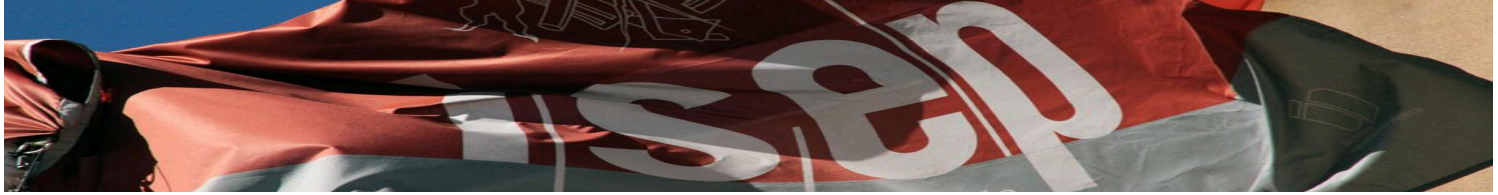
BASE DE DADOS

Oracle/Java

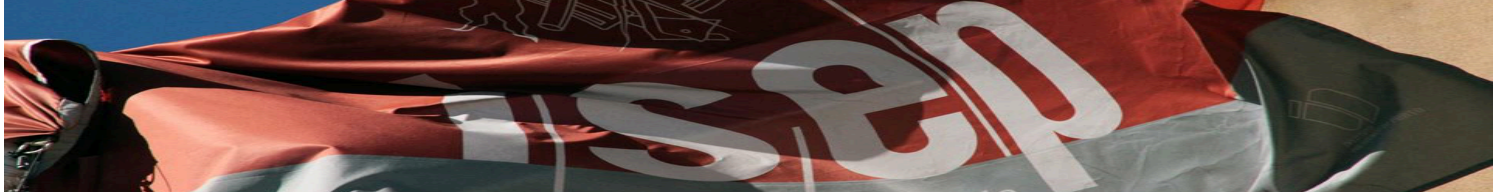
Teórico-Práticas
Ano Lectivo 2016/2017
Rosa Reis

