- Conceitos
- Abstrata
- Interface

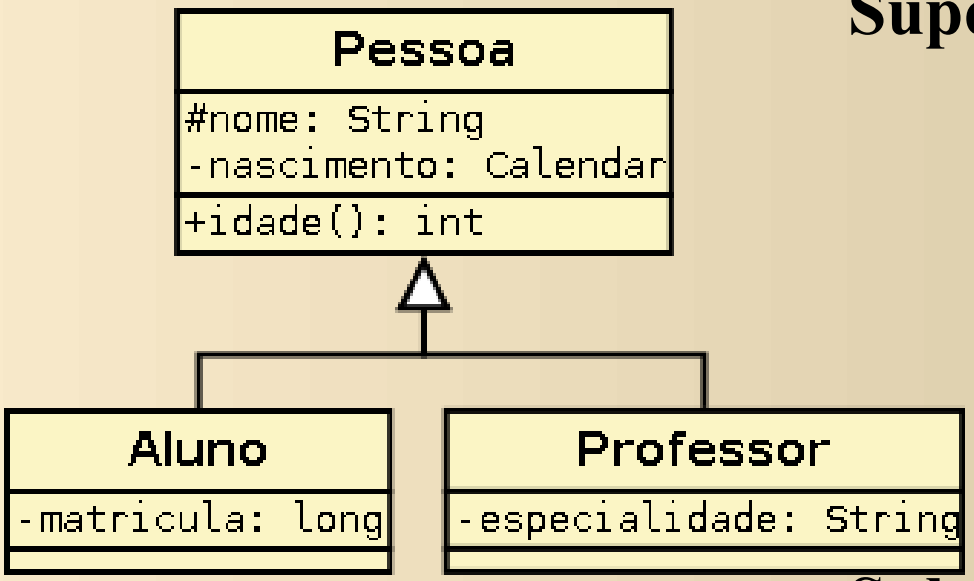
Herança

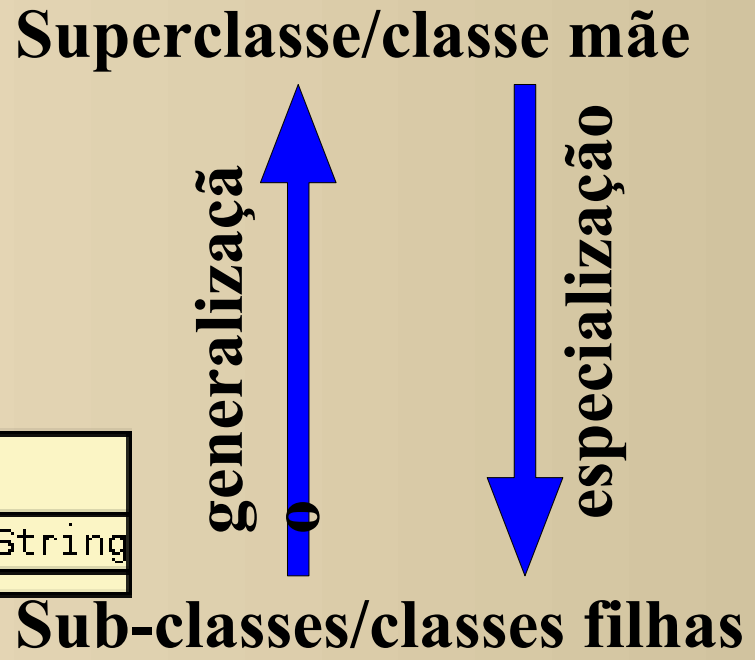
Prof.Dr. Enzo Seraphim

Generalização

Conceitos

- Palavras chaves usadas para identifica-la:
  - Relação semântica: “é um” / “é uma”)
  - Aluno é uma pessoa; Professor é uma Pessoa
- Sub-classes herdam da super-classe seus atributos, operações e relações, podendo acrescentar outras



