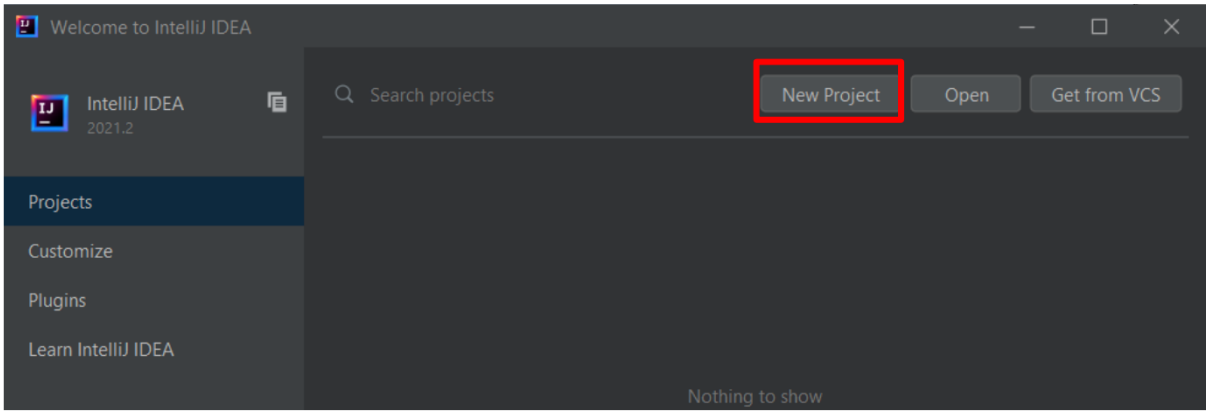
Aula 8 - Integração MySQL - Java Parte 2

Created	@February 3, 2022 3:28 PM
Class	
Type	
Materials	
Reviewed	☐
Property	

Integração entre MySQL e Java com Herança e Abstração

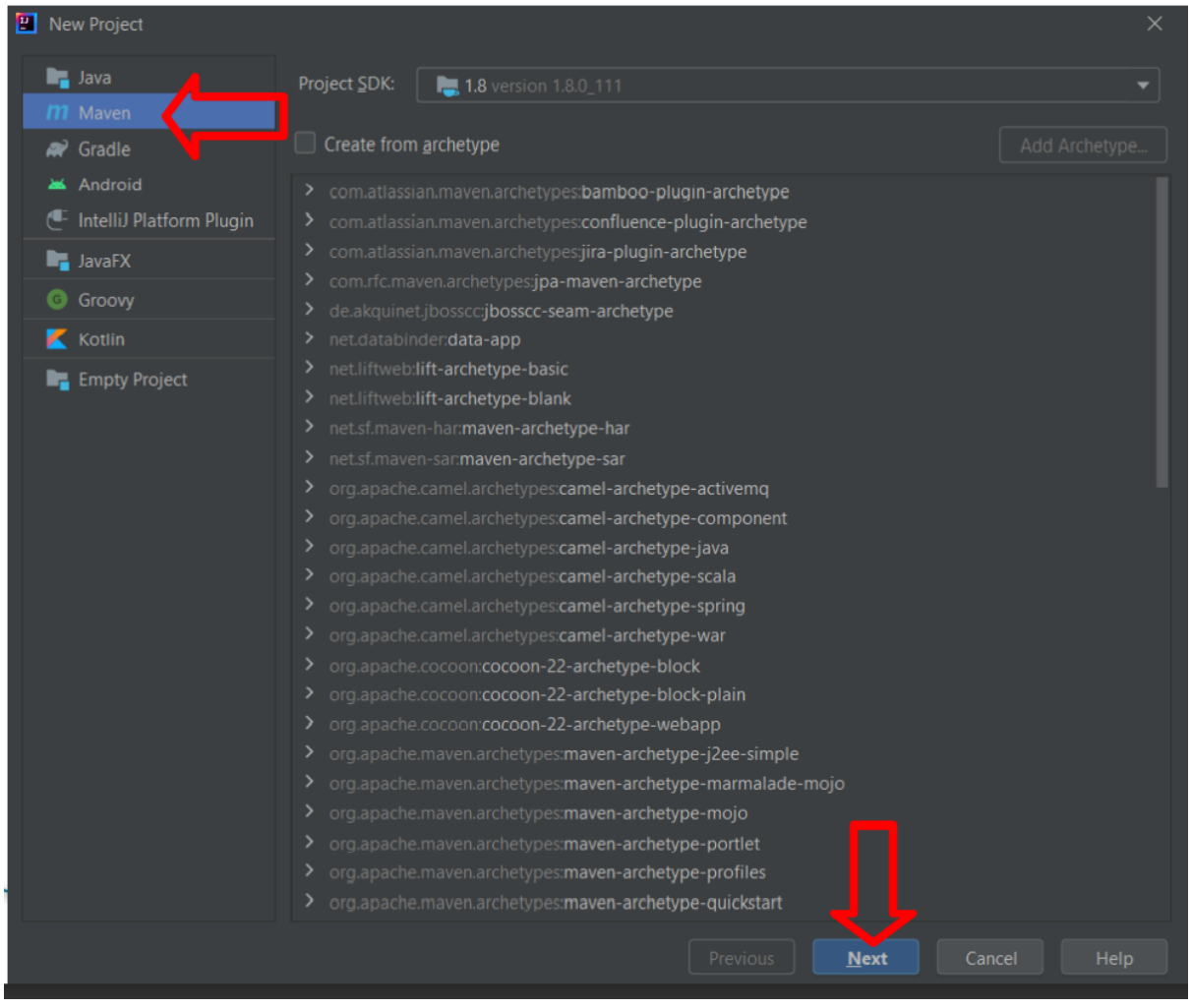
Criando novo projeto no Intellij

Ao iniciar o Intellij, se não tiver criado nenhum projeto anteriormente, a seguinte tela será exibida:

