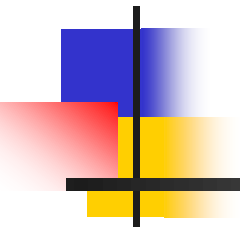
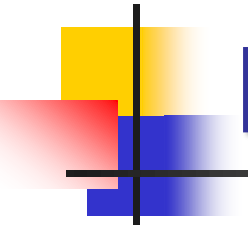


Objetos e Classes

Alberto Costa Neto

DComp - UFS



Roteiro

- Objetos
- Classe
- Implementando Classes
- Manipulando Objetos
- Tempo de Vida

Objetos

- Lembrando... o que é um "objeto"?
  - Entidade que existe no tempo e no espaço
  - Exemplos de objetos...
    - Aluno
    - Curso
    - Turma
    - ...

Objetos

- O que os objetos na seguinte lista tem em comum ?

1 - barraca
2 - caverna
3 - barracão
4 - garagem
5 - celeiro
6 - casa
7 - edifício

- E estes?

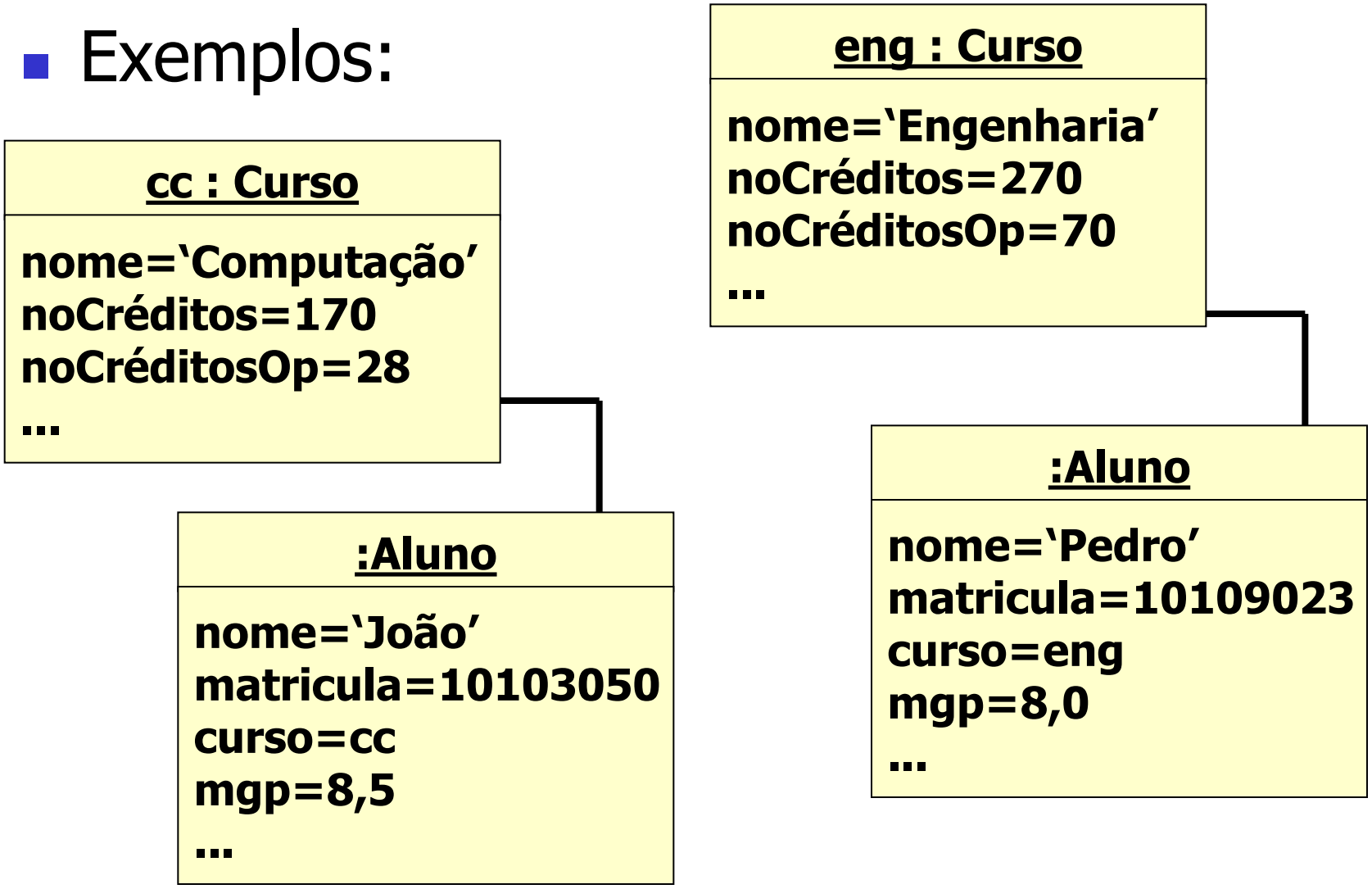
1 - microscópio
2 - óculos
3 - telescópio
4 - binóculo