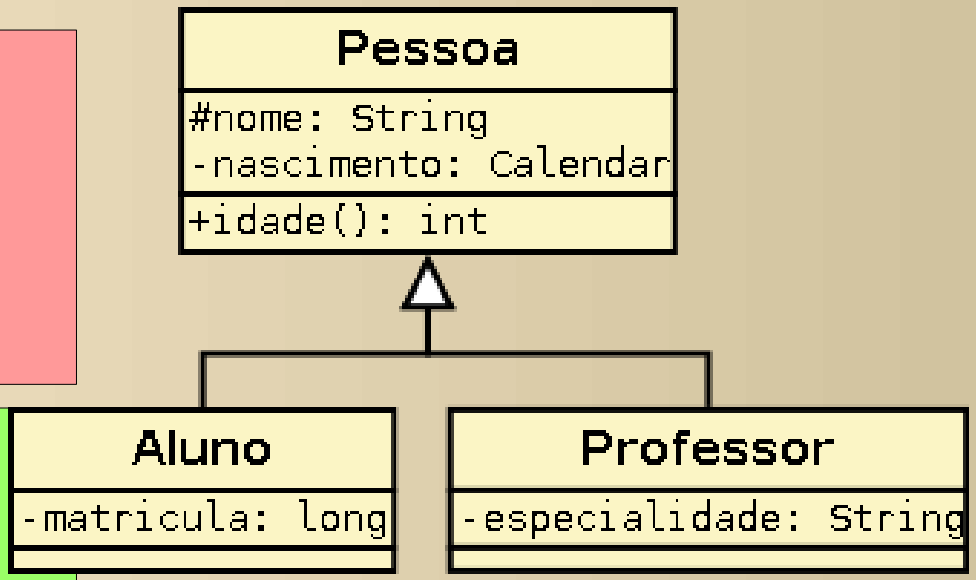


Herança das assinaturas

- Subclasse herda atributos e operações da super-classe
- Acesso ao atributo ou a operacao depende da visibilidade: private, protected e public

Conceitos

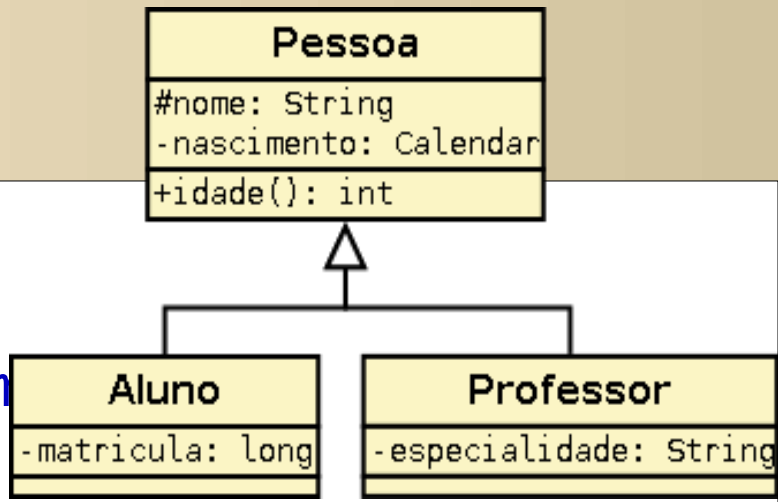
Atributo privado nascimento não é acessado por Aluno ou Professor



Atributo público idade() e atributo protegido nome são acessados por Aluno ou Professor

Visibilidade protected em Java dá acesso por amizade a todas classes do pacote

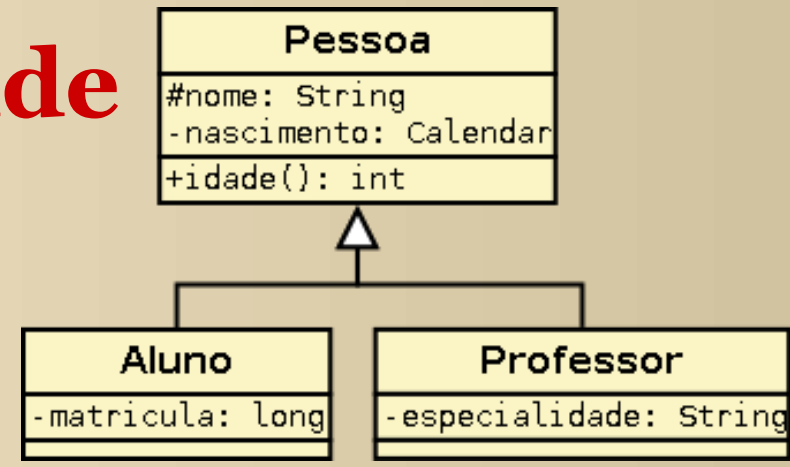
Extends




```
public class Pessoa {
 private Calendar nascim
 protected String nome;
 public int idade() {
  return (int)ChronoUnit.YEARS.between(
   LocalDate.of(
    nascimento.get(Calendar.YEAR),
    nascimento.get(Calendar.MONTH),
    nascimento.get(Calendar.DAY_OF_MONTH)),
    LocalDate.now());  }   //gets e sets   }
public class Aluno extends Pessoa{
   private long matricula;
   //gets e sets   }
public class Professor extends Pessoa{
   private String especialidade;
   //gets e sets   }
```