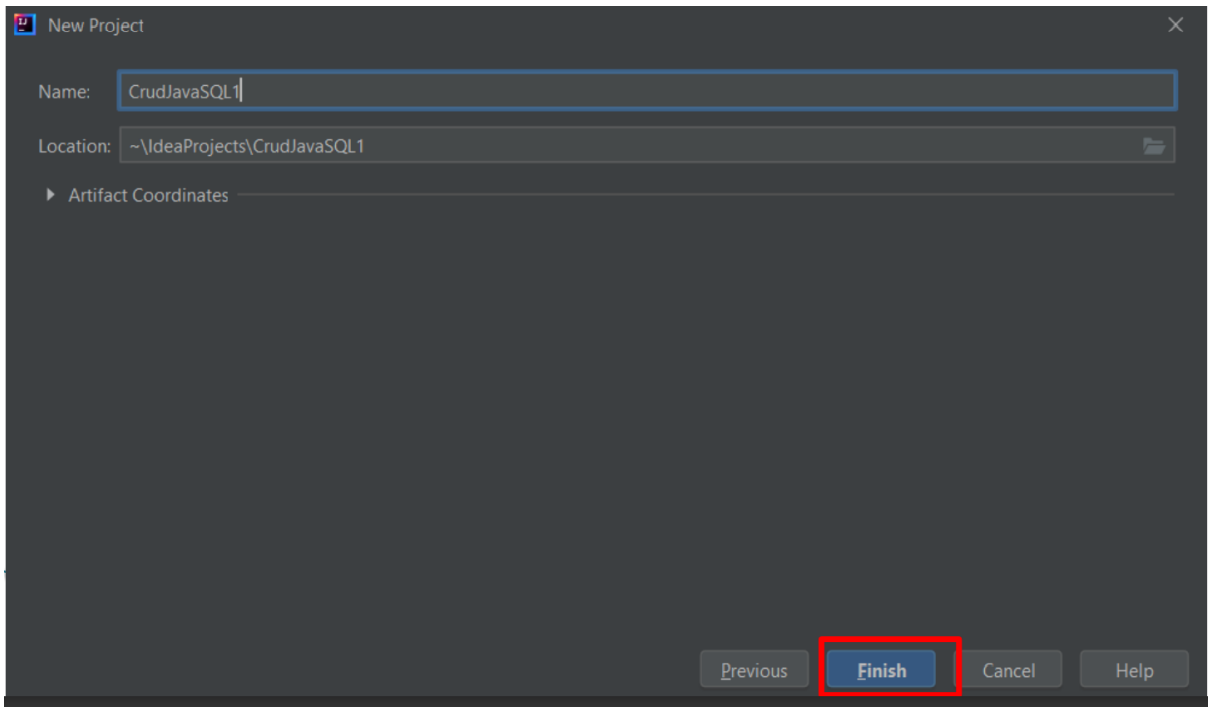


Clique em “New Project” para criar um novo projeto. Se tiver com projeto já aberto, vá em “File” → “Close Project”, será exibida a janela acima.

Escolha a opção Maven, a versão do Java e em seguida clique em “Next”:



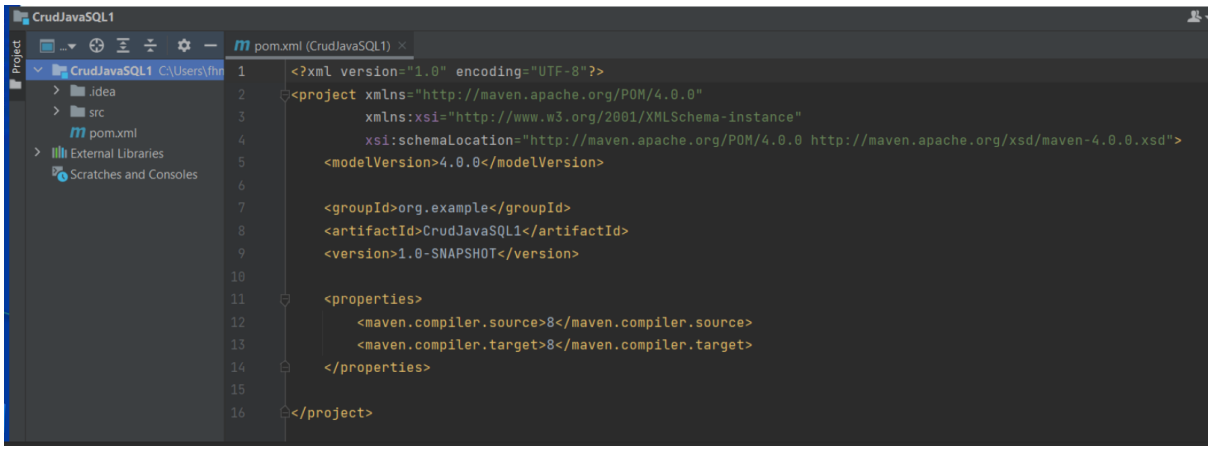
Nomeie seu projeto e clique em “Finish”:



Quando for finalizado a criação do projeto, o IntelliJ irá abrir o pom.xml, ele é o arquivo onde o Maven (Gerenciador do projeto Java) instala todas as dependências do projeto de maneira automática.

