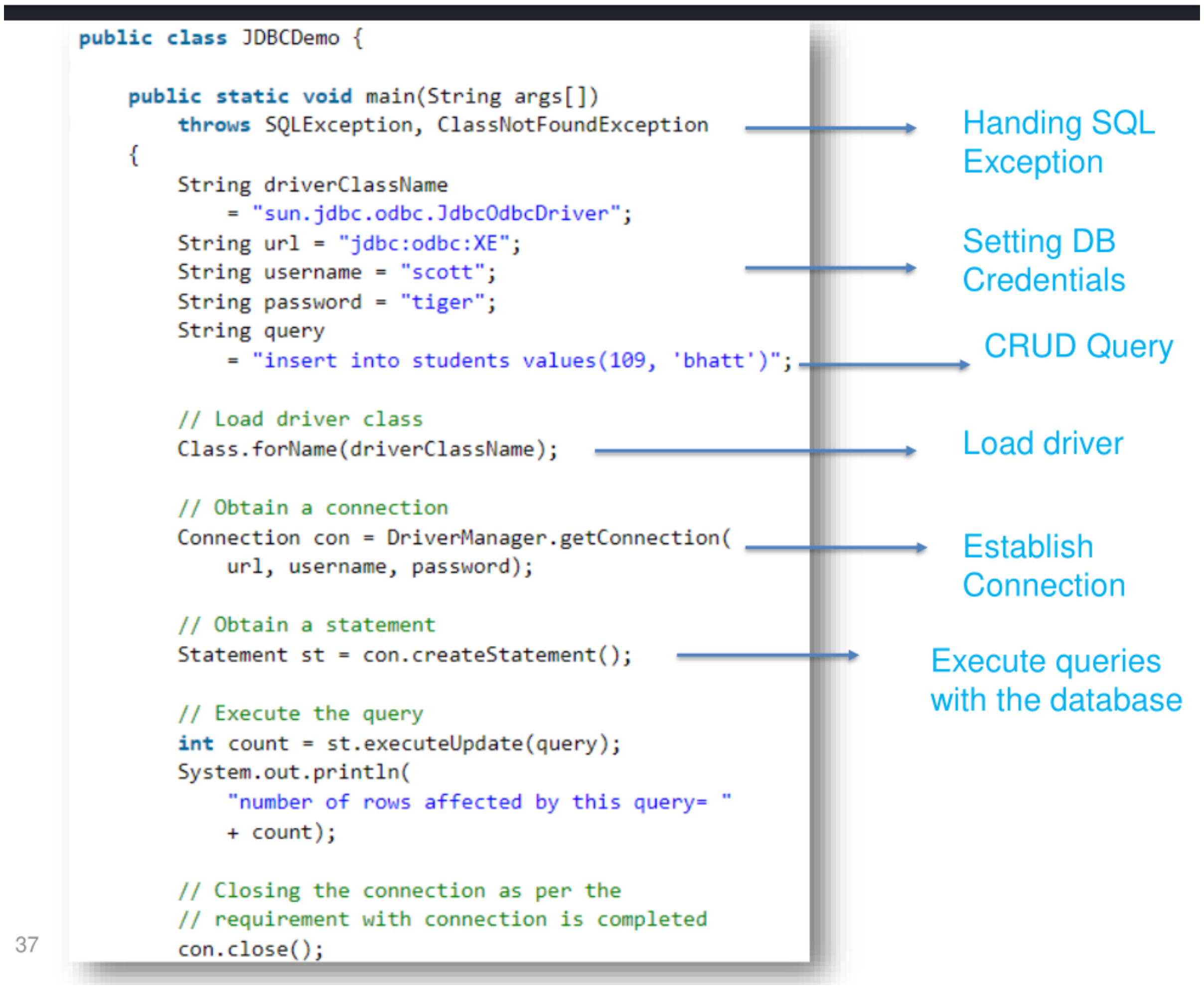
also the other way around

ODBC(was C based)