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA



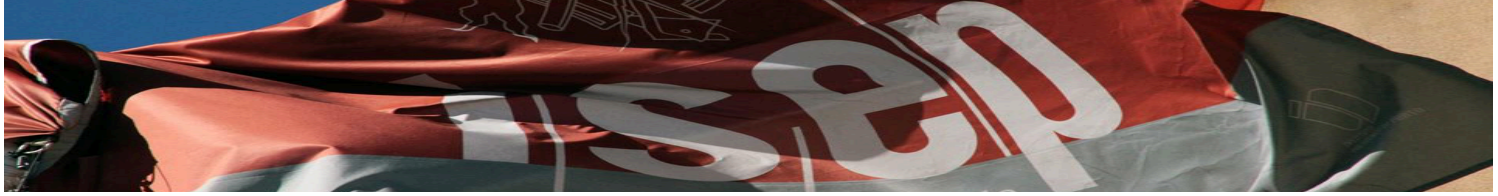
Objetivos

1. INTRODUÇÃO
2. MODELO DE COMUNICAÇAO
3. OBJECTO STATEMENT AND PREPAREDSTATEMENT
4. EXECUÇÃO DE CONSULTAS
5. EXCEPÇÕES

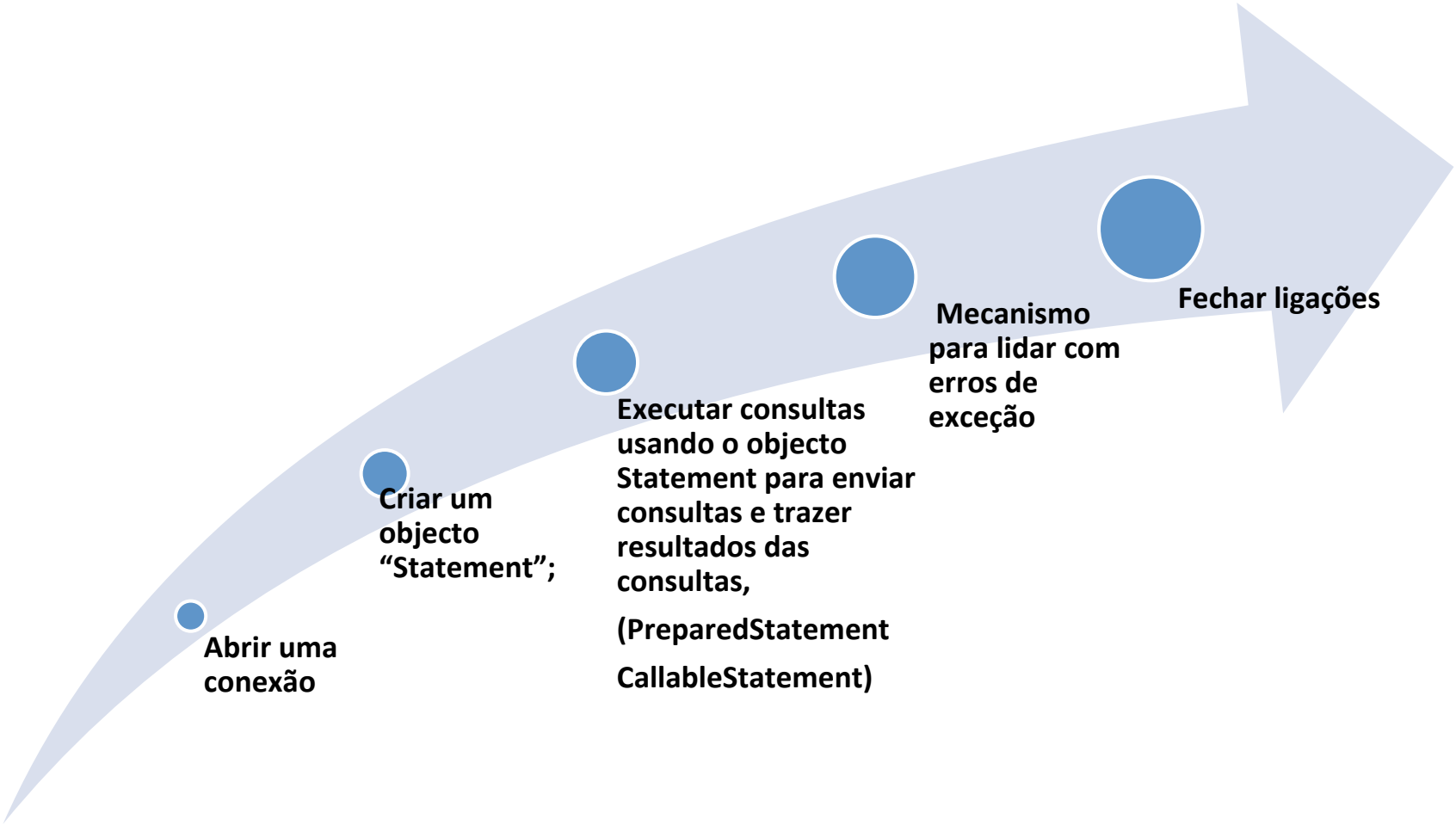


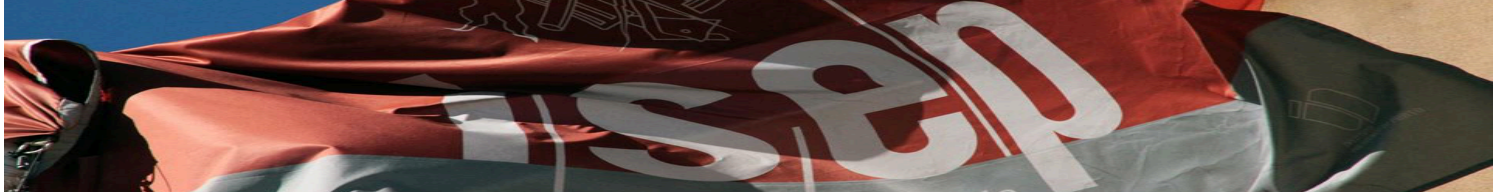
INTRODUÇÃO

- As aplicações desenvolvidas em Java acedem à base de dados Oracle através do JDBC (Java Database Connectivity Driver).
- Usam Oracle Net para comunicar com a Base de Dados
- JDBC é a Api que permite a comunicação com a Base de Dados e suporta SQL.
- Suporta uma variedade de recursos para consultar e atualizar dados, e para recuperar resultados da consulta.
- Também, permite a recuperação de metadados, tais como a consulta sobre as relações presentes na BD e os nomes e tipos de atributos de relação.
- É necessário fazer o download do Oracle JDBC driver:
  http://www.oracle.com/technetwork/apps-tech/jdbc-10201-088211.html
- Para instalar basta fazer o download e colocar ojdbcxx.jar no caminho de classe



