

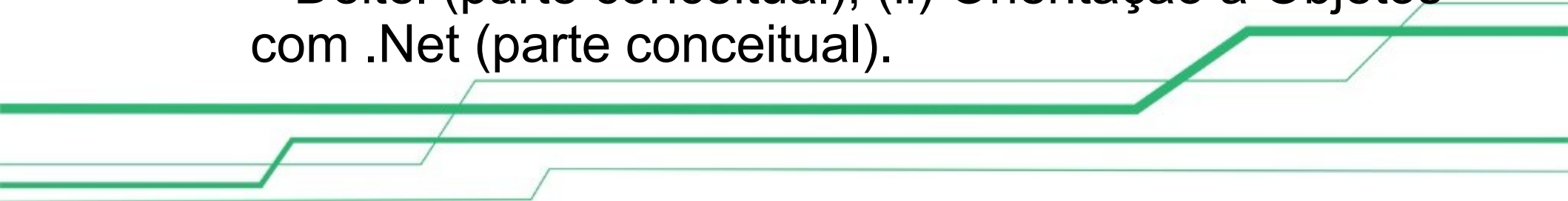
# MÓDULO 1

## Metodologias de Desenvolvimento


(Parte 2)



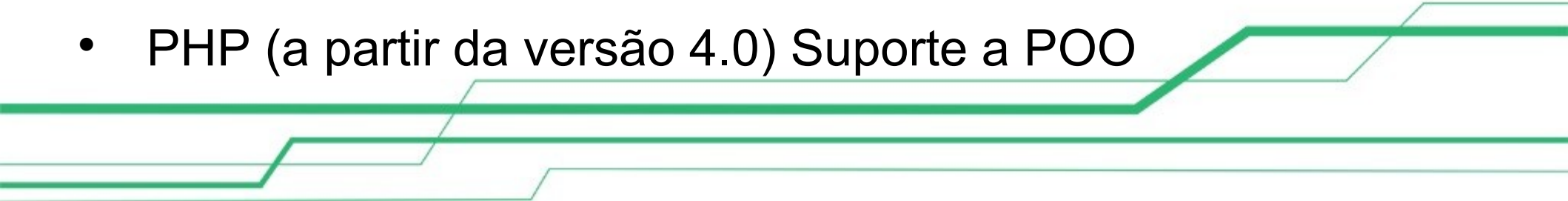
# Sobre o Módulo

- Objetivo do módulo: abordar o paradigma de programação OO.
  - Linguagem Java como ferramenta.
  - Material:
    - Apostila,
    - Slides de Aula
    - Lista Exercícios
    - Outros (referências): (i) Livro Java como Programar – Deitel (parte conceitual), (ii) Orientação a Objetos com .Net (parte conceitual).
- 

# O que é a POO?


- Paradigma de programação alternativo ao paradigma estruturado.
  - Implementa conceito de reutilização
  - Tende a facilitar a manutenção do projeto de código em comparação a PP
  - Permite encapsular dados contra acesso não autorizado, diferentemente da PP.
  - POO incentiva a criação de código modular e coeso, facilitando manutenções pontuais.
- 

# Exemplos de linguagens OO

- C++,
  - C#,
  - VB.NET,
  - Java,
  - Object Pascal,
  - Objective-C,
  - Python,
  - Ruby,
  - Smalltalk
  - PHP (a partir da versão 4.0) Suporte a POO
- 

# Agenda

## Aula 1

- Classes e Objetos
  - Atributos e Propriedades
  - Tipos de dados
  - Encapsulamento
  - Abstração
  - Métodos
  - Associação
- + Práticas e/ou Exercícios**
- 