Objetos

- Objetos são descritos por suas características
  - Como podemos descrever...
    - **Um "Curso"?**
    - **Um "aluno"?**
    - **Uma "turma"?**

Objetos

- Exemplos:



Objetos

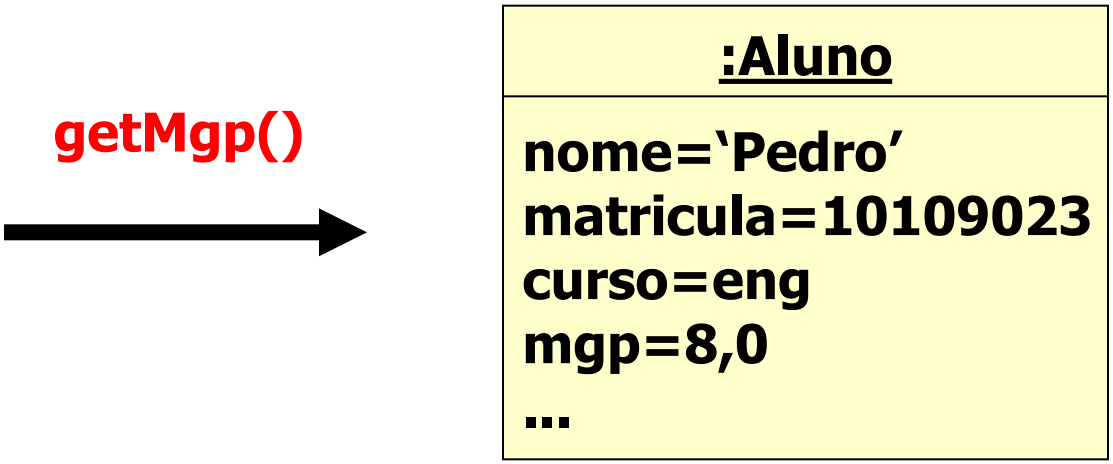
- Operações podem ser realizadas sobre os objetos

matricularEmCurso(10103078, cc)