● O Maven é semelhante ao npm do JavaScript e o pip do Python;

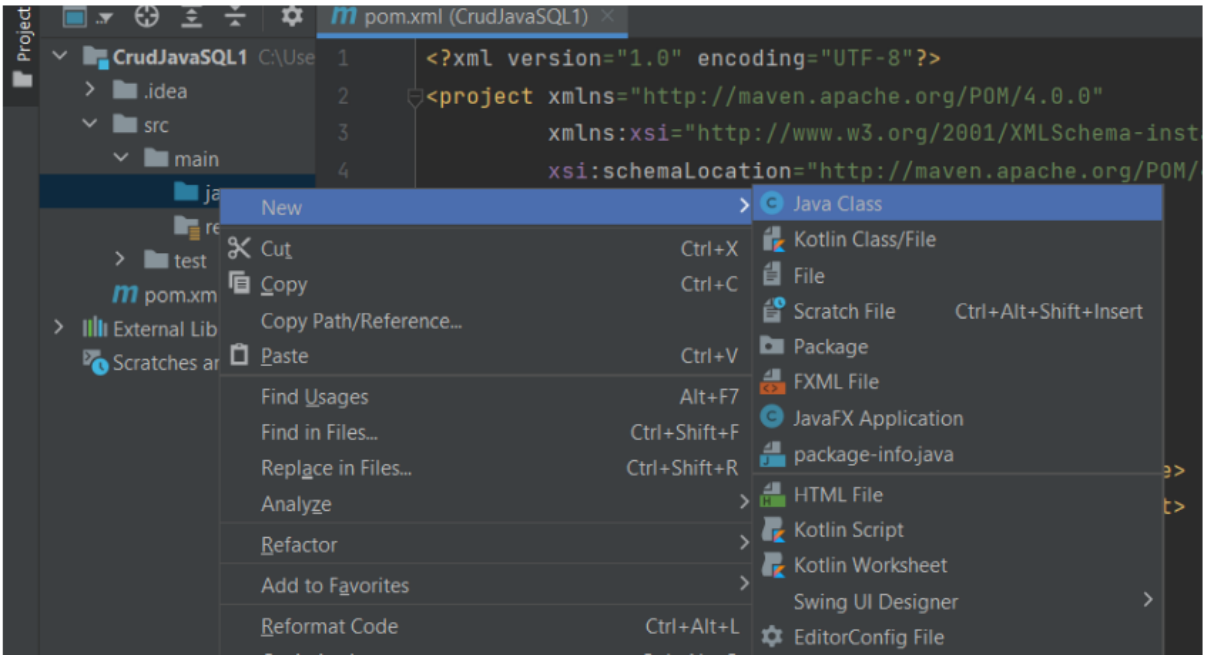
● O pom.xml é um arquivo semelhante package.json do JavaScript ou o pubspec.yaml do Flutter;

