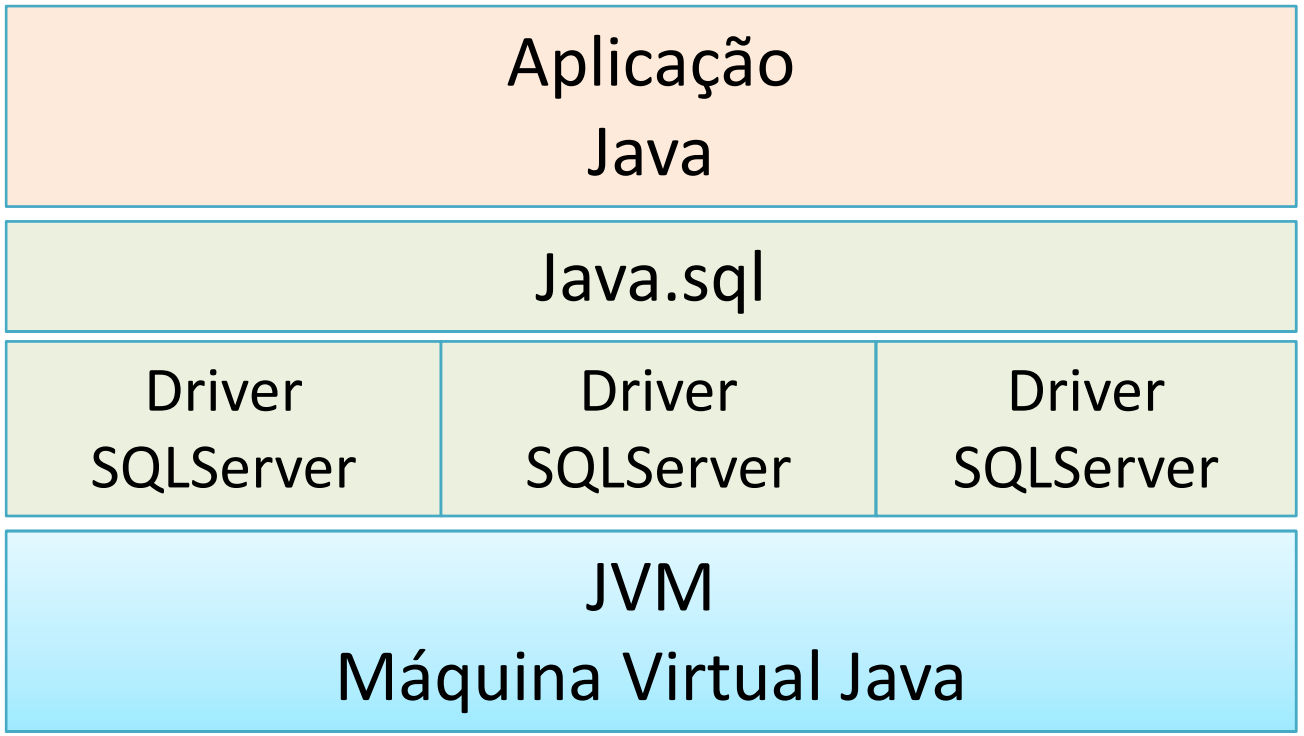
# Curso de Java

## Acesso a banco de dados através de JDBC


- SQL é a linguagem padrão para acesso a banco de dados e se estabeleceu como tal no mercado.
- SQL no entanto não é adequada para o desenvolvimento de aplicações e na prática é dependente de SGDB e de plataforma.
- Java, assim como outras linguagens de programação oferece mecanismos para acesso a bases de dados através de SQL.
- Em Java, esse mecanismo é chamado de JDBC: *Java DataBase Connectivity*.
- O mecanismo JDBC segue as premissas da linguagem Java de independência de plataforma e se propõe a ser independente do SGBD utilizado.


- O acesso a uma base de dados através de JDBC se baseia em dois elementos:
  - O pacote java.sql que contém as classes que permitem o acesso ao banco de dados pela aplicação.
  - O *driver* para acesso ao banco de dados a ser utilizado.