:Aluno
nome='Carolina' **matricula=** **curso=** **mgp=** **...**

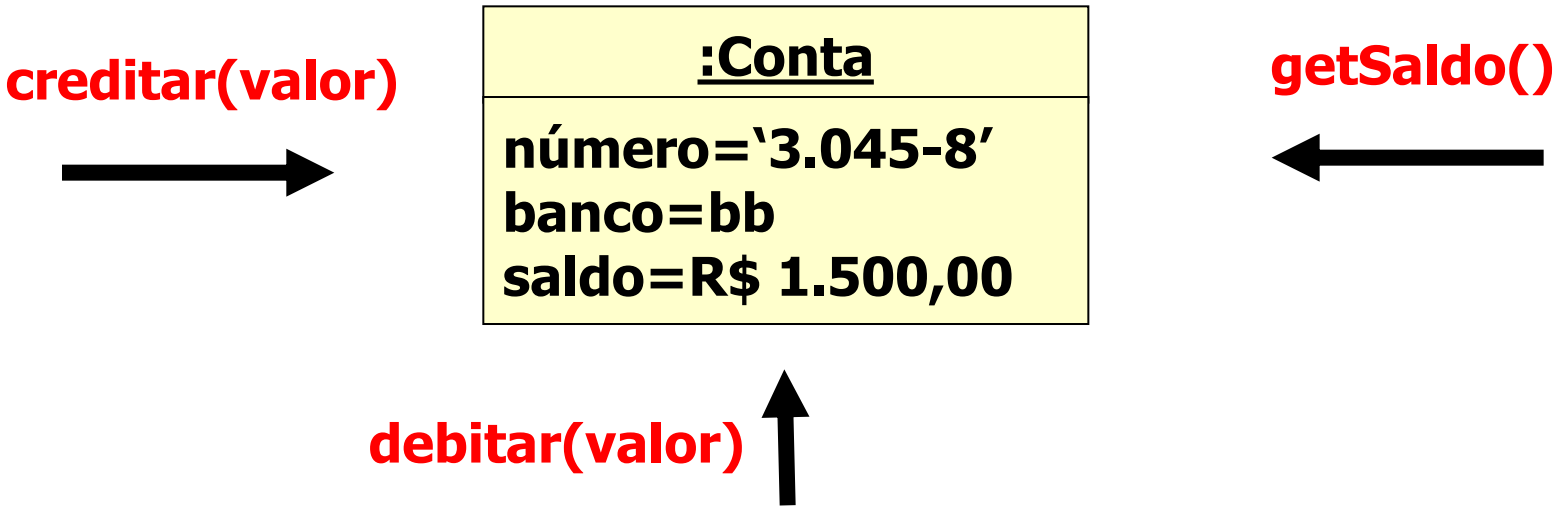
Objetos

- Operações podem ser realizadas sobre os objetos



Objetos

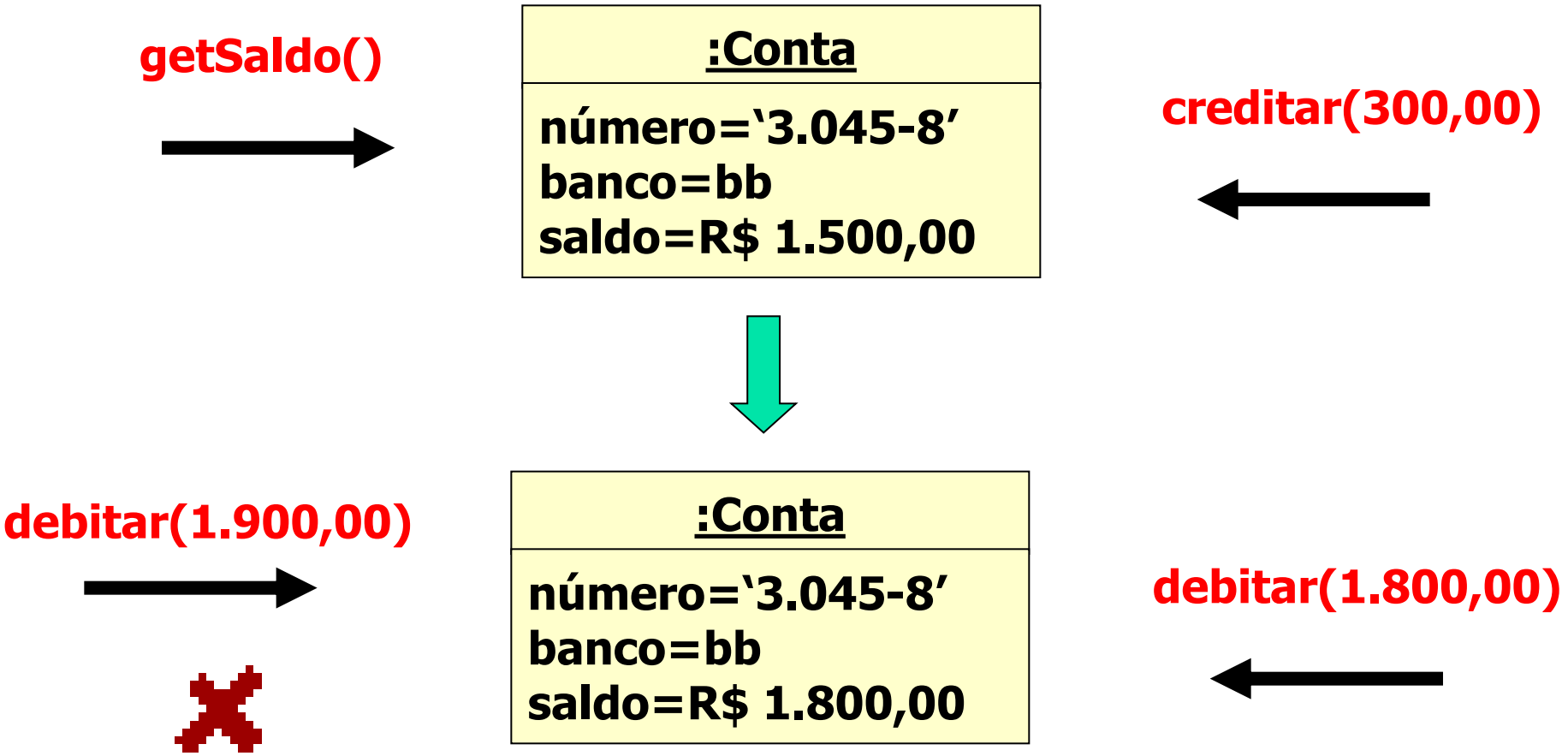
- Outro exemplo

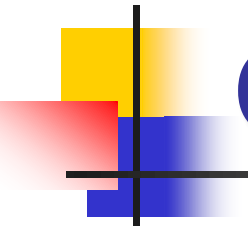


- Suas propriedades ou campos
  - Formam o **estado do objeto**
- O comportamento do objeto é definido pelo seu estado e operações possíveis

Objetos

- Qual o comportamento do objeto?