MODELO DE COMUNICAÇÃO





ABRIR CONEXÃO

- Para efetuar a conexão é necessário:
  - **Registar o driver**

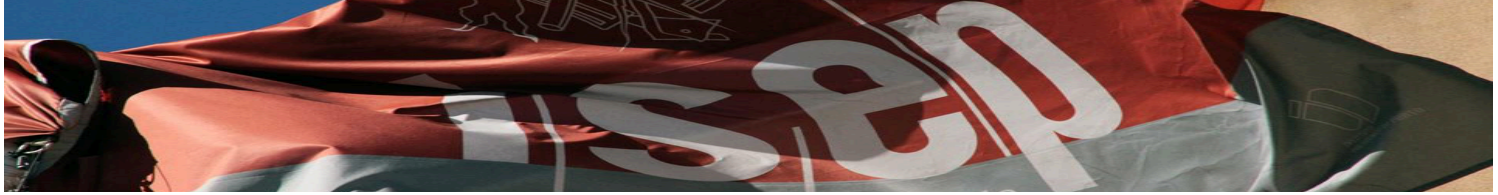
    Para que um SGBD seja utilizado via JDBC é necessário que o mesmo tenha um driver JDBC específico, que é carregado durante a execução da aplicação.

```
// Load Oracle JDBC Driver
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

  - **Conectar à BD**

    O JDBC usa um URL para identificar a conexão à Base de dad

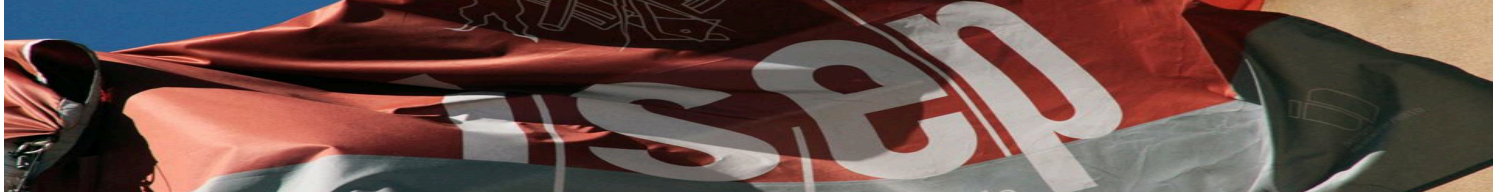
```
jdbc: oracle:<driver>:@<database>
```



ABRIR CONEXÃO

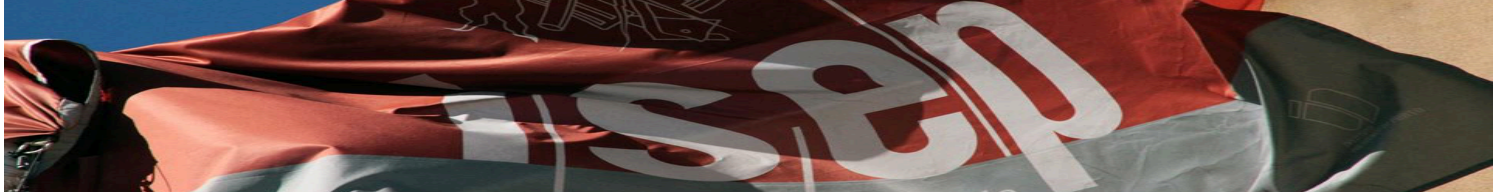
- Criar um data source usando a classe OracleDataSource.
- Instanciar um objeto da classe Connection usando o metodo getConnection do data source

```
...
// estabelece a liação à BD.

    */
.....