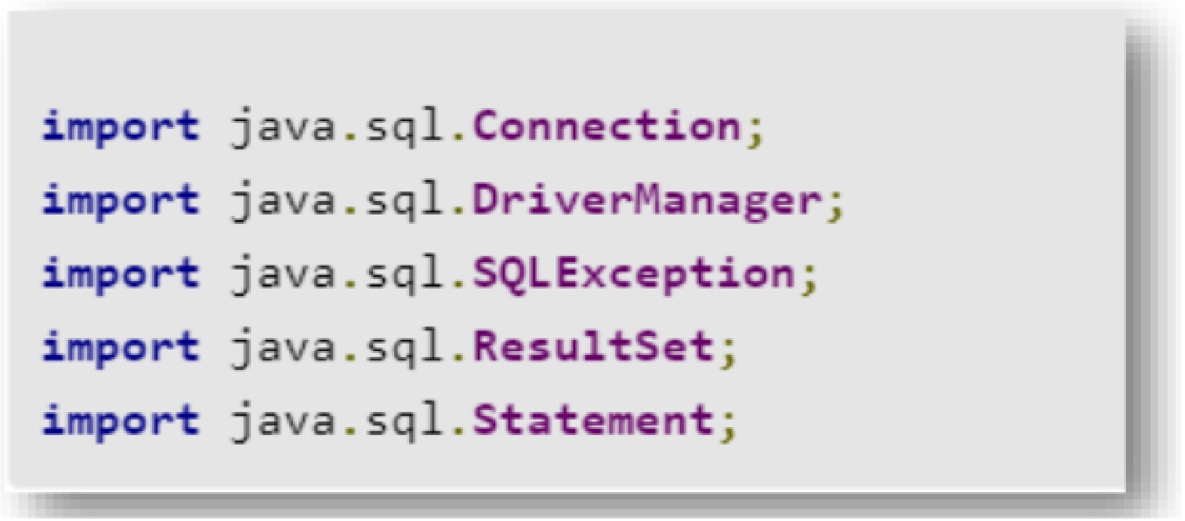
Steps:

1. Establish a connection.
2. Create JDBC Statements
3. Execute SQL Statements
4. GET ResultSet
5. Close connections

```
```



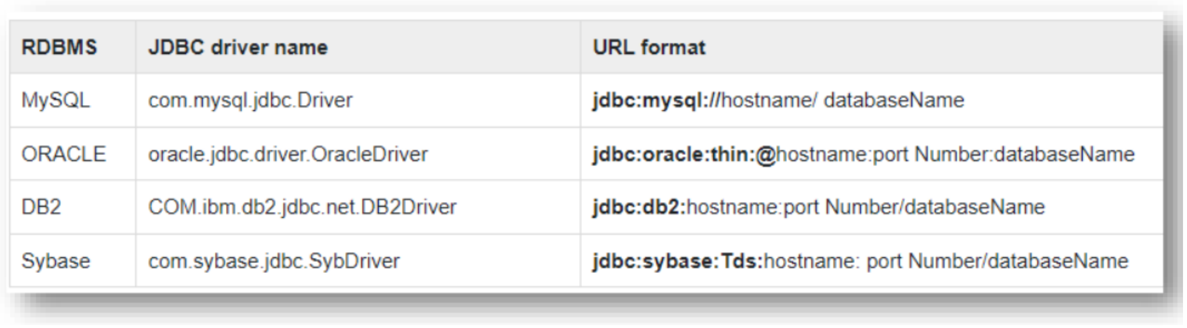
```java
public class JDBCDemo {

    public static void main(String args[])
        throws SQLException, ClassNotFoundException
    {
        String driverClassName
            = "sun.jdbc.odbc.JdbcOdbcDriver";
        String url = "jdbc:odbc:XE";
        String username = "scott";
        String password = "tiger";
        String query
            = "insert into students values(109, 'bhatt')";

        // Load driver class
        Class.forName(driverClassName);

        // Obtain a connection
        Connection con = DriverManager.getConnection(
            url, username, password);

        // Obtain a statement
        Statement st = con.createStatement();

        // Execute the query
        int count = st.executeUpdate(query);
        System.out.println(
            "number of rows affected by this query= "
            + count);

        // Closing the connection as per the
        // requirement with connection is completed
        con.close();
```



```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;
```

Connection interface

- Connection interface resides in **java.sql** package and it represents a session with a specific database you are connecting to.