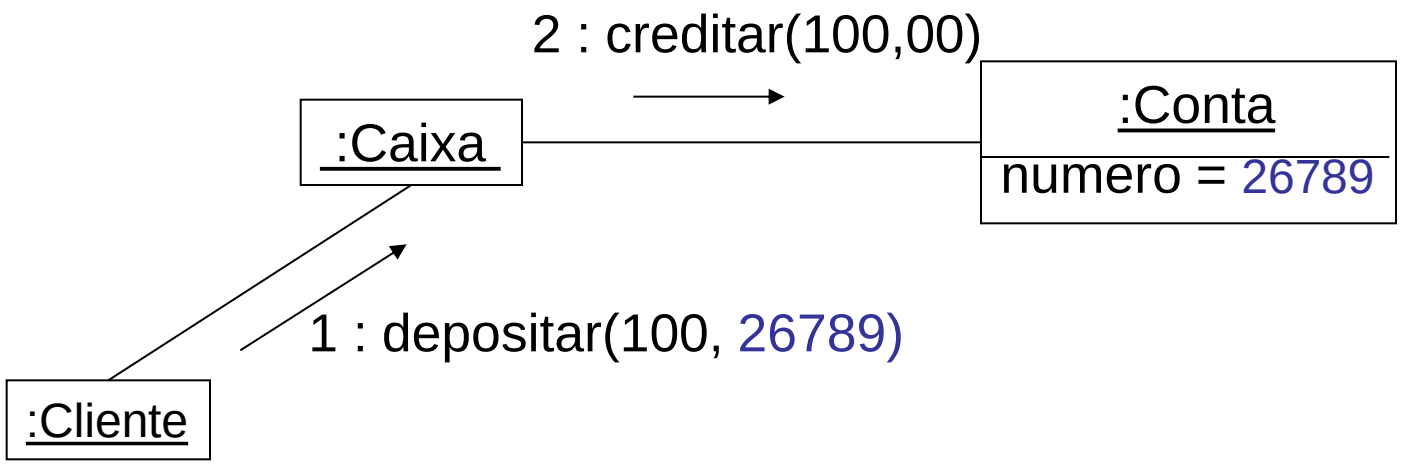


Objetos

Objeto = Estado + Comportamento + Identidade

Objetos

- Objetos interagem por meio de troca de mensagens
- Por que é importante a troca de mensagens?
  - Interação = Colaboração
  - Colaboração ➔ ações