Conceitos

Substitutabilidade



- um objeto da sub-classe podem-se passar um objeto da superclasse

Conceitos

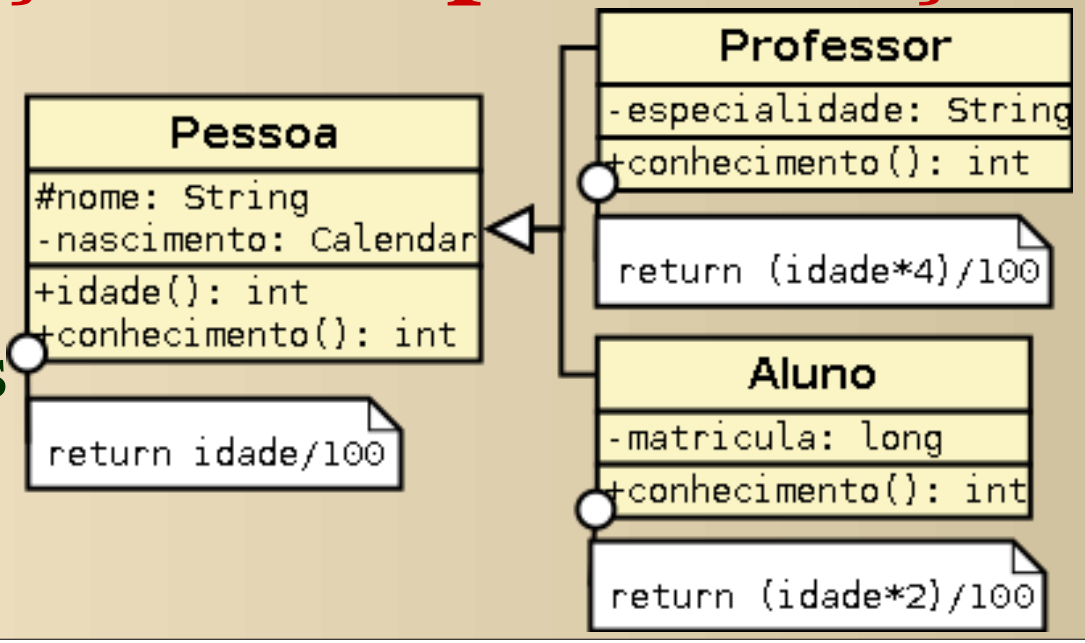
```
//Aluno a = new Pessoa() //não compila
Pessoa a = new Pessoa();
Pessoa b = new Aluno();
Pessoa c = new Professor();
List<Pessoa> l = new ArrayList<Pessoa>();
l.add(a);
l.add(b);
l.add(c);
l.add(new Aluno());
for (Pessoa pessoa : l) {
 System.out.println(pessoa);
}
```

```
Pessoa@282ba1e
Aluno@3ecf72fd
Professor@483bf400
Aluno@21a06946
```

Sobreposição de implementação

Conceitos

- Subclasse pode sobrepor com novas implementações as operações herdadas

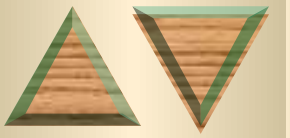



```
Pessoa a = new Pessoa();
a.setNascimento(new GregorianCalendar(1990,1,1));
System.out.println("a="+a.conhecimento()); //15
Pessoa b = new Aluno();
b.setNascimento(new GregorianCalendar(2000,1,1));
System.out.println("b="+b.conhecimento()); //21
Pessoa c = new Professor();
c.setNascimento(new GregorianCalendar(1970,1,1));
System.out.println("c="+c.conhecimento()); //102
```

Construtor (revisão)

Conceitos

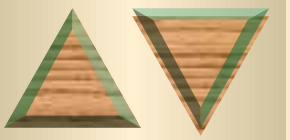
- Método chamado assim que uma nova instância do objeto é criada
- Caso não esteja declarada, toda linguagem declara automaticamente o construtor padrão (sem parâmetros)
- Qualquer declaração de construtor para classe suprime a declação automática
- Pode haver mais de um construtor na classe variando-se parâmetros