      OracleDataSource ds = new OracleDataSource();
      ds.setURL("jdbc:oracle:thin://@gandalf.dei.isep.ipp.pt:1521/pdborcl");
      ds.setUser(“rosa");
      ds.setPassword(”-------”);
      Connection conn= ds.getConnection();
       ........
      ds.close();
```



JDBC CODE

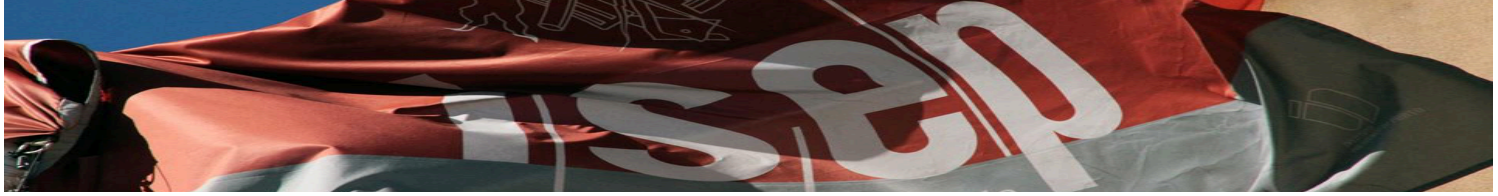
```
public static void JDBCexample(String dbid, String userid, String passwd)
 {
    try {
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                  "jdbc:oracle:thin://@gandalf.dei.isep.ipp.pt:1521/pdborcl", userid,
          passwd);
           …..
            … Do Actual Work ….

        con.close();
    }
    catch (SQLException sqle) {
        System.out.println("SQLException : " + sqle);
    }
 }
```



CRIAR UM OBJECTO STATMENT

- O objeto Statement é utilizado para a execução de comandos SQL através do driver da base de dados. Os comandos SQL são passados ao objeto Statement como strings.
- Suporta os métodos
  - **executeUpdate():**usado para operações que causam alteração na base de dados (create, update, insert, etc.)
  - **executeQuery():** usado em operações de consulta à base de dados.

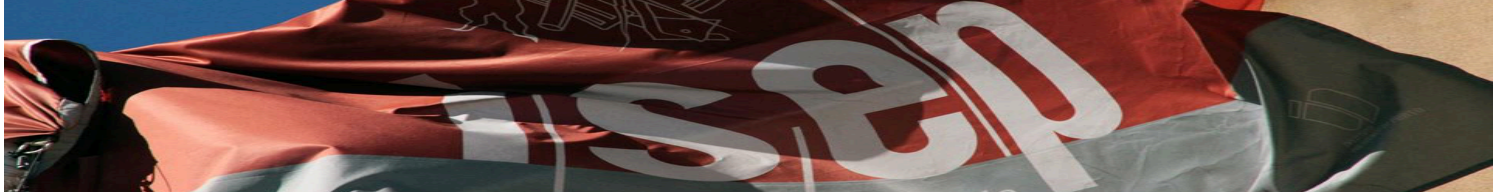
```
...
String insertString = "insert into Alunos values”  +
                          ”(01,’José da Silva’,’44.444.444’)”;
try {
        Statement stmt = con.createStatement();
        stmt.executeUpdate(insertString);
         stmt.close();
} catch(SQLException ex) { ... }
```



PREPAREDSTATEMENT

- Prepared Statement é utilizado para consultas que são executadas várias vezes.
- São interpretados pelo DBMS apenas uma vez (pré-compilados)
- Os valores da coluna podem ser definidos após a compilação.
- Em vez de valores, usa-se a notação '?'

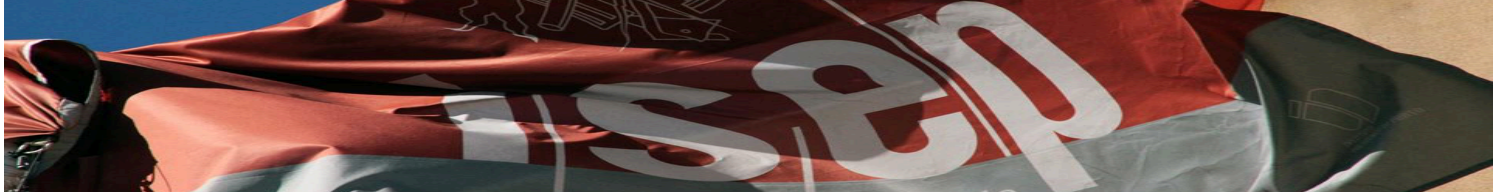
```
Connection con = DriverManager.getConnection(localBD);
String sql = "UPDATE aluno SET nota = ? WHERE codigo = ?";
PreparedStatement ps = con.preparedStatement(sql);
ps.setInt(1, 10);
ps.setInt(2, 1001);
/* Atualiza o aluno 1001 com a nota 10 */
int result = ps.executeUpdate();
```



EXECUTAR CONSULTAS

- Um objeto CallableStatement detém parâmetros para chamar procedimentos armazenados.
- Um objeto CallableStatement pode conter variáveis que são fornecidas cada vez que se executa a instrução.
- Quando o procedimento armazenado retorna valores calculados, eles são recuperadas por meio do objeto CallableStatement

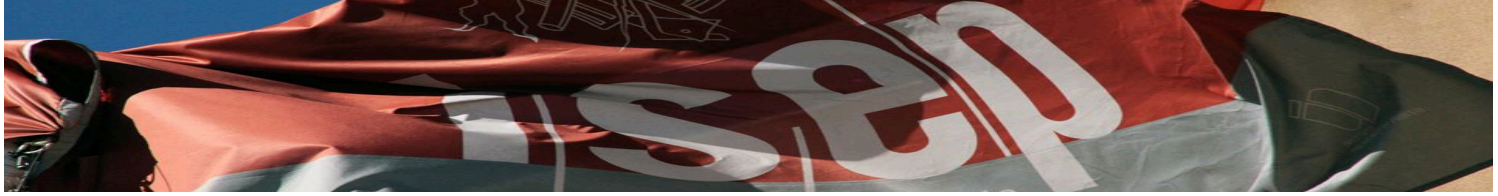
```
CallableStatement resultado = con.prepareCall("{?= call factorial (?)}");

