Programação Orientada a Objetos II

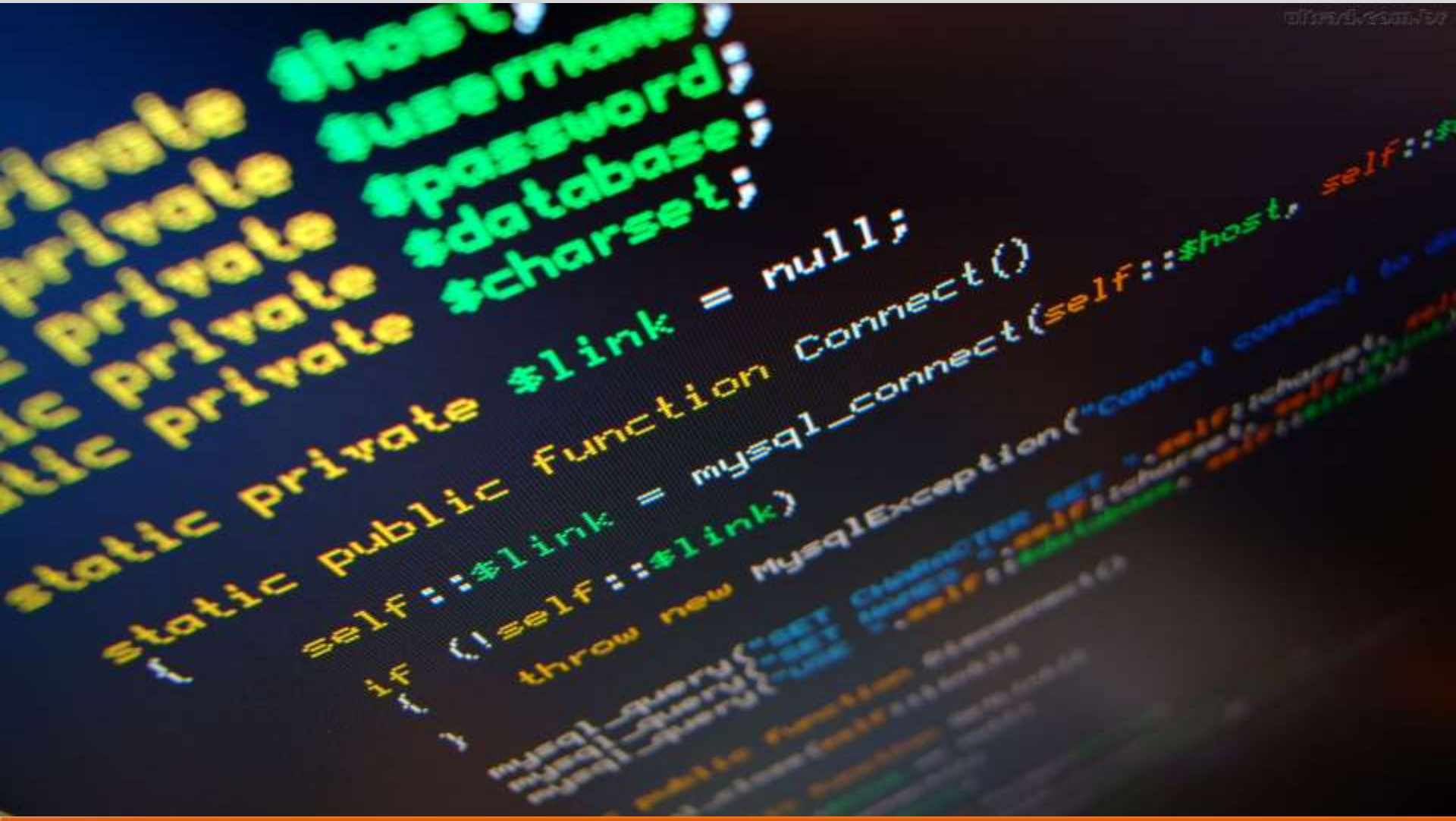
Prof. Walter Gima

walter.gima@anhanguera.com

Plano de Ensino e Aprendizagem

Cronograma de Aulas	
Semana n°.	**Tema**
1	Apresentação da Disciplina e Metodologia de Trabalho. Revisão de Orientação à Objetos.
2	Introdução ao Desenvolvimento de Interfaces Gráficas (Bibliotecas AWT e Swing, Containers Swing, Botões, Caixas de Texto e Rótulos).
3	Introdução ao Desenvolvimento de Interfaces Gráficas (Gerenciadores de Layout, Tratamento de Eventos).
4	Introdução ao Desenvolvimento de Interfaces Gráficas (Tratamento de Eventos).
5	Revisão (Herança e Polimorfismo). Conceito de Herança Múltipla.
6	Polimorfismo (Chamadas de Métodos Polimórficas, Passagem de Parâmetros Polimórficos).
7	Classe Abstrata (Definição de Métodos Abstratos, Implementação de Classes Abstratas).
8	Classe Abstrata (Definição de Métodos Abstratos, Implementação de Classes Abstratas).
9	Atividades de Avaliação.
10	Interfaces (definição de Contratos de Métodos, Implementação de Interfaces).
11	Interfaces (Implementação de Interfaces, Herança Múltipla Através de Interfaces).
12	Interfaces (Implementação de Interfaces, Herança Múltipla Através de Interfaces).
13	Tratamento de Exceções (Definição dos Mecanismos de Exceções, Exceções Verificadas e Não Verificadas).