*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. A maioria dos SGBD's oferece drivers para JDBC.*


- O driver para o SGBD em uso deve ser carregado dinamicamente pela aplicação. O código abaixo apresenta o padrão de usado para carregar o driver.

```
...
/** nome da classe que contém o driver a ser utilizado **/
static String driverName = "org.postgresql.Driver";

...
try {
     Class.forName(driverName);//

} catch(java.lang.ClassNotFoundException e) {
     System.err.print("ClassNotFoundException: ");
     ...
}
```

# O objeto Connection


- O objeto Connection é utilizado para 'conectar' a aplicação a uma base de dados. Uma sessão típica de acesso a uma base de dados segue o padrão abaixo.

```
...
// estabelecimento da conexão
Connection con = DriverManager.getConnection(
                        url,       // caminho de acesso ao SGBD
                        userid,    // login
                        password   // senha
                    );
...
// uso da conexão p/ acesso ao SGBD (select, update,etc)
// todo acesso ao SGBD se baseia no objeto conexão.
...
...
// encerramento da conexão
con.close();
```


- 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.

```
...
String createString = "create table Alunos ("       +
                      "Aluno_ID INTEGER,"           +
                      "Aluno_nome VARCHAR(30),"     +
                      "Aluno_RG VARCHAR(15)"        +
                     ")";
...
try {
     Statement stmt = con.createStatement();
     stmt.executeUpdate(createString);
     stmt.close();
} catch(SQLException ex) { ... }
```


- O objeto Statement 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.

# O objeto Statement - outro exemplo


- Neste exemplo o objeto Statement é utilizado para inserção de dados na 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(Exception ex){  … }
```

# O objeto 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 ResultSet contém metadados que descrevem o resultado obtido.
- O método executeQuery(), do objeto Statement retorna um objeto ResultSet.

```
String selectString = "select * from Alunos";
try{
     Statement stmt = con.createStatement();
     ResultSet rs = stmt.executeQuery(selectString);
     while (rs.next()) {
        int id      = rs.getInt("Aluno_ID");
        String nome = rs.getString("Aluno_nome");
        String rg   = rs.getString("Aluno_RG");
        System.out.println("id:"+id+" nome:"+nome+" rg:"+rg);
     }
     con.close();
 }catch(Exception ex) { ... }
```

# O objeto ResultSet


- No exemplo anterior, 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.

# *Os tipos possíveis de ResultSet*


- Um objeto ResultSet pode ser de um dos tipos abaixo. Cada um desses tipos oferecem diferentes recursos de 'navegação' sobre o resultado da consulta.
  - TYPE_FORWARD_ONLY
  - TYPE_SCROLL_INSENSITIVE
  - TYPE_SCROLL_SENSITIVE
- O tipo do ResultSet é definido ao se criar o objeto Statement.

```
...
Statement stmt = com.createStatement(
                                        ResultSet.TYPE_FORWARD_ONLY,
                                        ResultSet.CONCUR_READ_ONLY
                                      );
...
```

# O objeto PreparedStatement


- Um objeto PreparedStatement contém um comando SQL 'parametrizável'.
- Os parâmetros aparecem no comando como '?' e são substituídos conforme sua posição no comando.

```
String updateString =
          "update Alunos set Aluno_nome = ? where Aluno_Id  = ?";
try {
     PreparedStatement pstmt = con.prepareStatement (updateString);

     pstmt.setInt(2, id);//os índices começam com 1
     pstmt.setString(1, nome);

     int k = pstmt.executeUpdate();

     pstmt.close();