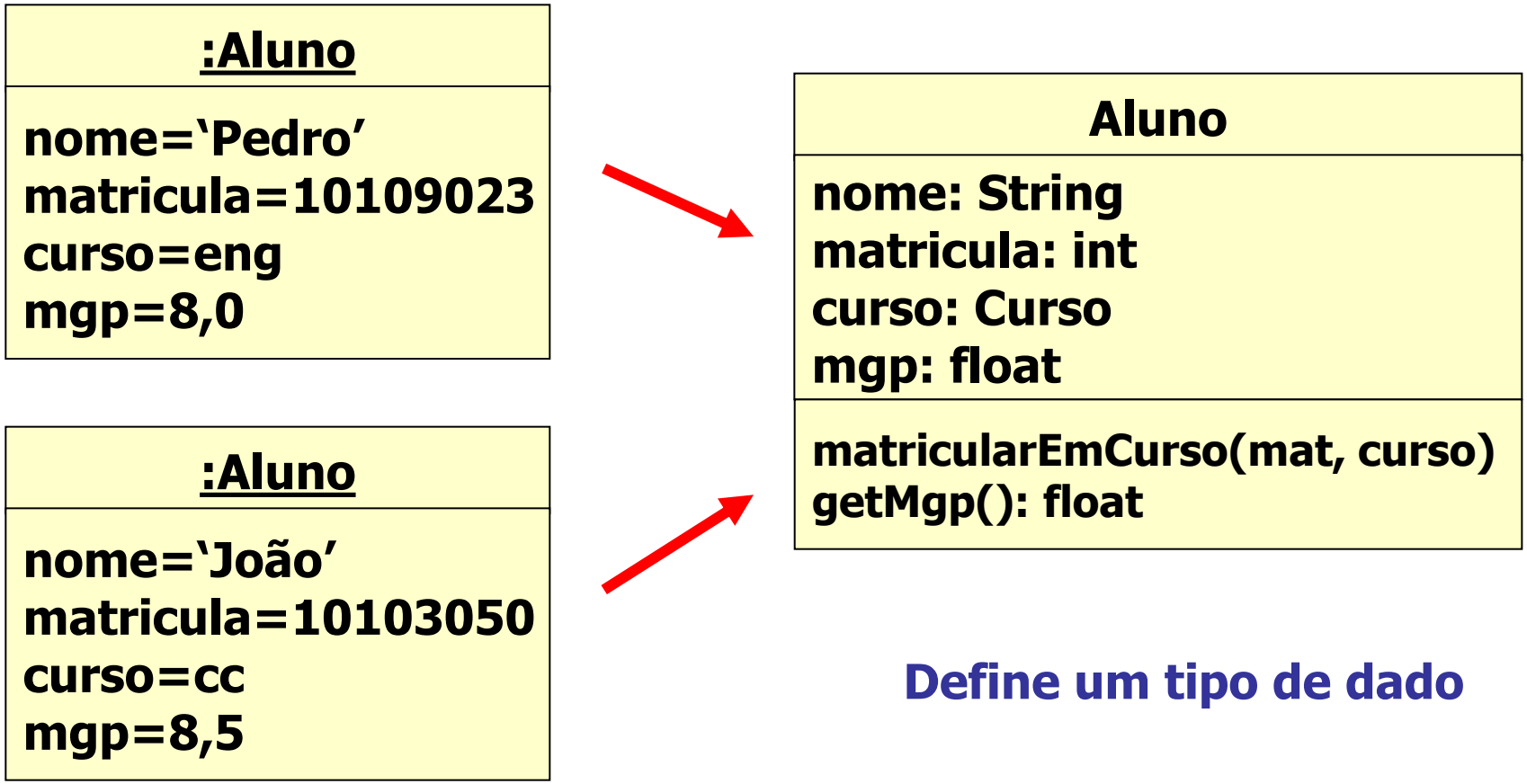


Classes

- Alguém lembra o que é uma classe?
  - Uma classe é um *template* (molde/padrão) que permite criar objetos (instâncias)
  - Uma classe descreve um grupo de objetos
    - com propriedades semelhantes
    - comportamentos semelhantes
    - relacionamentos comuns com outros objetos

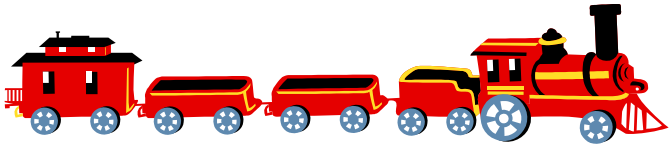
Classes

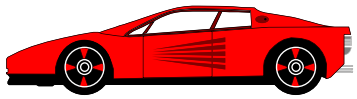
- Abstração que representa a estrutura e comportamento comum

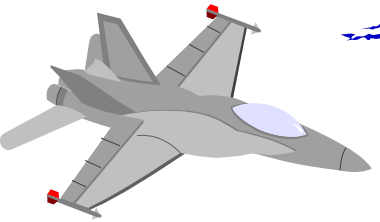


Define um tipo de dado

Classes







Classificação

instanciação

Classe

Atributos

MeioTransporte
capacidade: int **velocidadeMaxima: int** **...**
irPara(local) **addPassageiro(passageiro)**

Operações

Objetos/Instâncias

Classes

- Atributos
  - São as propriedades nomeadas de um objeto
    - Conta Bancária:
      - saldo, data abertura, titular, etc.
    - Pessoa:
      - nome, endereço, telefone, etc.
    - Livro:
      - isbn, nome, autores, data publicação, edição, etc.
    - Todos os atributos são sempre importantes?

Classes

- Operações
  - Definem o comportamento possível para o objeto
    - Correspondem às ações a serem executadas
  - Nomenclatura utilizada
    - Chamada de método
    - Chamada de função
    - Passar uma mensagem

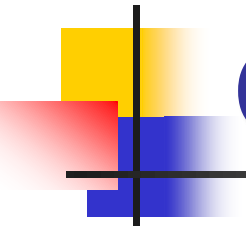
Classes

- Teoricamente...
  - Operação <> Método
    - Operação = especificação da ação (assinatura)

```
public static void soma(int num1, int num2);
```

    - Método = implementação da ação

```
public static void soma(int num1, int num2) {
        int resultado = num1 + num2;
       System.out.println("Soma = " + resultado);
  }
```



Classes

- Detalhes...
  - Exemplos de passagem de parâmetro e retorno

```
public class Teste1 {
    public static void main(String[] args) {
        soma();      // sem a passagem de parâmetro e sem retorno
    }
    public static void soma() {
        Scanner e = new Scanner (System.in);
        int num1, num2, resultado;
        System.out.println("Digite o primeiro número:");
        num1 = e.nextInt();
        System.out.println("Digite o segundo número:");
        num2 = e.nextInt();
        resultado = num1 + num2;
        System.out.println("Soma = " + resultado);
   }
}
```

```
public class Teste2 {
   public static void main(String[] args) {
        Scanner e = new Scanner (System.in);
        int num1, num2;
        System.out.println("Digite o primeiro número:");
        num1 = e.nextInt();
        System.out.println("Digite o segundo número:");
       num2 = e.nextInt();
        soma(num1, num2);        //com parâmetros e sem retorno
    }
    public static void soma(int num1, int num2) {
        int resultado = num1 + num2;
        System.out.println("Soma = " + resultado);
    }
}
```

```
public class Teste3 {
   public static void main(String[] args) {
         int resultado = soma();        //sem parâmetro, com retorno,
         System.out.println("Soma = " + resultado);
   }
   public static int soma() {
         Scanner e = new Scanner (System.in);
         int num1, num2;
         System.out.println("Digite o primeiro número:");
         num1 = e.nextInt();
         System.out.println("Digite o segundo número:");
         num2 = e.nextInt();
         return num1 + num2;
   }
}
```

```
public class Teste2 {
    public static void main(String[] args) {
        Scanner e = new Scanner (System.in);
        int num1, num2;
        System.out.println("Digite o primeiro número:");
        num1 = e.nextInt();
        System.out.println("Digite o segundo número:");
        num2 = e.nextInt();
        int resultado = soma(num1, num2);   //com par., com retorno
        System.out.println("Soma = " + resultado);
    }
    public static int soma(int num1, int num2) {
            return num1 + num2;
    }
}
```