RDBMS	JDBC driver name	URL format
MySQL	com.mysql.jdbc.Driver	**jdbc:mysql:**//hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:@**hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

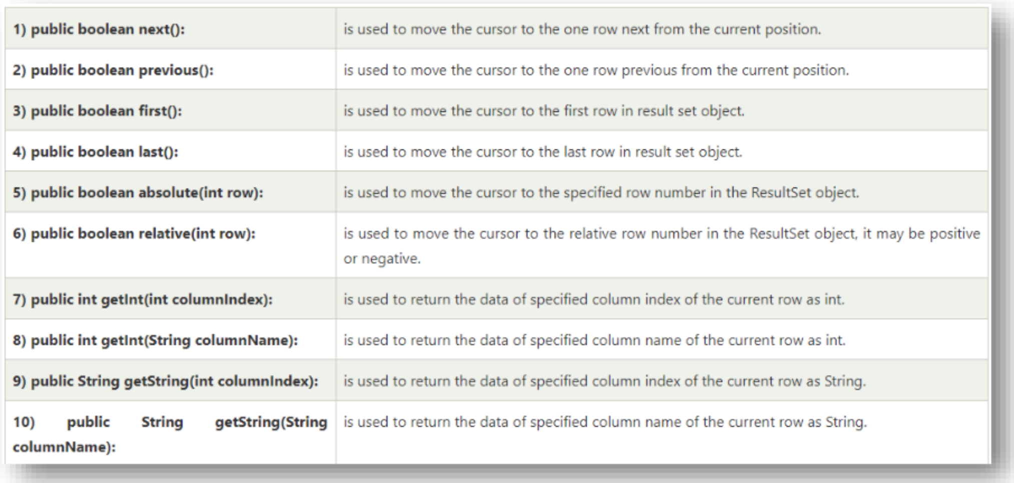
Statement interface

- The Statement interface *provides methods to execute queries* with the database.

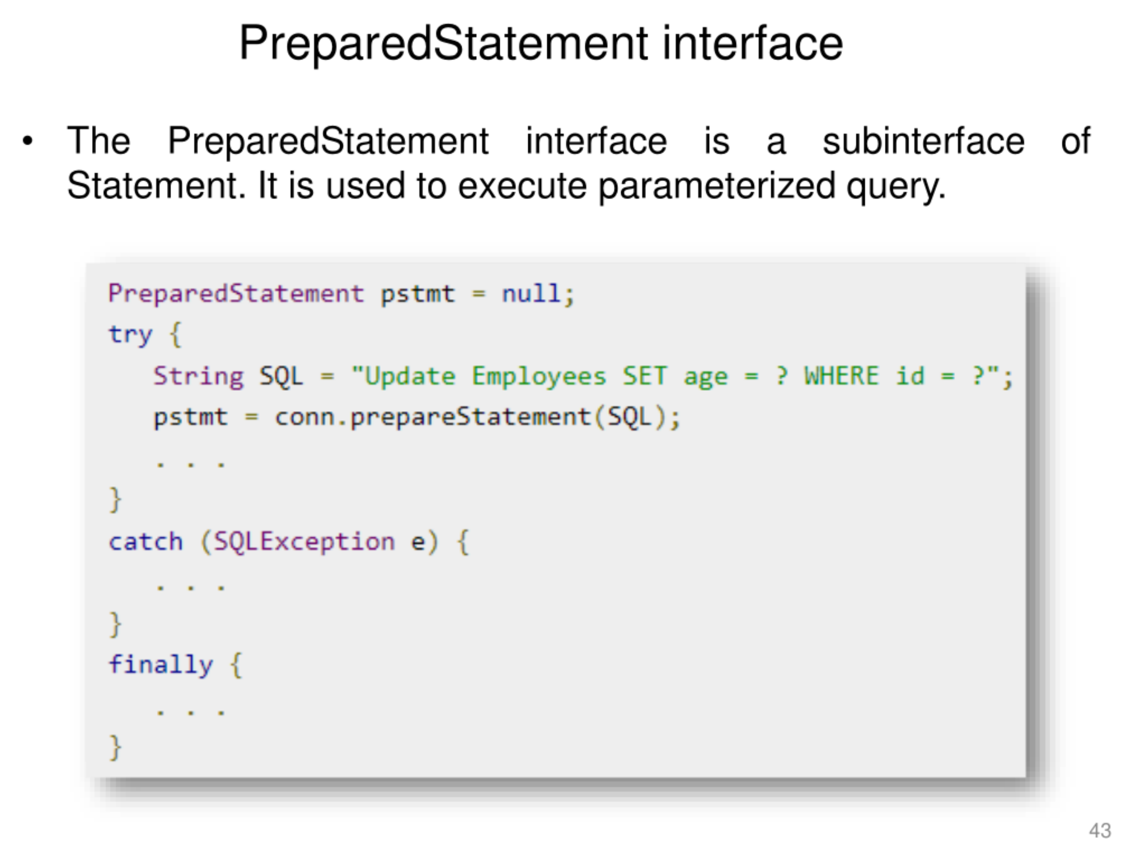
- The important methods of Statement interface are as follows:
  1. **public ResultSet executeQuery(String sql):** is used to execute SELECT query. It returns the object of ResultSet.
  2. **public int executeUpdate(String sql):** is used to execute specified query, it may be create, drop, insert, update, delete etc.
  3. **public boolean execute(String sql):** is used to execute queries that may return multiple results.
  4. **public int[] executeBatch():** is used to execute batch of commands.