# Agenda

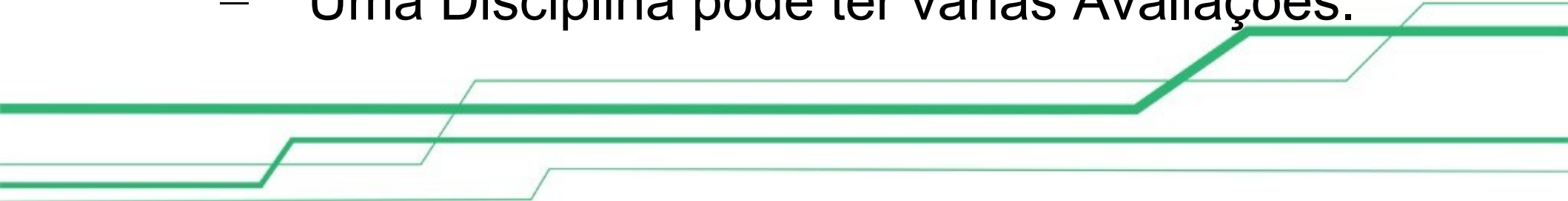
## Aula 2

- Herança
  - Polimorfismo
- + Práticas e/ou Exercícios**



# Classes e Objetos

- **Cenário inicial**


- A Faculdade XYZ deseja criar um sistema que permita aos professores e alunos acessarem informações sobre Avaliações, e Disciplinas.
  - Professores podem lançar e alterar a nota dos Alunos, nas respectivas Avaliações.
  - A Faculdade é histórica e está no mercado a mais de 100 anos formando profissionais em diversos cursos.
  - Uma Disciplina pode ter várias Avaliações.
- 

# Classes e Objetos

- **Classe:**

- Abstração que representa algo do mundo real:
- ✓ Disciplina
- ✓ Avaliação
- ✓ Aluno
- ✓ Professor

- **Objeto:**

- Instâncias das classes
  - Gabarito para a definição de Objetos
  - ✓ Disciplina: Metodologia de Desenvolvimento
  - ✓ Avaliação: Prova Prática
  - ✓ Aluno: William
  - ✓ Professor: Raimundo
- 

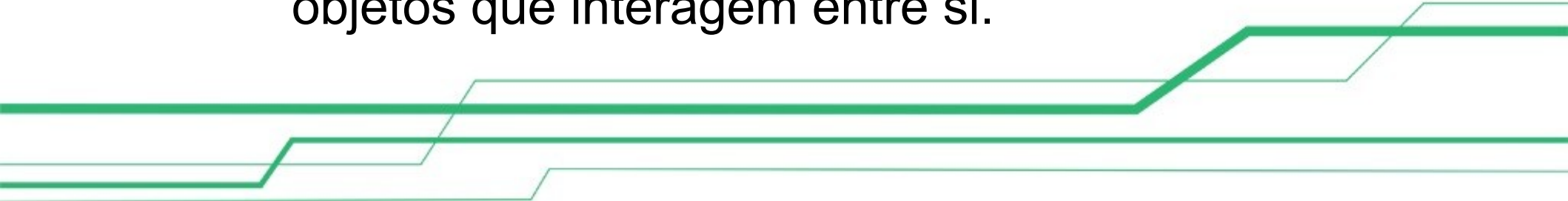


# Classes e Objetos

- **Classe:**

- Gabarito para a definição de Objetos
- Possui a “receita de bolo” para a criação de objetos
- Análogo à um arquivo de código em linguagens estruturadas.

- **Objeto:**

- Cria um espaço em memória para definição das informações
  - Um programa OO é composto por um conjunto de objetos que interagem entre si.
- 

# Atributos e Propriedades

- **Atributos:**
  - Dados de uma classe.
  - Características que formam a estrutura de uma classe.
    - ✓ Professor: cod, nome, cpf, rg e especialidade
    - ✓ Aluno: mat, nome, cpf e rg.
    - ✓ Avaliação: cod, título, data, aluno, professor e valor.
    - ✓ Disciplina: cod, duração, data início, data de término, avaliações e créditos.