Implementando Classes

Produto
codigo: int nome: String quantidade: int preco:float
Produto (int codigo, String nome, int quantidade, float preco) getCodigo(): int getNome(): String getQuantidade(): int getPreco(): float setCodigo(int codigo) setNome(String nome) setQuantidade(int quantidade) setPreco(float preco) obterValorEstoque(): float

Implementando Classes

```
public class Produto {

   /* Atributos da Classe */

   private int codigo;
   private String nome;
   private int quantidade;
   private float preco;
```

Implementando Classes

```
/* construtor da classe */

public Produto (int codigo, String nome, int quantidade,
                        float preco) {

    this.codigo = codigo;
    this.nome = nome;
    this.quantidade = (quantidade > 0)?  quantidade:0;
    this.preco = preco;

}
```

Conceitos

- Construtor é o método da classe que é executado quando deseja-se criar uma instância da classe (objeto)
- Responsável por alocar recursos
- Permite inicializar os atributos do objeto
- Toda classe possui ao menos um construtor
- O nome do construtor é o mesmo nome da classe e não possui retorno

Manipulando Objetos...

- Cada objeto tem acesso a uma referência para si próprio, através da palavra-chave **this**
- Através de **this** é possível:
  - Diferenciar entre parâmetros de métodos e atributos de instância com o mesmo nome
  - Retornar uma referência para o próprio objeto em um de seus métodos

Implementando Classes

```
/* métodos acessores */
public int getCodigo() {
   return this.codigo;
}
public String getNome() {
   return this.nome;
}
public int getQuantidade() {
   return this.quantidade;
}
public float getPreco() {
   return this.preco;
}
```

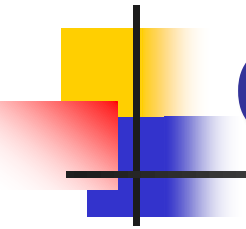
Conceitos

- O método acessor é a única forma de ter acesso aos dados internos (privados)

- Protegem a representação dos dados.

Implementando Classes

```
/* métodos modificadores ou mutantes */
public void setCodigo(int codigo) {
        this.codigo = codigo;
}
public void setNome(String nome) {
        this.nome = nome;
}
public void setQuantidade(int quantidade){
        this.quantidade = (quantidade > 0)?  quantidade:0;
}
public void setPreco(float preco) {
        this.preco = preco;
}
```



Conceitos

- O método modificador ou mutante permite alterar o estado interno de um objeto.

Implementando Classes

- Métodos do negócio são necessários para implementar as funcionalidades requeridas pela aplicação

```
/* método do negócio */

public float obterValorEstoque() {
   return this.preco * this.quantidade;
}
```

Manipulando Objetos...

- O que faz este código?

```
Produto p1 ;

p1 = new Produto(1, "Resma Papel", 10, 14.5);

Produto p2 = new Produto(2, "Lápis", 15, 1.0);

System.out.println ( p1.getNome() );

p1.setPreco(16.0);

System.out.println( p1.getPreco() );
```

Manipulando Objetos...

- O que faz este código?

Declara a variável p1

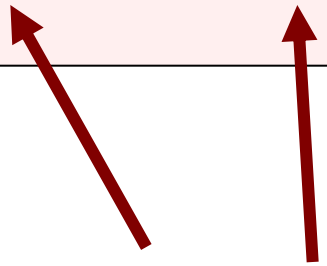
```
Produto p1 ;
p1 = new Produto(1, "Resma Papel", 10, 14.5);
```

New cria uma instância de Produto a partir de seu construtor

Manipulando Objetos...

- O que faz este código?

```
Produto p2 = new Produto(2, "Lápis", 15, 1.0);
```



Declaração + Inicialização

Manipulando Objetos...

- O que faz este código?

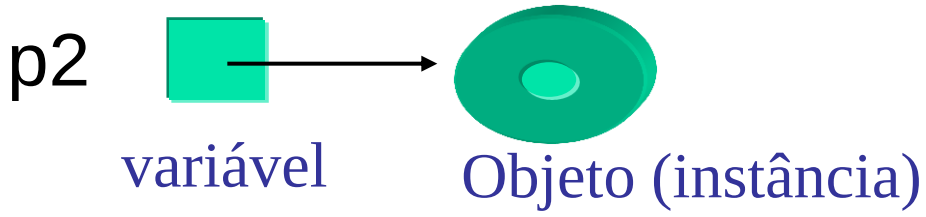
Um método sempre é invocado de um objeto alvo.

```
System.out.println ( p1.getNome() );

p1.setPreco(16.0);

System.out.println( p1.getPreco() );
```

Entendendo melhor...

- Importante

```
Produto p2 = new Produto(2, "Lápis", 15, 1.0);
```



Em java:
A variável não é (não contém) o objeto!!!
Uma variável contém uma referência a um objeto!!!