14	Tratamento de Exceções (Captura e Tratamento de Exceções, Definição de Novos Tipos de Exceções).
15	Arquivos
16	Arquivos
17	Coleções
18	Prova Escrita Oficial
19	Exercícios de Revisão.
20	Prova Substitutiva

Java – Conexão banco de dados



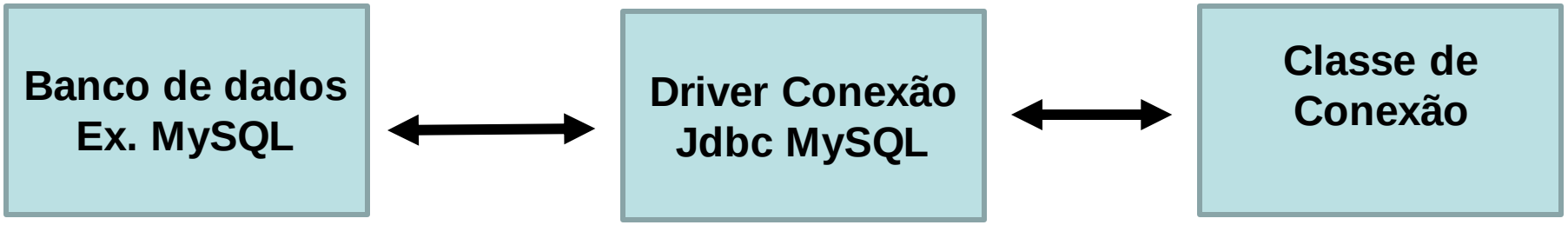
Anhanguera

Agenda

- Conexão Banco de Dados
- Padrão DAO (Data Access Object)

Conexão Banco de dados

Para estabelecer conexão entre Java com um banco de dados:



Driver de Conexão com Banco de dados

- O driver de conexão é um biblioteca responsável por realizar a comunicação com o banco de dados

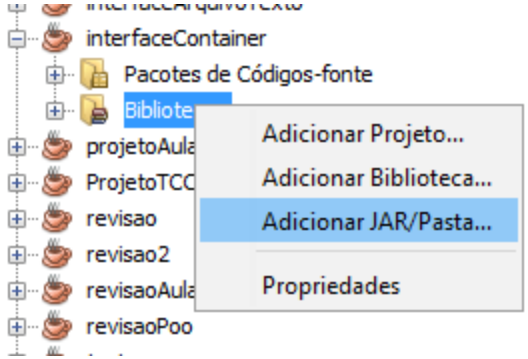
- Existe um drive especifico para cada banco de dados. Ex. Se o banco de dados utilizado é Oracle é necessário utilizar o driver de conexão JDBC Oracle, caso o banco de dados seja MySQL é preciso usar o driver de conexão JDBC MySQL.

- Geralmente é possível fazer o download dos drivers JDBC nos sites dos fabricantes de banco de dados.
  Ex. http://dev.mysql.com/downloads/connector/j/5.1.html

Driver de Conexão com Banco de dados

Para adicionar um driver JDBC em um projeto Java utilizando NETBEANS:

- Realizar o download do driver JDBC. Ex. driver do MySQL.
- Criar uma pasta chamada biblioteca e copiar o driver para essa pasta.
- Procure a sessão bibliotecas no projeto Netbeans, clique com o botão direito do mouse e escolha a opção Adicionar JAR/Bibliotecas.