resultado.registerOutParameter(1, Types.FLOAT);
resultado.setInt(2, n1);
```



EXECUTAR CONSULTAS

- O JDBC retorna os resultados de uma consulta num objecto chamado de **ResultSet**
- O objeto **ResultSet** é uma estrutura de objetos que mantém os dados resultantes de uma consulta à base de dados.
- Além dos dados relativos à consulta, o objeto **ResultSe**t contém metadados que descrevem o resultado obtido.
- O método executeQuery(), do objeto Statement retorna um objeto ResultSet.

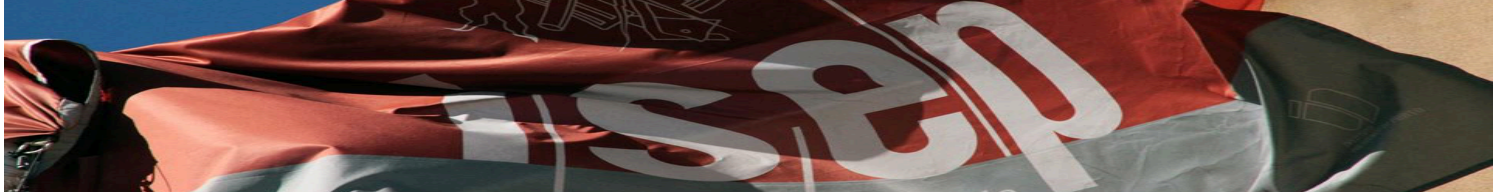
```
try {
        Statement stmt = con.createStatement();
         ResultSet rs = statement.executeQuery("SELECT * FROM casas order by numc ");

        ..................