ResultSet interface

- The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.



1) public boolean next():	is used to move the cursor to the one row next from the current position.
2) public boolean previous():	is used to move the cursor to the one row previous from the current position.
3) public boolean first():	is used to move the cursor to the first row in result set object.
4) public boolean last():	is used to move the cursor to the last row in result set object.
5) public boolean absolute(int row):	is used to move the cursor to the specified row number in the ResultSet object.
6) public boolean relative(int row):	is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative.
7) public int getInt(int columnIndex):	is used to return the data of specified column index of the current row as int.
8) public int getInt(String columnName):	is used to return the data of specified column name of the current row as int.
9) public String getString(int columnIndex):	is used to return the data of specified column index of the current row as String.
10) public String getString(String columnName):	is used to return the data of specified column name of the current row as String.



PreparedStatement interface

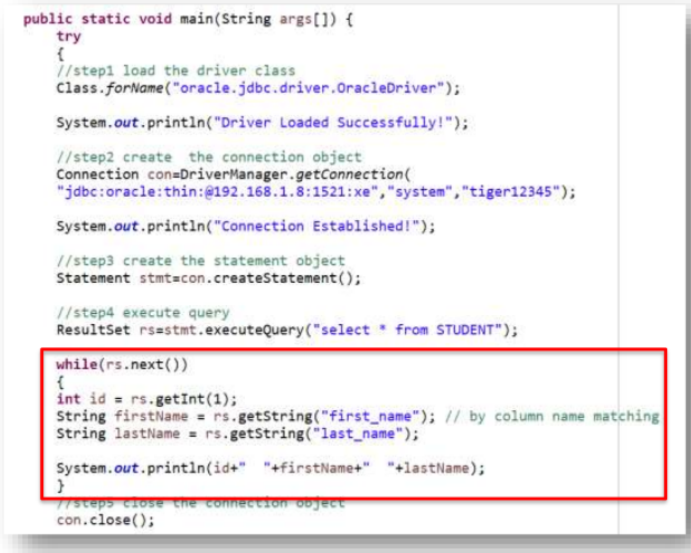
- The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

```
PreparedStatement pstmt = null;
try {
   String SQL = "Update Employees SET age = ? WHERE id = ?";
   pstmt = conn.prepareStatement(SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   . . .
}
```

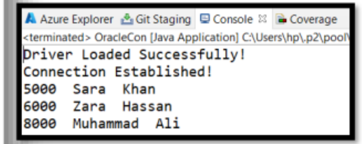
43

Oracle :

Retrieve Data from Database



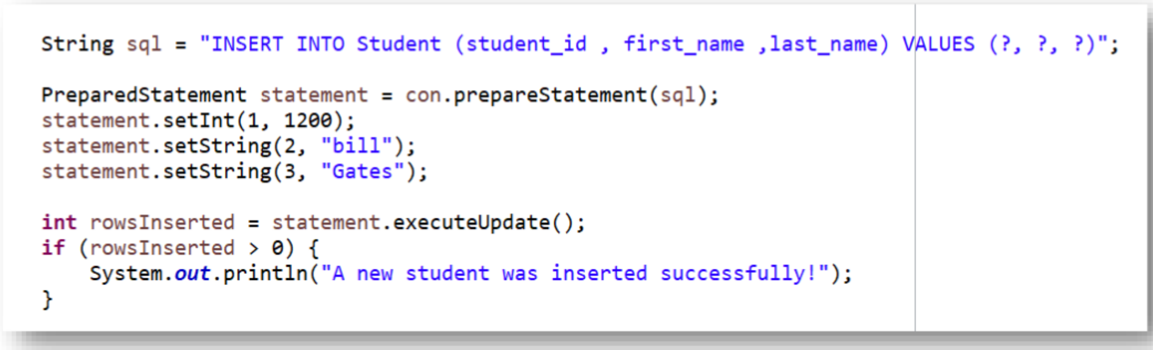
```java
public static void main(String args[]) {
    try
    {
    //step1 load the driver class
    Class.forName("oracle.jdbc.driver.OracleDriver");

    System.out.println("Driver Loaded Successfully!");

    //step2 create  the connection object
    Connection con=DriverManager.getConnection(
    "jdbc:oracle:thin:@192.168.1.8:1521:xe","system","tiger12345");

    System.out.println("Connection Established!");

    //step3 create the statement object
    Statement stmt=con.createStatement();

    //step4 execute query
    ResultSet rs=stmt.executeQuery("select * from STUDENT");

    while(rs.next())
    {
    int id = rs.getInt(1);
    String firstName = rs.getString("first_name"); // by column name matching
    String lastName = rs.getString("last_name");

    System.out.println(id+"  "+firstName+"  "+lastName);
    }
    //step5 close the connection object
    con.close();
```