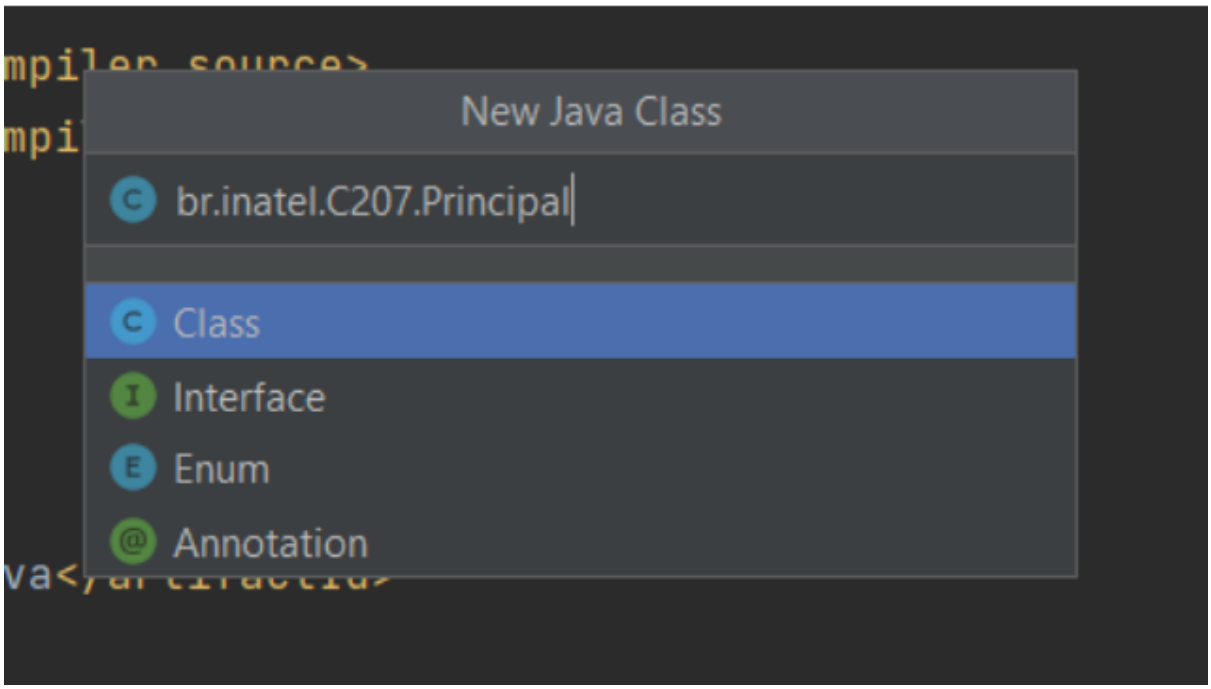


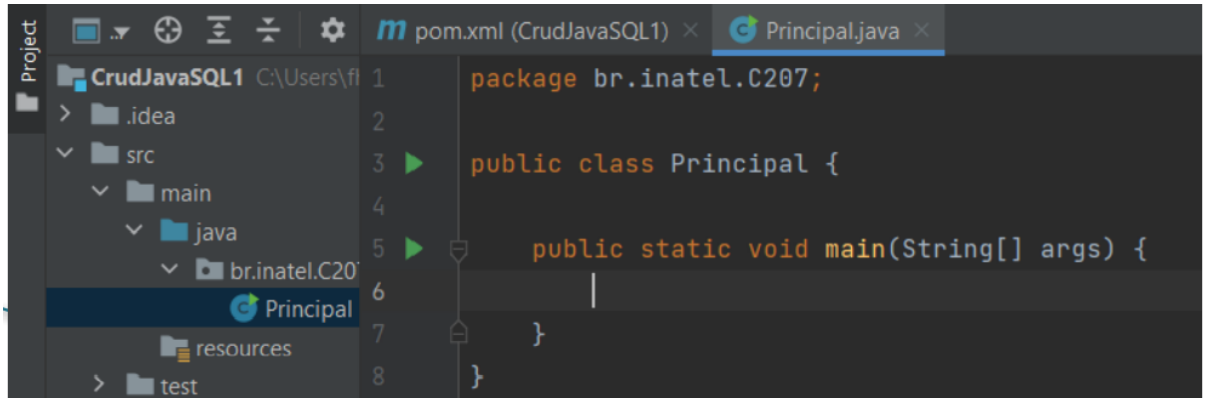
Vamos instalar a dependência do MySQL: Pra isso escreva “dependencies” e dentro dela, escreva “mysql”, o intellisense ajuda muito!

Criando Classe Principal

Dentro da pasta “java” em “main”, clique com o botão direito, vá em “New” → “Java Class”. Será aberto uma janela onde colocará o nome do pacote e da classe. Após pressionar “Enter”, será criado a classe e deverá escrever o método main.







Qual banco vamos trabalhar?

Vamos criar um banco no MySQL Workbench para podermos realizar um CRUD. Ele terá duas tabelas num relacionamento 1xN.

```sql
DROP DATABASE IF EXISTS exercicio;
CREATE DATABASE exercicio;
USE exercicio;

CREATE TABLE IF NOT EXISTS curso(
    id INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR(45),
    PRIMARY KEY(id)
);

CREATE TABLE IF NOT EXISTS aluno(
    matricula INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    ano_nasc int,
    sexo ENUM('F', 'M'),
    fk_idcurso INT,
    CONSTRAINT fk
        FOREIGN KEY (fk_idcurso) REFERENCES curso(id) ON UPDATE CASCADE
);
```