# Atributos e Propriedades

- **Propriedades:**
  - São Moderadores de acesso
    - ✓ Public
    - ✓ Private
    - ✓ Protected



# Tipos de Dados

- **Tipos primitivos**

- *Int*
- *Boolean.*
- *float,*
- *Double,*
- *String.*
- *Arraylist.*
- *Object*
- Etc.*

- **Tipos compostos**

- Tipos criados a partir de tipos simples
- 

# Encapsulamento

- Ref 1: Técnica que faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos.
  - Ref 1: Orientação a Objetos com .net 2ed.



# Encapsulamento

- Ref 2: O **encapsulamento** é um conceito da POO onde o estado de objetos (as variáveis da classe) e seus comportamentos (os métodos da classe) são agrupados em conjuntos segundo o seu grau de relação.
- Ref 2: O mecanismo para restringir o acesso a alguns dos componentes do objeto é a **definição de ocultação de informações**.
  - Ref 2:  
<http://www.devmedia.com.br/encapsulamento-em-java-primeiros-passos/31177#ixzz3yy8xLuuA>

# Abstração

- Prática de se concentrar somente nos detalhes mais importantes de um objeto.
- Possibilita o descarte de aspectos menos importantes para a funcionalidades do mesmo.



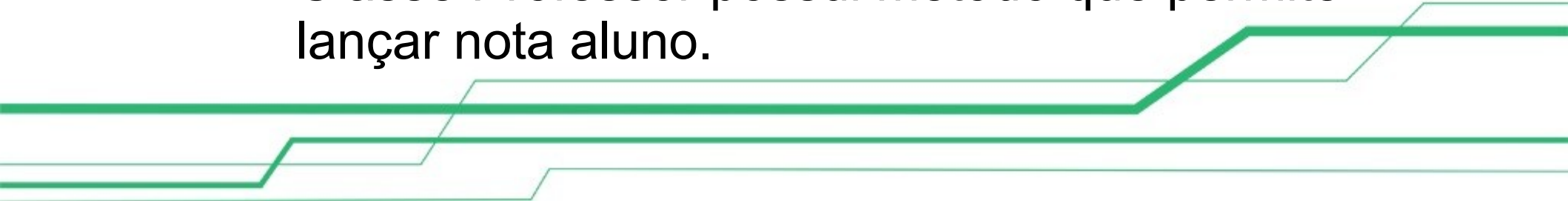
# Prática 01

- Breve sobre o Eclipse
- Implementar as Classes e Atributos do cenário descrito.





# Métodos

- Análogo aos procedimentos e funções da programação estruturada.
  - Definem o comportamento dos objetos de uma classe.
  - Ações ou operações que uma classe executa.
  - Composto por: (i) moderador de acesso, (ii) tipo de retorno (tipo de dado ou *void*), (iii) nome e (iv) parâmetros.
    - Classe Professor possui método que permite lançar nota aluno.
- 

# Métodos

- Métodos get e set
- Métodos construtores ou de criação
- Métodos vazios ou sobrecarregados
- Métodos com retorno e sem retorno.
- Métodos estáticos



# Associação

- Relacionamento entre classes
- Indica dependência estrutural entre classes.
- Fluxo de dados pode ser unidirecional ou bidirecional
- Exemplo:
  - Avaliação está associada com Aluno.



# Prática 02

- Implementar os Métodos das classes e as Associações (código a seguir).



```
public static void gravarAvaliacao(Avaliacao avaliacao)
{
    // persiste a avaliação no banco, por exemplo
}

public static ArrayList recuperaAvaliacoes(Aluno aluno)
{
    // Como se tivesse vindo da base de dados
    Aluno aluno1 = new Aluno(1, "Maria", "0000000", "444444");

    Avaliacao ava1 = new Avaliacao();
    ava1.setAluno(aluno1);
    ava1.setTitulo("Prova 01");
    ava1.setValor(13.5f);

    Avaliacao ava2 = new Avaliacao();
    ava2.setTitulo("Prova 02");
    ava2.setAluno(aluno1);
    ava2.setValor(14.5f);

    ArrayList listaAvalicoes = new ArrayList();
    listaAvalicoes.add(ava1);
    listaAvalicoes.add(ava2);

    return listaAvalicoes;
}
```