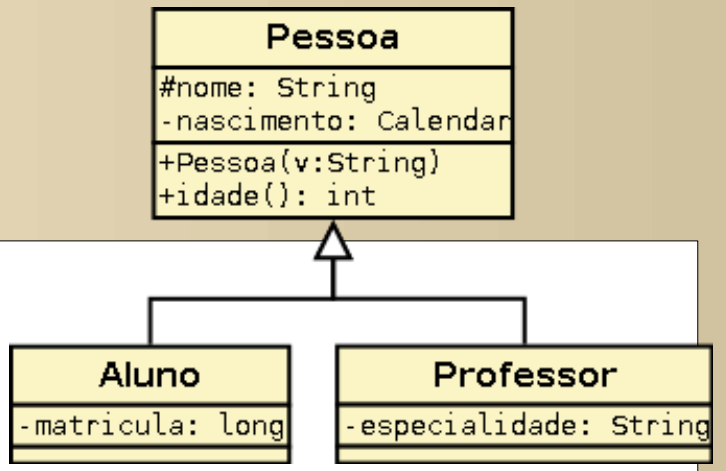
Construtor em Sub-Classes

Conceitos

- Construtor da Sub-Classe deve invocar o construtor padrão da Superclasse
- Caso não esteja declarada, toda linguagem declara nos construtores a invocação ao construtor padrão da Superclasse
- Ausência do construtor padrão na Superclasse provoca erro de compilação na Subclasse



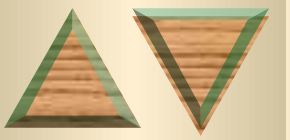
Construtor



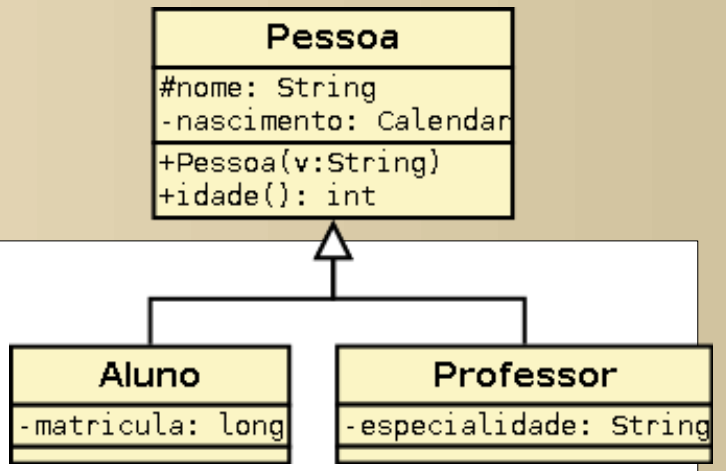
Conceitos

```
public class Pessoa {
 protected String nome;
 private Calendar nascimento =
  new GregorianCalendar();
 public Pessoa(String nome) {
  this.nome = nome;
 }  //gets e sets    }
```

```
public class Aluno extends Pessoa{
 private long matricula;
 //gets e sets    }//não compila
```

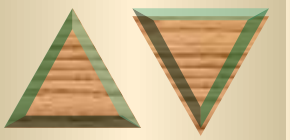


Construtor



Conceitos

```
public class Pessoa {
 protected String nome;
 private Calendar nascimento =
  new GregorianCalendar();
 public Pessoa(String nome) {
  this.nome = nome;
 }  //gets e sets    }
public class Aluno extends Pessoa{
   private long matricula;
   public Aluno(String nome) {
      super(nome);
   }
   //gets e sets
}
```