Criando Classes de Registros

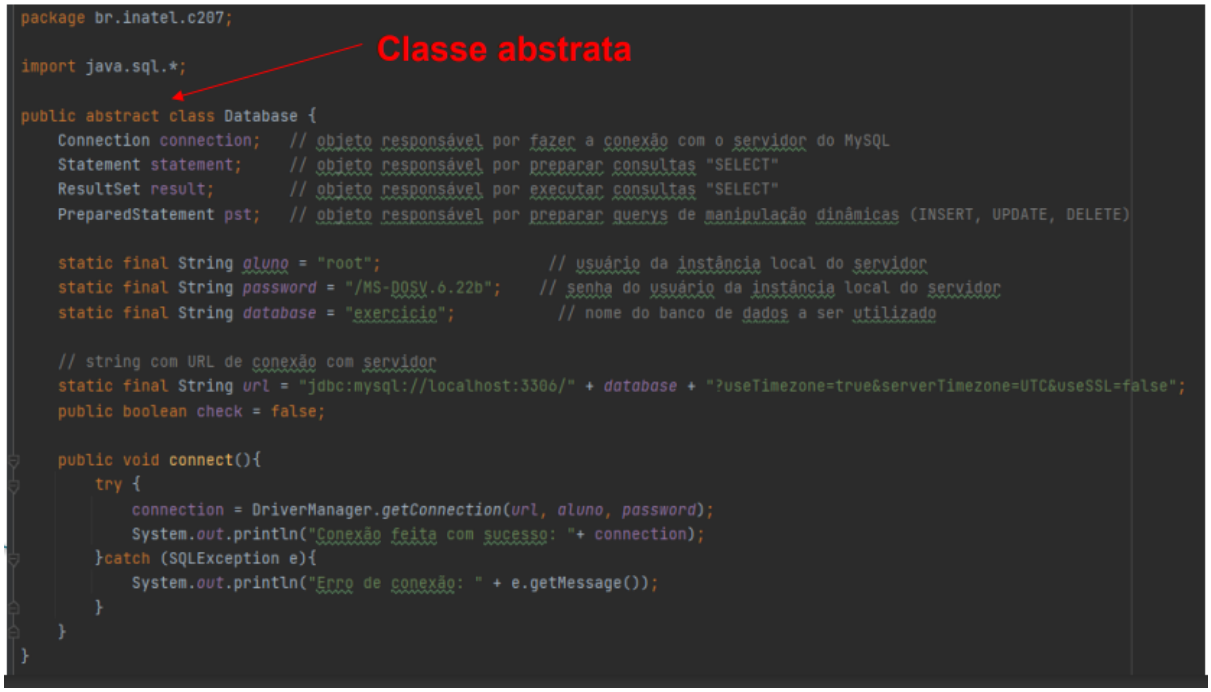
De modo análogo a criação da classe principal, vamos criar as classes Aluno e Curso para receber ou enviar registros vindos do banco.

```java
package br.inatel.c207;

public class Aluno {
    public int matricula;
    public int fk_idcurso;
    private String nome;
    private int ano_nasc;
    private String sexo;

    public Aluno(String nome, int ano_nasc, String sexo){
        this.nome = nome;
        this.ano_nasc = ano_nasc;
        this.sexo = sexo;
    }

    public String getNome() { return nome; }

    public void setNome(String nome) { this.nome = nome; }

    public int getAno_nasc() { return ano_nasc; }

    public void setAno_nasc(int ano_nasc) { this.ano_nasc = ano_nasc; }

    public String getSexo() { return sexo; }

    public void setSexo(String sexo) { this.sexo = sexo; }
}
```

```java
package br.inatel.c207;

public class Curso {
    private String nome;
    public int id;

    public Curso(String nome){
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Criando a classe do Banco

Dentro da classe Database, iremos colocar somente o método de conexão e os objetos de manipulação com o banco. O CRUD iremos fazer em outras classes.




```java
package br.inatel.c207;

import java.sql.*;

public abstract class Database {
    Connection connection;   // objeto responsável por fazer a conexão com o servidor do MySQL
    Statement statement;     // objeto responsável por preparar consultas "SELECT"
    ResultSet result;        // objeto responsável por executar consultas "SELECT"
    PreparedStatement pst;   // objeto responsável por preparar querys de manipulação dinâmicas (INSERT, UPDATE, DELETE)

    static final String aluno = "root";                 // usuário da instância local do servidor
    static final String password = "/MS-DOSV.6.22b";    // senha do usuário da instância local do servidor
    static final String database = "exercicio";          // nome do banco de dados a ser utilizado

    // string com URL de conexão com servidor
    static final String url = "jdbc:mysql://localhost:3306/" + database + "?useTimezone=true&serverTimezone=UTC&useSSL=false";
    public boolean check = false;

    public void connect(){
        try {
            connection = DriverManager.getConnection(url, aluno, password);
            System.out.println("Conexão feita com sucesso: "+ connection);
        }catch (SQLException e){
            System.out.println("Erro de conexão: " + e.getMessage());
        }
    }
}
```

Criando Classe CursoDB - Insert

Com essa classe herdada de Database, podemos criar todos os métodos necessários para fazer o CRUD com a tabela Curso, pois todos os objetos e métodos presentes na classe Database, podem ser acessados pelas classes que receberem a herança. Olhe o método de inserção criado de maneira direta.

```java
package br.inatel.c207;

import java.sql.SQLException;
import java.util.ArrayList;

public class CursoDB extends Database{

    // ---------------------------INSERINDO REGISTRO---------------------------
    public boolean insertCurso(Curso curso){
        connect();
        String sql = "INSERT INTO curso(nome) VALUES(?)";
        try {

            pst = connection.prepareStatement(sql);
            pst.setString( e 1, curso.getNome());      // concatena nome na primeira ? do comando
            pst.execute();                            // executa o comando
            check = true;

        } catch (SQLException e) {
            System.out.println("Erro de operação: " + e.getMessage());
            check = false;
        }
        finally {
            try{
                connection.close();
                pst.close();
            }catch (SQLException e){
                System.out.println("Erro ao fechar a conexão: " + e.getMessage());
            }
        }
        return check;
    }
```

SELECT