        con.close();
}catch(Exception ex) { ... }
```



EXECUTAR CONSULTAS

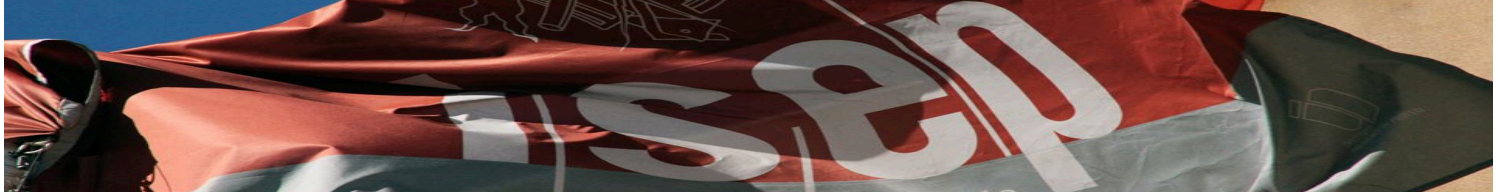
- A navegação pelo objeto ResultSet é feita através do **método next()**. Outros métodos disponíveis:
  - previous()
  - first()
  - last()

- O acesso às colunas de uma linha do ResultSet é feito através dos métodos
  - **getInt(nomeDaColuna)**
  - **getString(nomeDaColuna)**
  - Outros métodos disponíveis para outros tipos de dados Java 'mapeáveis' para os tipos SQL.
    - **rs.getString("dept_name") and rs.getString(1) é equivalent se é o primeiro argumento do resultado do select**



MAPEANDO TIPOS DE JAVA PARA SQL

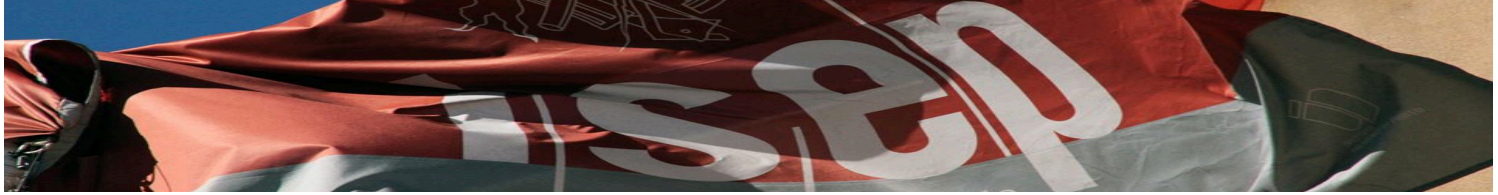
SQL	Java
CHAR, VARCHAR, LONGVARCHAR	String
NUMERIC, DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT, DOUBLE	double
BINARY, VARBINARY, LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp



EXECUTAR CONSULTAS

Exemplo

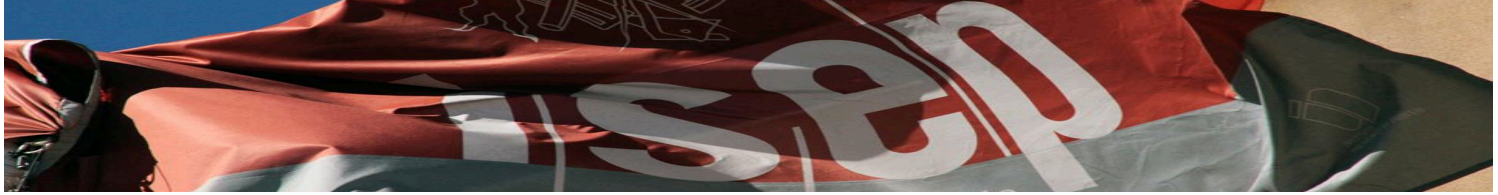
```
ResultSet rs = statement.executeQuery( "SELECT * FROM casas order by
numc ");

        while (rs.next()) {
                int codigo = rs.getInt("NUMC");
                String zona1 = rs.getString("ZONA");
                 int preço = rs.getInt("PREÇO");
                 int Nrcliente = rs.getInt("NumCliente");
                System.out.println(  "\t" + codigo + "\t" + zona1 + "\t" + preço +
                        "\t" + Nrcliente);
            }
```



EXECUTAR CONSULTAS