Entendendo melhor...

- Variável Primitiva x Variável de Referência
  - Variável primitiva ➔ um valor
  - Variável de Referência ➔ como chegar a um objeto

int quantidade = 20;

Produto p1 = new Produto();

Entendendo melhor...

- Declaração, criação e atribuição de objetos

Declara a variável de referência p1

```
Produto p1 ;
p1 = new Produto(1, "Resma Papel", 10, 14.5);
```

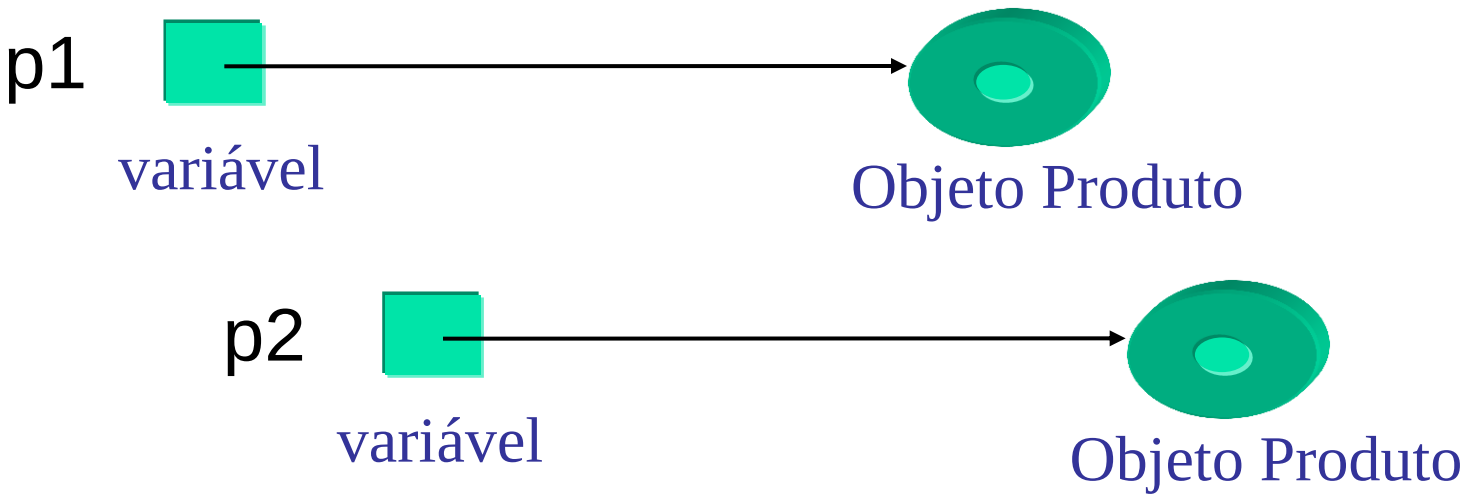
New cria uma instância de Produto a partir de seu construtor

Vincula objeto à referência

Entendendo melhor...