```java
// ----------------------------BUSCANDO TODOS REGISTROS----------------------------
public ArrayList<Curso> researchCurso(){
    connect();
    ArrayList<Curso> cursos = new ArrayList<>();
    String sql = "SELECT * FROM curso";
    try{
        statement = connection.createStatement();
        result = statement.executeQuery(sql);

        while(result.next()){
            Curso cursoTemp = new Curso(result.getString( s: "nome"));
            cursoTemp.id = result.getInt( s: "id");
            System.out.println("ID = " + cursoTemp.id);
            System.out.println("Nome = " + cursoTemp.getNome());
            System.out.println("------------------------------");
            cursos.add(cursoTemp);
        }
    }catch (SQLException e){
        System.out.println("Erro de operação: " + e.getMessage());
    }finally {
        try {
            connection.close();
            statement.close();
            result.close();
        }catch (SQLException e){
            System.out.println("Erro ao fechar a conexão: " + e.getMessage());
        }
    }
    return cursos;
}
```

UPDATE

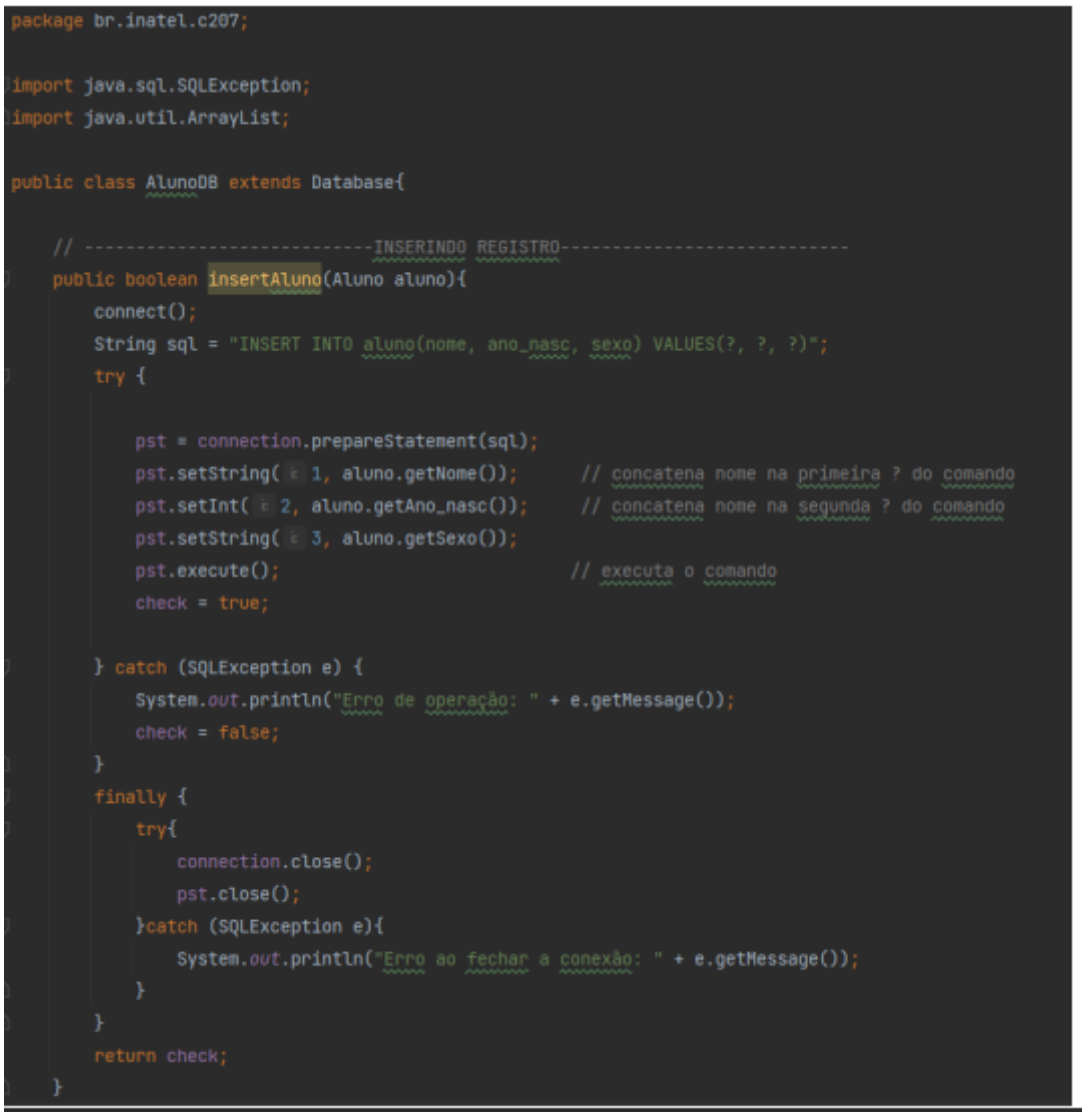
```java
// ---------------------------ATUALIZANDO NOME NO REGISTRO---------------------------
public boolean updateCurso(int id, String nome){
    connect();
    String sql = "UPDATE curso SET nome=? WHERE id=?";
    try{
        pst = connection.prepareStatement(sql);
        pst.setString( i: 1, nome);
        pst.setInt( i: 2, id);
        pst.execute();
        check = true;
    }catch (SQLException e){
        System.out.println("Erro de operação: " + e.getMessage());
        check = false;
    }finally {
        try {
            connection.close();
            pst.close();
        }catch (SQLException e) {
            System.out.println("Erro ao fechar a conexão: " + e.getMessage());
        }
    }
    return check;
}
```

DELETE