```
int n = 10;
CallableStatement resultado = con.prepareCall("{?= call factorial (?)}");
resultado.registerOutParameter(1, Types.FLOAT);
resultado.setInt(2, n);
resultado.executeUpdate();
float fatorial = resultado.getFloat(1);
System.out.println("O factorial de " + n +" é: "+ fatorial + " ");
```

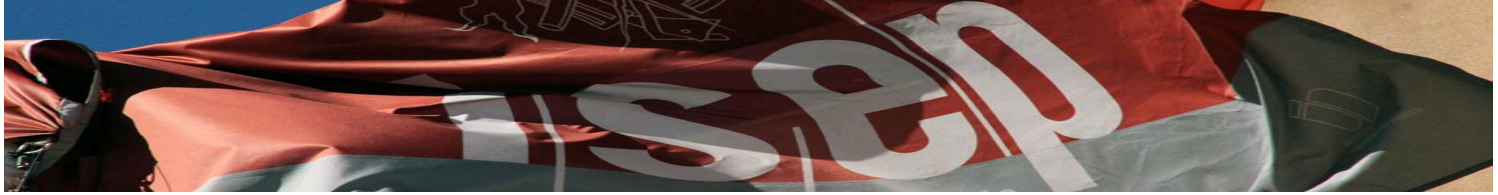
```
create or replace function factorial
( n in number) return float
is
   fact float;
   i number(2);
begin
        fact:=1;
        for i in 1..n loop
          fact:=fact*I;
        end loop;
return(fact);
end;
```



EXECUTAR PROCEDIMENTOS

```
System.out.print("\n\n PROCEDIMENTO FACTORIAL - OUT\n");

nn1 = 10;

  CallableStatement resultado1 = con.prepareCall("{call factor (?,?)}");

resultado1.setInt(1, nn1);
resultado1.registerOutParameter(2, OracleTypes.FLOAT);

resultado1.executeUpdate();

float resfat = resultado1.getFloat(2);

System.out.println( "n\n PROCEDIMENTO COM PARAMETROS OUT
- O factorial Procedimento de " + nn1 +" é:" + resfat);
        resultado1.close();
```

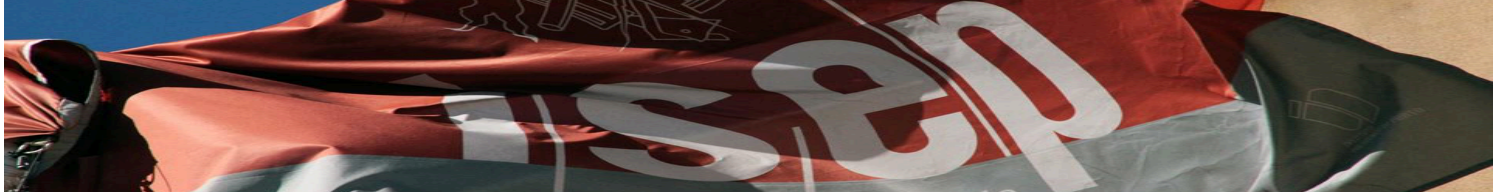
```
create or replace procedure
factor ( n in number, f out float) is
   i number(3);
begin
  f:=1;
 for i in 1..n loop
     f:=f*i;
 end loop;
end;
```



EXECUTAR PROCEDIMENTOS

```
CallableStatement cs = con.prepareCall ("{call BUSCA_CASAS
(?,?)}");

cs.setString(1, "Norte");
cs.registerOutParameter(2,OracleTypes.CURSOR);
cs.executeUpdate();

//Recuperando o cursor como um Resultset
ResultSet cs1 = (ResultSet)cs.getObject(2);

while(cs1.next()){
  //Obtendo o valor das colunas
  System.out.println(”Numero da Casa" + cs1.getInt("NUMC"))
  System.out.println("Zona: " + cs1.getString("ZONA"));
  System.out.println("Preço: " + cs1.getInt("PREÇO"));
 System.out.println(”Numero cliente" +
            cs1.getInt("NUMCLIENTE"));
 }
```

```
create or replace PROCEDURE BUSCA_CASAS
(nzona  casas.zona%TYPE, RETORNO OUT
SYS_REFCURSOR )
AS
BEGIN

 OPEN RETORNO FOR
   SELECT numc, zona, preço, numcliente
   FROM casas
   WHERE zona= nzona;
END;
```



EXCEPTIONS

- Uma exceção pode englobar outras exceções:

```
Catch (SQLException e) {
  while (e != null {
            system.outprintln(e.getSQLState());
            system.outprintln(e.getMessage());
      system.outprintln(e.getErrorCode());
            e=e.getNextException();
}
}
```

getMessage() -String descrevendo o erro
SQLState - String no Padrão XOPEN SQL
getErrorCode() -Código de erro específico do fabricante
getNextException() -Link para próxima exceção, contendo informação adicional