- Selecione o driver (arquivo.JAR) que foi copiado na pasta biblioteca dentro do projeto.

Classe de Conexão com Banco de dados

- A classe de conexão utiliza o driver para estabelecer a conexão com o banco de dados.

- A classe de conexão é responsável também por prover as classes e métodos necessários para enviar comandos SQL para o banco de dados Ex. realizar INSERT, DELETE e UPDATE.

- A classe de conexão é responsável também por prover classes e métodos para realizar uma consulta e administrar os registros dessa consulta.

Classe de Conexão com Banco de dados

Ex. Classe de conexão com o MySQL

```
public class ConexaoMySql {

    static String status= "";

    public static Connection getConexao() {
        Connection conn = null;

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();

            String url = "jdbc:mysql://localhost/projetoaula4?user=root&password=";
            conn = DriverManager.getConnection(url);
            status = "conectado";
            System.out.println("Conectado.");
        } catch (ClassNotFoundException e) {
            //Driver não encontrado
            System.out.println("O driver expecificado nao foi encontrado.");
        } catch (SQLException e) {
            //Não conseguindo se conectar ao banco
            System.out.println("Nao foi possivel conectar ao Banco de Dados.");
        } catch (Exception e) {
            //Não conseguindo se conectar ao banco
            System.out.println("Nao foi possivel conectar ao Banco de Dados.");
        }
        return conn;
    }
}
```

Informações da string de conexão:

IP onde está o banco de dados: ex. localhost

Nome do banco de dados: ex. projetoaula4

Usuário de acesso ao banco: ex. root

Senha de acesso ao banco: ex. sem senha

Anhanguera

Classe de Conexão com Banco de dados

- Realizar a conexão com banco de dados:

```
Connection con = ConexaoMySql.getConexao();
```

- Método getConexao() retorna instancia de Connection materializada no objeto con.

Classe de Conexão com Banco de dados

- Realizar a comando SQL:

Chamar o método createStatement() para obter uma instantancia de Statement materializada no objeto st.
Com acesso ao st é possível chamar o método execute(sql) que irá receber os comandos SQL.
Importante: método st.execute é destinado a operações que não retornam dados (INSERT, UPDATE, DELETE).

```
try
{
    Statement st = con.createStatement();
    String sql = "INSERT INTO   MARCA (descricao) VALUES ('" + txtDescricao.getText()  +"')";
    st.execute(sql);

    JOptionPane.showMessageDialog(null,  "Marca Inserida com Sucesso");
    System.out.println("Inserido  com Sucesso");

    txtDescricao.setText("");

}
catch (SQLException e) {
    System.out.println(e.getMessage());

}
```

Classe de Conexão com Banco de dados

- Realizar a consulta SQL:

```
try
 {
    Statement st = con.createStatement();
    String sql = "SELECT * FROM MARCA";

    ResultSet rs = st.executeQuery(sql);

    while (rs.next()) {
      System.out.println(rs.getInt("codigo"));
      System.out.println(rs.getString("descricao"));
    }

    st.close()
 }
 catch (SQLException e) {
    System.out.println(e);
}
```

Chamar o método createStatement() para obter uma instantancia de Statement materializada no objeto st. Com acesso ao st é possível chamar o método executeQuery(sql) que irá receber os comandos SQL e irá destinar os registros retornados pelo select no objeto ResultSet.
ResultSet é uma lista onde é possível obter seus valores através de um laço de repetição.

Dentro do laço de repetição é possível obter o valor da coluna desejada do banco de dados informando o nome da coluna como parâmetro dos métodos get. Utilizar getString se o valor da coluna for texto ou getInt se o valor for um inteiro. Utilizar get conforme tipagem dos valores armazenados no banco.

Bibliografia

1) SANTOS, Rafael. Introdução à Programação Orientada a Objetos Usando Java. 2ª ed. Rio de Janeiro: Campus - Elsevier, 2013.

2) Serson,Roberto Rubinstein. Programação Orientada a Objetos com Java 6. Brasport, 2007.

Anhanguera

Dúvidas ?

walter.gima@anhanguera.com