```
    // ----------------------------EXCLUINDO REGISTRO----------------------------
    public boolean deleteCurso(int id) {
        connect();
        String sql = "DELETE FROM curso WHERE id=?";
        try{
            pst = connection.prepareStatement(sql);
            pst.setInt( i 1, id);
            pst.execute();
            check = true;
        }catch (SQLException e){
            System.out.println("Erro de operação: " + e.getMessage());
            check = false;
        }finally {
            try {
                connection.close();
                pst.close();
            }catch (SQLException e){
                System.out.println("Erro ao fechar a conexão: " + e.getMessage());
            }
        }
        return check;
    }
}
```

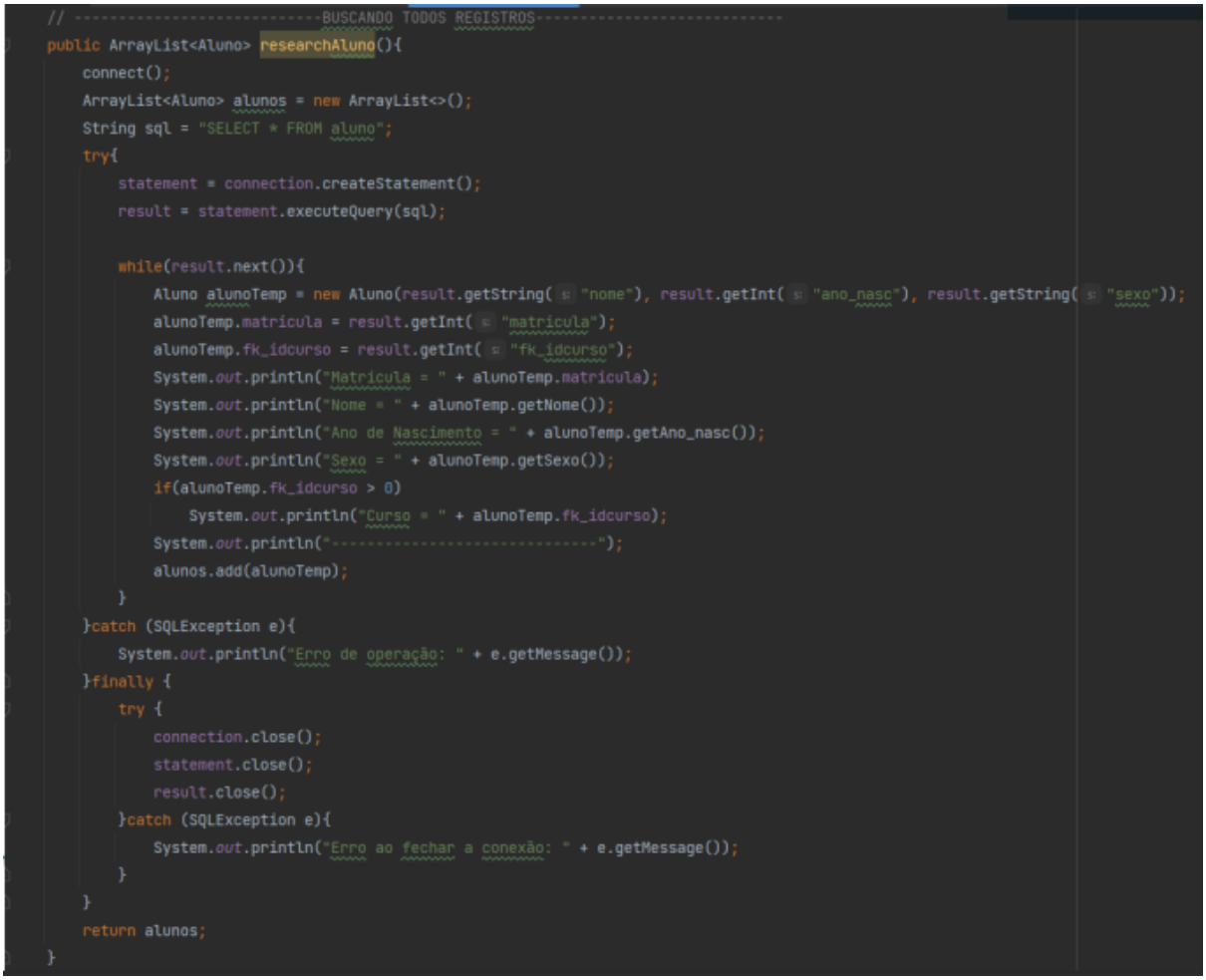
Criando Classe AlunoDB - INSERT

Com essa classe herdada de Database, podemos criar todos os métodos necessários para fazer o CRUD com a tabela Aluno, pois todos os objetos e métodos presentes na classe Database, podem ser acessados pelas classes que receberem a herança. Olhe o método de inserção criado de maneira direta.



```java
package br.inatel.c207;

import java.sql.SQLException;
import java.util.ArrayList;

public class AlunoDB extends Database{

    // ---------------------------INSERINDO REGISTRO--------------------------
    public boolean insertAluno(Aluno aluno){
        connect();
        String sql = "INSERT INTO aluno(nome, ano_nasc, sexo) VALUES(?, ?, ?)";
        try {

            pst = connection.prepareStatement(sql);
            pst.setString( i: 1, aluno.getNome());      // concatena nome na primeira ? do comando
            pst.setInt( i: 2, aluno.getAno_nasc());     // concatena nome na segunda ? do comando
            pst.setString( i: 3, aluno.getSexo());
            pst.execute();                              // executa o comando
            check = true;

        } catch (SQLException e) {
            System.out.println("Erro de operação: " + e.getMessage());
            check = false;
        }
        finally {
            try{
                connection.close();
                pst.close();
            }catch (SQLException e){
                System.out.println("Erro ao fechar a conexão: " + e.getMessage());
            }
        }
        return check;
    }
```

SELECT