} catch(SQLException ex) {  … }
```


- JDBC cumpre seus objetivos no sentido de prover um mecanismo de acesso a bases de dados independente de plataforma e de SGBD.
- Os problemas de incompatibilidade que em geral ocorrem se devem a diferenças na implementação da linguagem SQL entre os diversos SGBD.
- Como Java é orientada a objetos e SQL é voltada a 'operações relacionais', existe um conflito que pode ser problema às aplicações.
- Atualmente existe uma tendência crescente no uso de ambientes voltados ao mapeamento objeto-relacional, como por exemplo Hibernate e JPA.

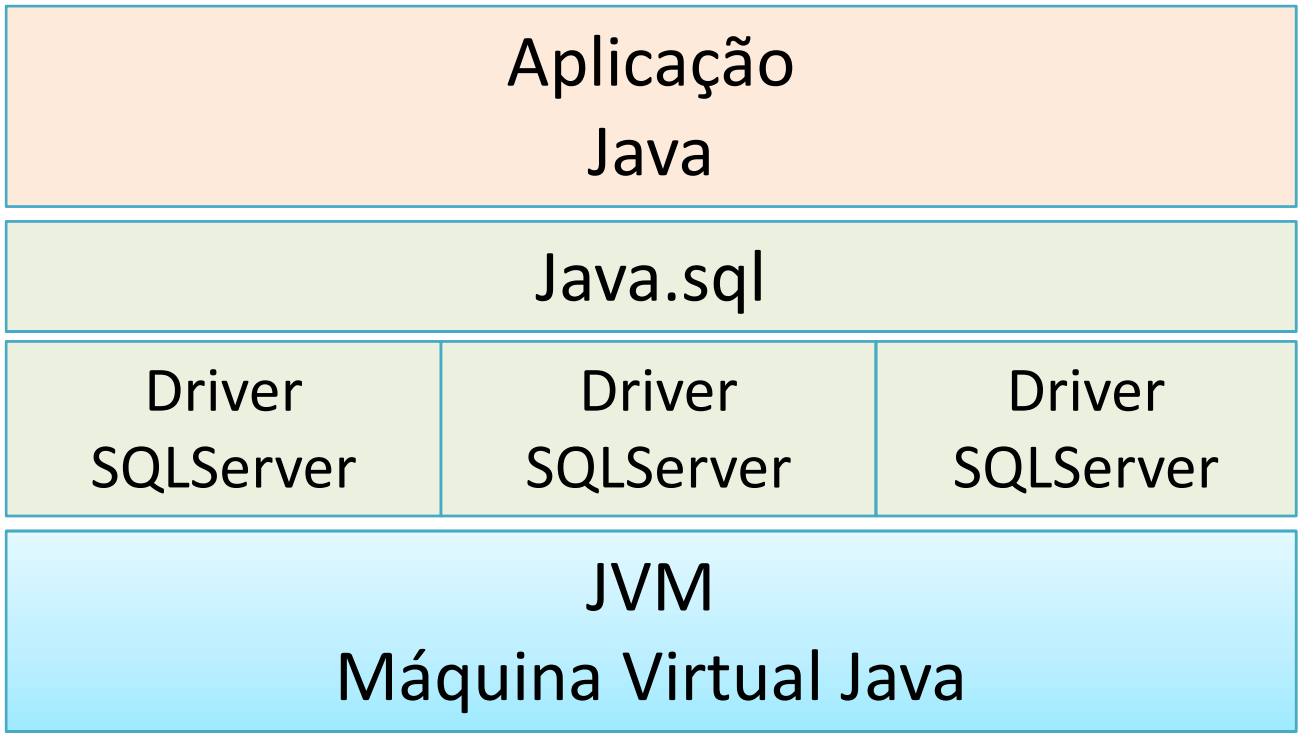
# Curso de Java

## Acesso a banco de dados através de JDBC


- SQL é a linguagem padrão para acesso a banco de dados e se estabeleceu como tal no mercado.
- SQL no entanto não é adequada para o desenvolvimento de aplicações e na prática é dependente de SGDB e de plataforma.
- Java, assim como outras linguagens de programação oferece mecanismos para acesso a bases de dados através de SQL.
- Em Java, esse mecanismo é chamado de JDBC: *Java DataBase Connectivity*.
- O mecanismo JDBC segue as premissas da linguagem Java de independência de plataforma e se propõe a ser independente do SGBD utilizado.


- O acesso a uma base de dados através de JDBC se baseia em dois elementos:
  - O pacote java.sql que contém as classes que permitem o acesso ao banco de dados pela aplicação.
  - O *driver* para acesso ao banco de dados a ser utilizado.

*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. A maioria dos SGBD's oferece drivers para JDBC.*


- O driver para o SGBD em uso deve ser carregado dinamicamente pela aplicação. O código abaixo apresenta o padrão de usado para carregar o driver.