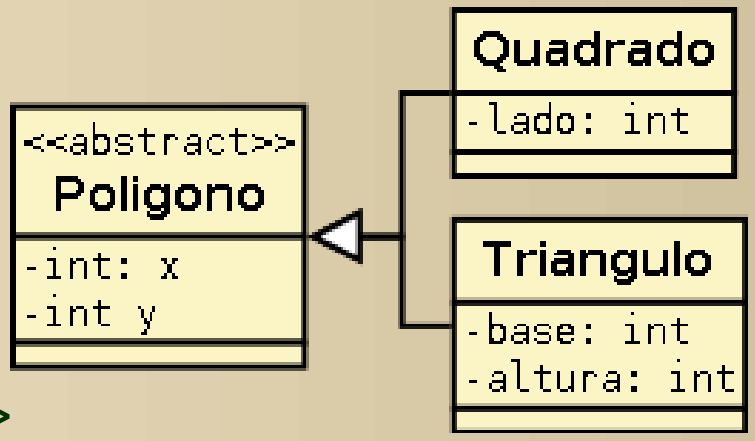


- Conceitos
- Abstrata
- Interface

Herança

Prof.Dr. Enzo Seraphim

Classes abstratas

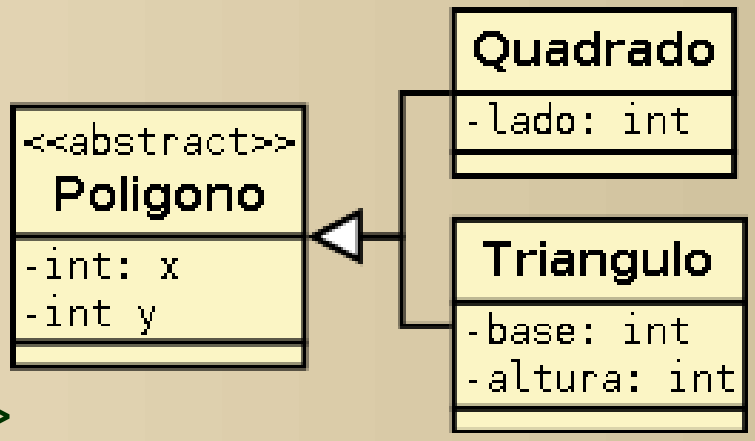


- Classe que não pode ter instâncias diretas.
- Estereótipo: <<abstract>>
- Instâncias ocorrem nas subclasses concretas

Abstrata

```
public abstract class Poligono {
    private int x;
    private int y;
    //gets e sets    }
public class Quadrado extends Poligono {
    private int lado;
    //gets e sets    }
public class Triangulo extends Poligono {
    private int base;
    private int altura;
    //gets e sets    }
```

Classes abstratas



- Classe que não pode ter instâncias diretas.
- Estereótipo: <<abstract>>
- Instâncias ocorrem nas subclasses concretas

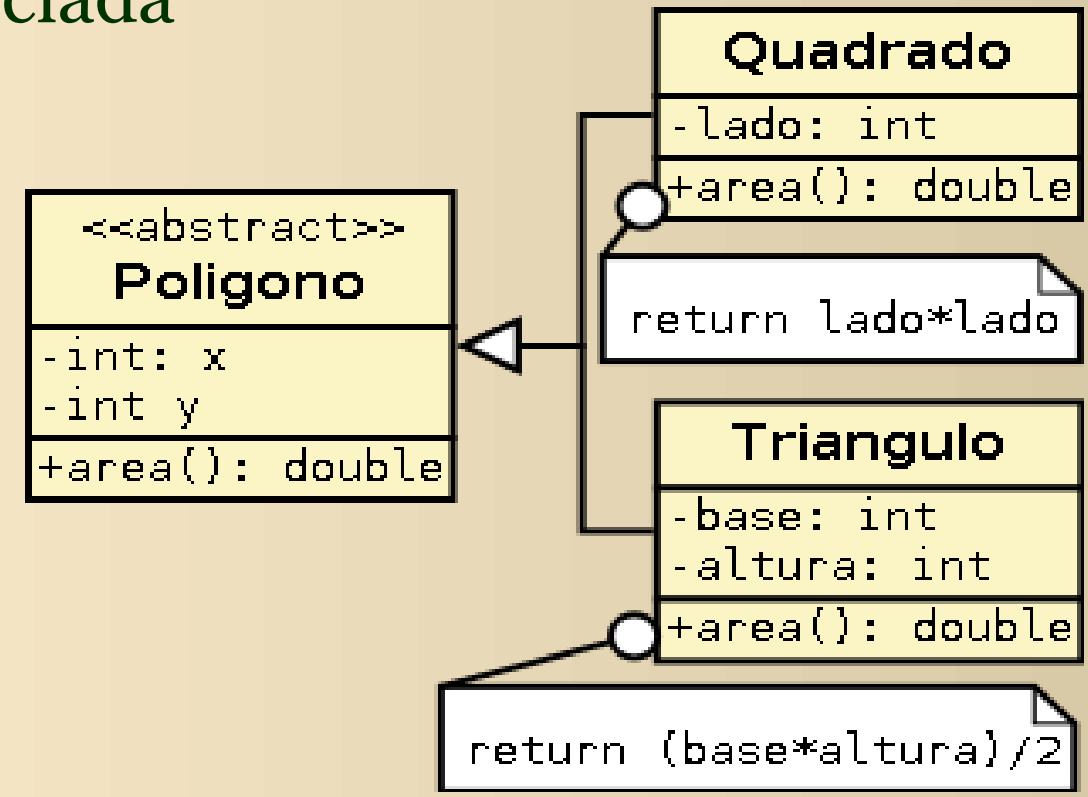
```
//Poligono p = new Poligono(); // não compila
Poligono p = new Poligono(){};
Quadrado q = new Quadrado();
Triangulo t = new Triangulo();
```