```java
// -------------------------BUSCANDO TODOS REGISTROS-------------------------
public ArrayList<Aluno> researchAluno(){
    connect();
    ArrayList<Aluno> alunos = new ArrayList<>();
    String sql = "SELECT * FROM aluno";
    try{
        statement = connection.createStatement();
        result = statement.executeQuery(sql);

        while(result.next()){
            Aluno alunoTemp = new Aluno(result.getString( s: "nome"), result.getInt( s: "ano_nasc"), result.getString( s: "sexo"));
            alunoTemp.matricula = result.getInt( s: "matricula");
            alunoTemp.fk_idcurso = result.getInt( s: "fk_idcurso");
            System.out.println("Matricula = " + alunoTemp.matricula);
            System.out.println("Nome = " + alunoTemp.getNome());
            System.out.println("Ano de Nascimento = " + alunoTemp.getAno_nasc());
            System.out.println("Sexo = " + alunoTemp.getSexo());
            if(alunoTemp.fk_idcurso > 0)
                System.out.println("Curso = " + alunoTemp.fk_idcurso);
            System.out.println("----------------------------");
            alunos.add(alunoTemp);
        }
    }catch (SQLException e){
        System.out.println("Erro de operação: " + e.getMessage());
    }finally {
        try {
            connection.close();
            statement.close();
            result.close();
        }catch (SQLException e){
            System.out.println("Erro ao fechar a conexão: " + e.getMessage());
        }
    }
    return alunos;
}
```

UPDATE

```java
// ---------------------------ATUALIZANDO REGISTRO---------------------------
// OBS: Os registros foram criados sem relacionamentos, aqui estamos criando
// relacionamentos em registros já existentes
public boolean updateFkAluno(int matricula, int id_curso){
    connect();
    String sql = "UPDATE aluno SET fk_idcurso=? WHERE matricula=?";
    try{
        pst = connection.prepareStatement(sql);
        pst.setInt( i: 1, id_curso);
        pst.setInt( i: 2, matricula);
        pst.execute();
        check = true;
    }catch (SQLException e){
        System.out.println("Erro de operação: " + e.getMessage());
        check = false;
    }finally {
        try {
            connection.close();
            pst.close();
        }catch (SQLException e) {
            System.out.println("Erro ao fechar a conexão: " + e.getMessage());
        }
    }
    return check;
}
```

DELETE

```java
    // ----------------------------EXCLUINDO REGISTRO----------------------------
    public boolean deleteAluno(int matricula) {
        connect();
        String sql = "DELETE FROM aluno WHERE matricula=?";
        try{
            pst = connection.prepareStatement(sql);
            pst.setInt( i: 1, matricula);
            pst.execute();
            check = true;
        }catch (SQLException e){
            System.out.println("Erro de operação: " + e.getMessage());
            check = false;
        }finally {
            try {
                connection.close();
                pst.close();
            }catch (SQLException e){
                System.out.println("Erro ao fechar a conexão: " + e.getMessage());
            }
        }
        return check;
    }
}
```

Fazendo CRUD na Main

Chamando todos os métodos na classe principal e realizando testes

```java
public static void main(String[] args) {
    // Criando objetos de manipulação no BD
    AlunoDB alunoDB = new AlunoDB();
    CursoDB cursoDB = new CursoDB();
    // Criando objetos para inserir no BD
    Aluno aluno1 = new Aluno( nome: "Flavio", ano_nasc: 1998, sexo: "M");
    Aluno aluno2 = new Aluno( nome: "Aline", ano_nasc: 1998, sexo: "F");
    Curso curso1 = new Curso( nome: "Eng. Computação");
    Curso curso2 = new Curso( nome: "Pedagogia");
    // Inserindo informações no BD
    alunoDB.insertAluno(aluno1);
    alunoDB.insertAluno(aluno2);
    cursoDB.insertCurso(curso1);
    cursoDB.insertCurso(curso2);
    // Buscando informações no BD
    alunoDB.researchAluno();
    cursoDB.researchCurso();
    System.out.println("--------------------Atualizando informações no BD--------------------");
    alunoDB.updateFkAluno( matricula: 1, id_curso: 1);
    alunoDB.researchAluno();
    cursoDB.updateCurso( id: 2, nome: "Odontologia");
    cursoDB.researchCurso();
    System.out.println("--------------------Excluindo informações no BD--------------------");
    alunoDB.deleteAluno( matricula: 2);
    cursoDB.deleteCurso( id: 2);
    // Mostrando resultado final no BD
    alunoDB.researchAluno();
    cursoDB.researchCurso();
}
}
```

Resultados Finais

Observando o estado final tanto pela aplicação quanto pelo banco diretamente

```
Matricula = 1
Nome = Flavio
Ano de Nascimento = 1998
Sexo = M
Curso = 1
------------------------------
Conexão feita com sucesso: com.mysql.cj.jdbc.ConnectionImpl@27efef64
ID = 1
Nome = Eng. Computação
------------------------------

Process finished with exit code 0
```

	matricula	nome	ano_nasc	sexo	fk_idcurso
▸	1	Flavio	1998	M	1
•	NULL	NULL	NULL	NULL	NULL

	id	nome
▸	1	Eng. Computação
•	NULL	NULL