Classe  
Avaliacoes

```
public void lancarNotaAvaliacaoParaAluno(Avaliacao avaliacao, float nota)
{
    avaliacao.setValor(nota);
    Avaliacao.gravarAvaliacao(avaliacao);
}

public ArrayList buscaAvaliacoesPorAluno(Aluno alunoX)
{
    return Avaliacao.recuperaAvaliacoes(alunoX);
}
```

Classe  
Professor



```
1 package codigo;
2
3 import java.util.ArrayList;
4
5 public class principal {
6
7     public static void main(String[] args) {
8
9         // Monta aluno
10        Aluno aluno1 = new Aluno(01, "Willian", "00000000000", "11222333");
11
12        // Monta Avaliação
13        Avaliacao prova01 = new Avaliacao();
14        prova01.setCod(1);
15        prova01.setData("10/10/16");
16
17        // Associa aluno à avaliação
18        prova01.setAluno(aluno1);
19
20        // Professor lançando nota
21        Professor prof1 = new Professor();
22        prof1.lancarNotaAvaliacaoParaAluno(prova01, 10.5f);
23
24        // Professor recuperando nota
25        ArrayList listaAvaliacoes = prof1.buscaAvaliacoesPorAluno(aluno1);
26        // Imprime lista Avaliações
27
28        for(int i = 0; i < listaAvaliacoes.size(); i++)
29        {
30            Avaliacao aux = (Avaliacao) listaAvaliacoes.get(i);
31            System.out.println("Aluno: " + aux.getAluno().getNome());
32            System.out.println("Avaliação: " + aux.getTítulo());
33            System.out.println("Nota: " + aux.getValor());
34            System.out.println("");
35        }
36    }
37 }
38
39 }
40
```

Classe  
principal

# Herança

- Recurso da POO para compartilhar similaridades entre classes e ao mesmo tempo preservar as diferenças entre elas.
- Permite que atributos e métodos de uma classe, comuns a várias outras fiquem centralizados em classes pai, de onde classes filhas herdam.





# Polimorfismo

- Polimorfismo é a capacidade de um objeto poder ser referenciado de várias formas diferentes.



# Prática 03

- Implementar Herança entre as classes Aluno e Professor (próximo slide).



```
public class Pessoa
{
    protected String nome;
    protected String cpf;
    protected String rg;

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getCpf() {
        return cpf;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
    public String getRg() {
        return rg;
    }
    public void setRg(String rg) {
        this.rg = rg;
    }

    public Pessoa(String pNome, String pCpf, String pRg ) {
        this.nome = pNome;
        this.cpf = pCpf;
        this.rg = pRg;
    }

    public Pessoa() {
    }
}
```

Classe  
Pessoa

```
public Aluno (int pMat, String pNome, String pCpf, String pRg )  
{  
    super(pNome, pCpf, pRg);  
    this.mat = pMat;  
}
```

Classe  
Aluno  
(Construtor)

# Prática 04

- Crie um projeto Chamado ExemploPolimorfismo.
- Implementar Polimorfismo para operações matemáticas (prox slide).



OperacaoMatematica.java

Calculos.java

Soma.java

```
package codigos;
```

```
public class OperacaoMatematica {
```

```
    public double calcular(double x, double y){  
        return 0;
```

```
    }
```

```
}
```

Classe  
OperacaoMatematica

```
public class Soma extends OperacaoMatematica {  
    public double calcular(double x, double y){  
        return x + y;  
    }  
}
```

Classe  
Soma



```
public class Calculos {  
  
    //EXECUTA A OPERACAO - COM POLIMORFISMO  
    public static void calculaOperacao(OperacaoMatematica o, double x, double y){  
        System.out.println(o.calcular(x, y));  
    }  
  
    public static void main(String[] args) {  
        // Uma Forma  
        calculaOperacao (new Soma(), 10, 5);  
    }  
}
```

Classe  
Calculos