Abstrata

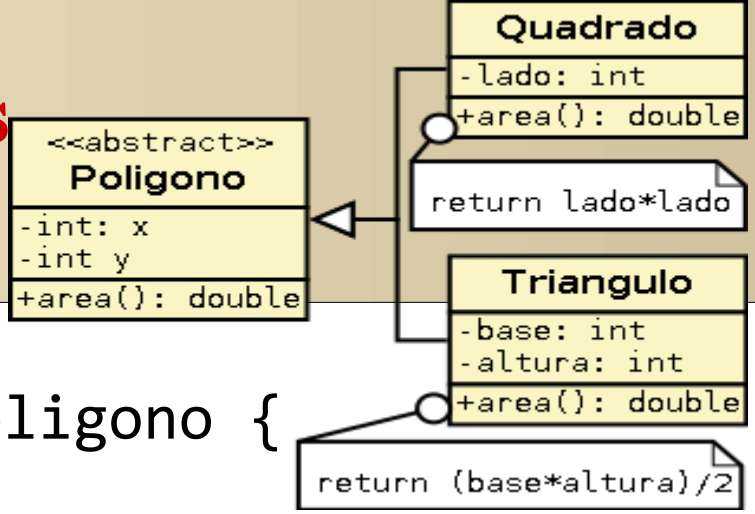
Operações abstratas

- Operação sem implementação na Superclasse.
- SuperClasse com operação abstrata é abstrata
- Sub-classe deve implementar operação para ser instanciada

Abstrata



Operações abstratas



Abstrata

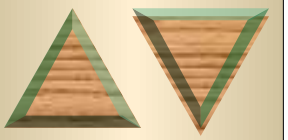
```
public abstract class Poligono {
 private int x;
 private int y;
 public abstract double area();
 //gets e sets   }
public class Triangulo extends Poligono {
 private int base;
 private int altura;
 @Override public double area() {
  return (base*altura)/2;   }
 //gets e sets      }
public class Quadrado extends Poligono {
   private int lado;
   @Override public double area() {
      return lado*lado;   } //gets e sets    }
```

Classes Anônima

Abstrata

- Classe não declarada explicitamente mas representa o comportamento de uma classe abstrata (ou interface)
- Compilador gera uma classe sem nome que herda o comportamento da classe abstrata.
- Uso exclusivo, pois nova declaração criar uma nova classe anônima.

```
Poligono p1 = new Poligono() {
    @Override public double area() {
    return 0; } };
System.out.println("p1="+p1.area());
Poligono p2 = new Poligono() {
    @Override public double area() {
    return 0; } };
System.out.println("p2="+p2.area());
```