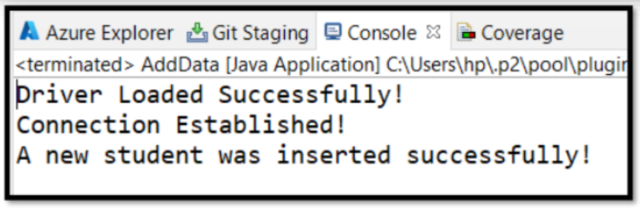
<terminated> OracleCon [Java Application] C:\Users\hp\.p2\pool\

```
Driver Loaded Successfully!
Connection Established!
5000  Sara  Khan
6000  Zara  Hassan
8000  Muhammad  Ali
```

INSERT Statement

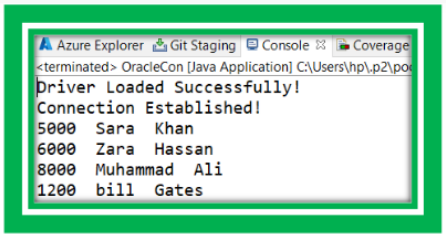


```java
String sql = "INSERT INTO Student (student_id , first_name ,last_name) VALUES (?, ?, ?)";

PreparedStatement statement = con.prepareStatement(sql);
statement.setInt(1, 1200);
statement.setString(2, "bill");
statement.setString(3, "Gates");

int rowsInserted = statement.executeUpdate();
if (rowsInserted > 0) {
    System.out.println("A new student was inserted successfully!");
}
```



<terminated> AddData [Java Application] C:\Users\hp\.p2\pool\plugir

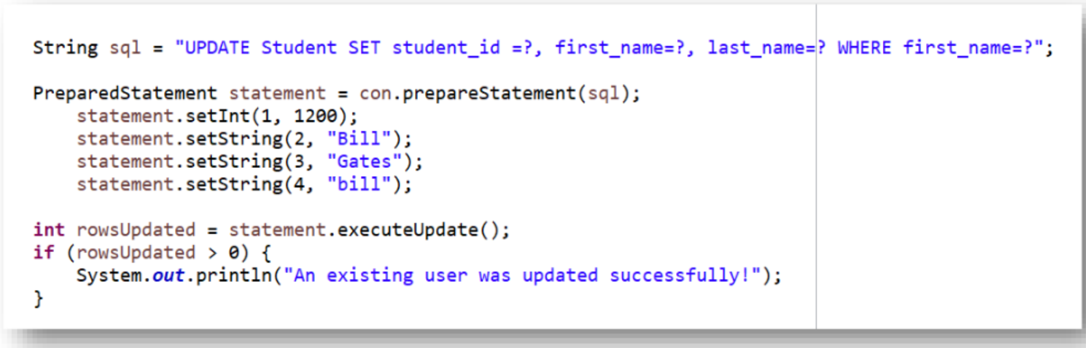
```
Driver Loaded Successfully!
Connection Established!
A new student was inserted successfully!
```



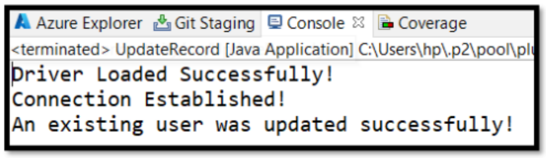
<terminated> OracleCon [Java Application] C:\Users\hp\.p2\poo

```
Driver Loaded Successfully!
Connection Established!
5000  Sara  Khan
6000  Zara  Hassan
8000  Muhammad  Ali
1200  bill  Gates
```