```
...
/** nome da classe que contém o driver a ser utilizado **/
static String driverName = "org.postgresql.Driver";

...
try {
     Class.forName(driverName);//

} catch(java.lang.ClassNotFoundException e) {
     System.err.print("ClassNotFoundException: ");
     ...
}
```

# O objeto Connection


- O objeto Connection é utilizado para 'conectar' a aplicação a uma base de dados. Uma sessão típica de acesso a uma base de dados segue o padrão abaixo.

```
...
// estabelecimento da conexão
Connection con = DriverManager.getConnection(
                        url,       // caminho de acesso ao SGBD
                        userid,    // login
                        password   // senha
                    );
...
// uso da conexão p/ acesso ao SGBD (select, update,etc)
// todo acesso ao SGBD se baseia no objeto conexão.
...
...
// encerramento da conexão
con.close();
```


- 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.

```
...
String createString = "create table Alunos ("       +
                      "Aluno_ID INTEGER,"           +
                      "Aluno_nome VARCHAR(30),"     +
                      "Aluno_RG VARCHAR(15)"        +
                     ")";
...
try {
     Statement stmt = con.createStatement();
     stmt.executeUpdate(createString);
     stmt.close();
} catch(SQLException ex) { ... }
```


- O objeto Statement 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.

# O objeto Statement - outro exemplo


- Neste exemplo o objeto Statement é utilizado para inserção de dados na 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(Exception ex){  … }
```

# O objeto 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 ResultSet contém metadados que descrevem o resultado obtido.
- O método executeQuery(), do objeto Statement retorna um objeto ResultSet.

```
String selectString = "select * from Alunos";
try{
     Statement stmt = con.createStatement();
     ResultSet rs = stmt.executeQuery(selectString);
     while (rs.next()) {
        int id      = rs.getInt("Aluno_ID");
        String nome = rs.getString("Aluno_nome");
        String rg   = rs.getString("Aluno_RG");
        System.out.println("id:"+id+" nome:"+nome+" rg:"+rg);
     }
     con.close();
 }catch(Exception ex) { ... }
```

# O objeto ResultSet


- No exemplo anterior, 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.

# *Os tipos possíveis de ResultSet*


- Um objeto ResultSet pode ser de um dos tipos abaixo. Cada um desses tipos oferecem diferentes recursos de 'navegação' sobre o resultado da consulta.
  - TYPE_FORWARD_ONLY
  - TYPE_SCROLL_INSENSITIVE
  - TYPE_SCROLL_SENSITIVE
- O tipo do ResultSet é definido ao se criar o objeto Statement.

```
...
Statement stmt = com.createStatement(
                                        ResultSet.TYPE_FORWARD_ONLY,
                                        ResultSet.CONCUR_READ_ONLY
                                      );
...
```

# O objeto PreparedStatement


- Um objeto PreparedStatement contém um comando SQL 'parametrizável'.
- Os parâmetros aparecem no comando como '?' e são substituídos conforme sua posição no comando.

```
String updateString =
          "update Alunos set Aluno_nome = ? where Aluno_Id  = ?";
try {
     PreparedStatement pstmt = con.prepareStatement (updateString);

     pstmt.setInt(2, id);//os índices começam com 1
     pstmt.setString(1, nome);

     int k = pstmt.executeUpdate();

     pstmt.close();

} catch(SQLException ex) {  … }
```


- JDBC cumpre seus objetivos no sentido de prover um mecanismo de acesso a bases de dados independente de plataforma e de SGBD.
- Os problemas de incompatibilidade que em geral ocorrem se devem a diferenças na implementação da linguagem SQL entre os diversos SGBD.
- Como Java é orientada a objetos e SQL é voltada a 'operações relacionais', existe um conflito que pode ser problema às aplicações.
- Atualmente existe uma tendência crescente no uso de ambientes voltados ao mapeamento objeto-relacional, como por exemplo Hibernate e JPA.