- Conceitos
- Abstrata
- Interface

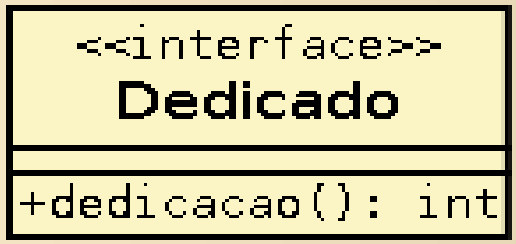
Herança

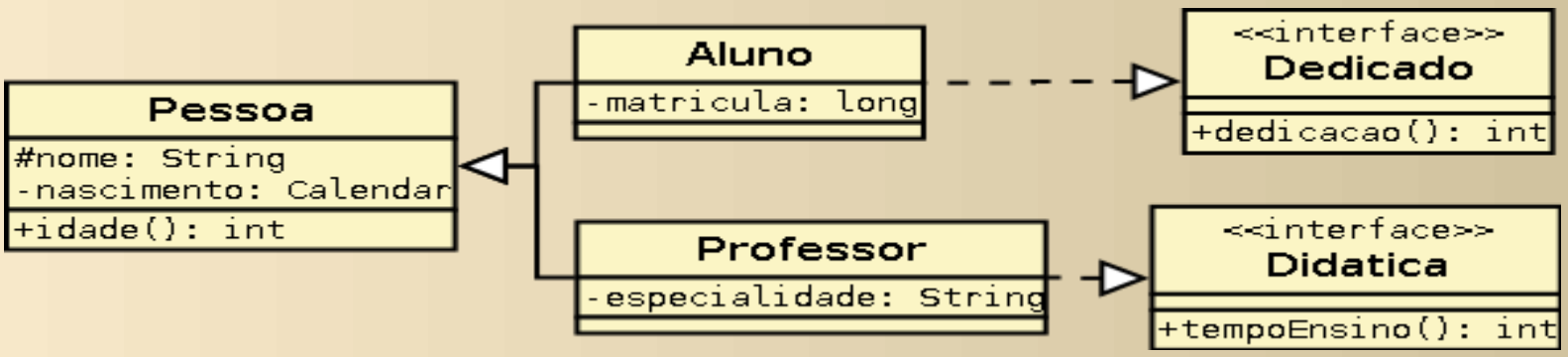
Prof.Dr. Enzo Seraphim

Interface

Interface

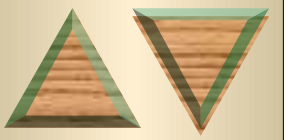
- É um tipo abstrato que é usado para especificar um comportamento que as classes devem implementar.
- Estereótipo: <<interface>>
- Herança: reta tracejada
- Somente pode haver declarações de:
  - Métodos abstratos (sem implementação)
  - Métodos concretos estáticos
  - Atributos estáticos e constantes (final)





Interface

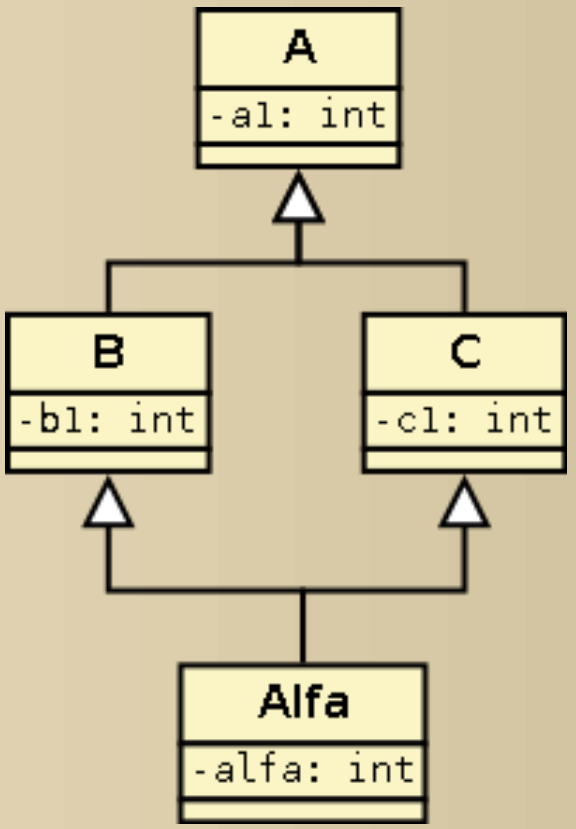
```
public interface Dedicado {
 public abstract int dedicacao(); }
public interface Didatica {
 public abstract int tempoEnsino(); }
public abstract class Pessoa { }
public class Aluno extends Pessoa
  implements Dedicado {
 @Override
 public int dedicacao(){return 100;}}
public class Professor extends Pessoa
  implements Didatica {
 @Override
 public int tempoEnsino(){return 10;}}
```



Problema do Diamante na Herança Múltipla de atributos

Interface

- Herança múltipla de atributos foi abolida das linguagens orientadas a objetos
- Quantos atributos a1 existem nos objetos da classe Alfa?



Prof.Dr.
Enzo Seraphim
seraphim@unifei.edu.br
IESTI/UNIFEI

Herança