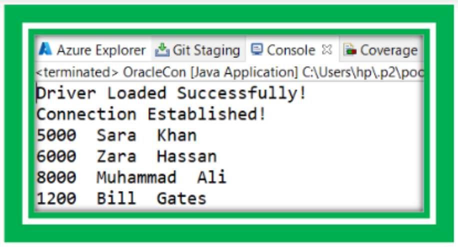
UPDATE Statement



```java
String sql = "UPDATE Student SET student_id =?, first_name=?, last_name=? WHERE first_name=?";

PreparedStatement statement = con.prepareStatement(sql);
    statement.setInt(1, 1200);
    statement.setString(2, "Bill");
    statement.setString(3, "Gates");
    statement.setString(4, "bill");

int rowsUpdated = statement.executeUpdate();
if (rowsUpdated > 0) {
    System.out.println("An existing user was updated successfully!");
}
```

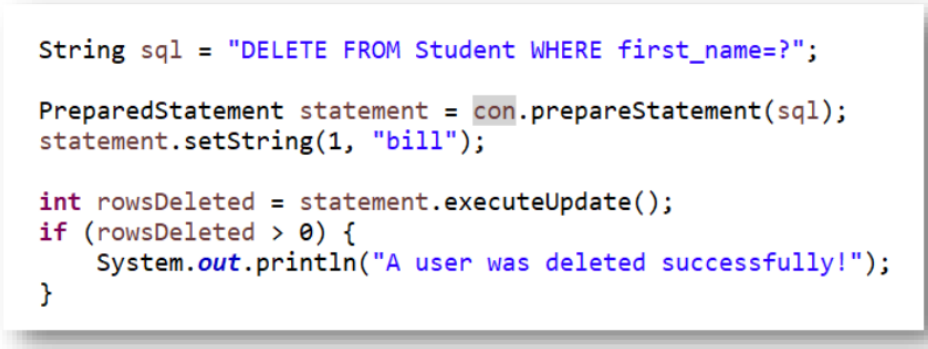


```
<terminated> UpdateRecord [Java Application] C:\Users\hp\.p2\pool\pl
Driver Loaded Successfully!
Connection Established!
An existing user was updated successfully!
```

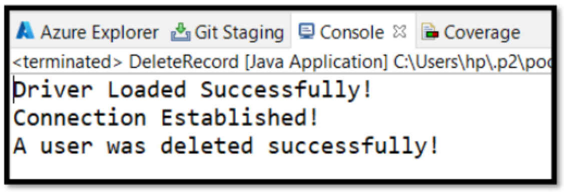


```
<terminated> OracleCon [Java Application] C:\Users\hp\.p2\poo
Driver Loaded Successfully!
Connection Established!
5000  Sara  Khan
6000  Zara  Hassan
8000  Muhammad  Ali
1200  Bill  Gates
```

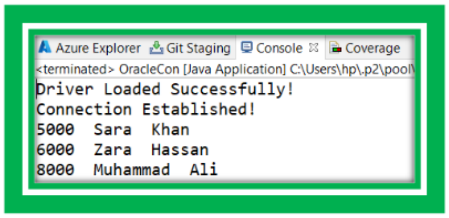
DELETE Statement



```java
String sql = "DELETE FROM Student WHERE first_name=?";

PreparedStatement statement = con.prepareStatement(sql);
statement.setString(1, "bill");

int rowsDeleted = statement.executeUpdate();
if (rowsDeleted > 0) {
    System.out.println("A user was deleted successfully!");
}
```



```
<terminated> DeleteRecord [Java Application] C:\Users\hp\.p2\poo
Driver Loaded Successfully!
Connection Established!
A user was deleted successfully!
```



```
<terminated> OracleCon [Java Application] C:\Users\hp\.p2\pool\
Driver Loaded Successfully!
Connection Established!
5000  Sara  Khan
6000  Zara  Hassan
8000  Muhammad  Ali
```

Making a DB Handler:

```
Class Student{
PersitenceHandler persHandler;
void save(){
    persHandler.saveStudent(this);
}
void setPersitenceHandler (PersitenceHandler ph)
{
    this.persHandler=ph;
}
```

```
// PersistenceHandler
Class PersistenceHandler{
abstract void saveStudent(Student s);
}

class OracleDBHandler extends PersistenceHandler{
    void saveStudent(Student s){
        //connection
        //insert query formulation
        //execute query
    }
}
class SQLHandler extends PersistenceHandler{
    void saveStudent(Student s){
        //connection
        //insert query formulation
        //execute query
    }
}

Void main()
{
    PersitenceHandler handler= new OracleHandler();
    University uni= new University();
    Uni. setPersitenceHandler(handler);
}
```