- Referências

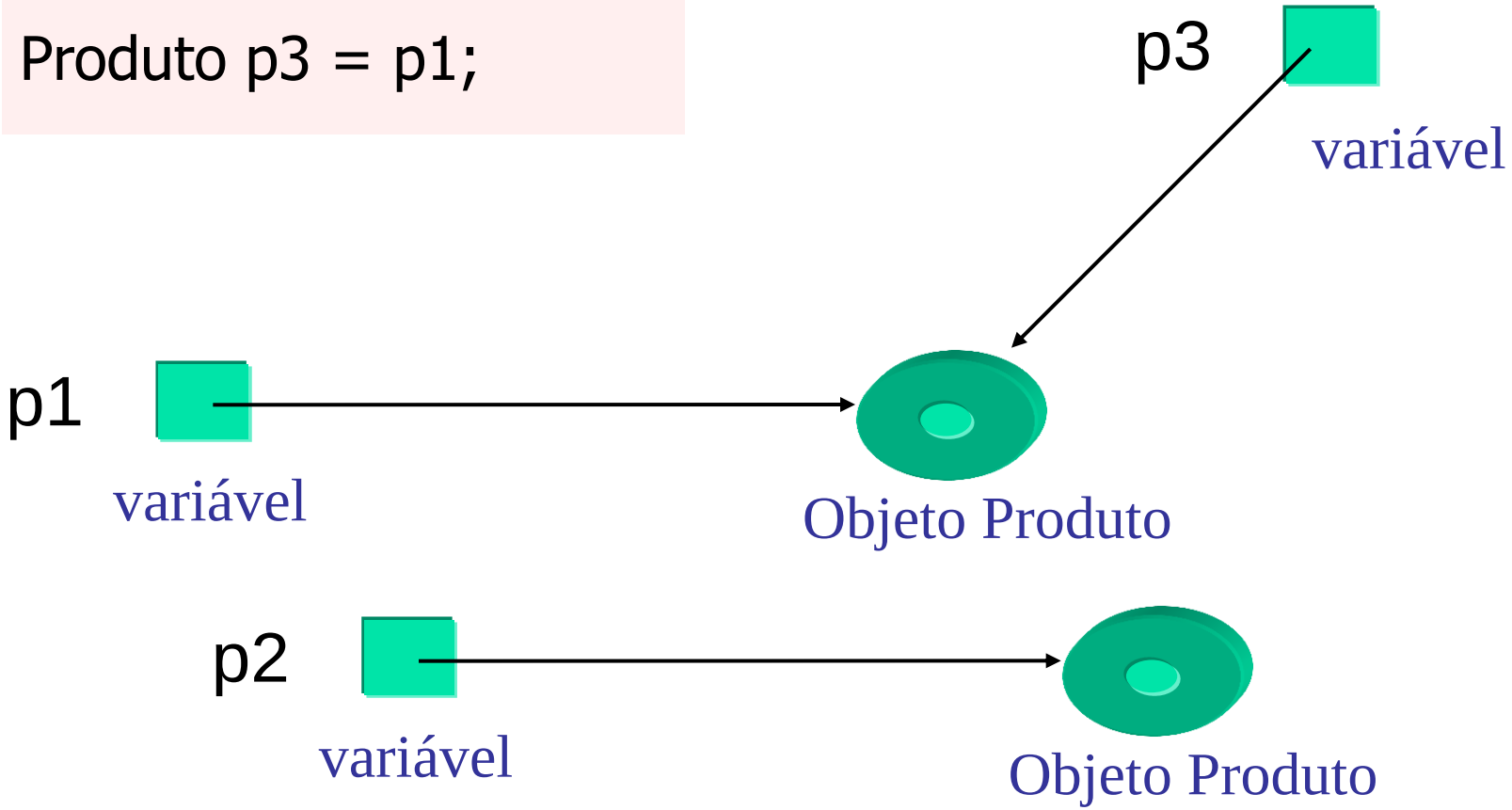
```
Produto p1 = new Produto(1, "Resma Papel", 10, 14.5f);

Produto p2 = new Produto(2, "Lápis", 15, 1.0f);
```



Entendendo melhor...

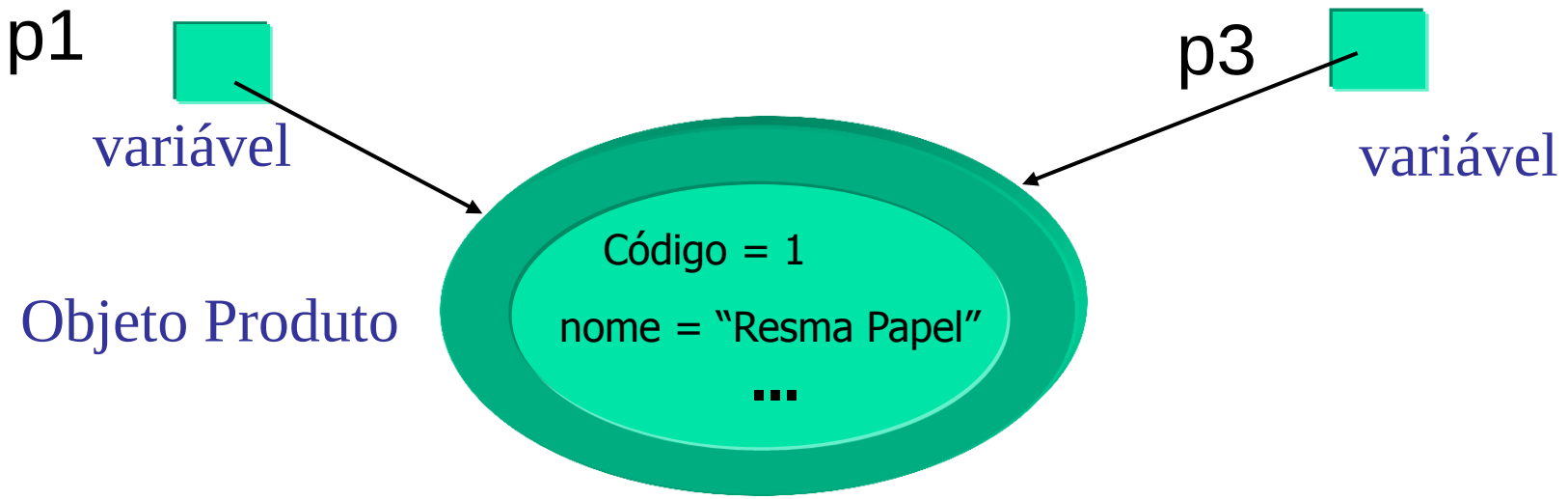
- p3 referencia qual objeto?

```
Produto p3 = p1;
```



Entendendo melhor...

- Então...



Fazendo...

```
p3.setNome("Papel");
```

Qual o resultado?

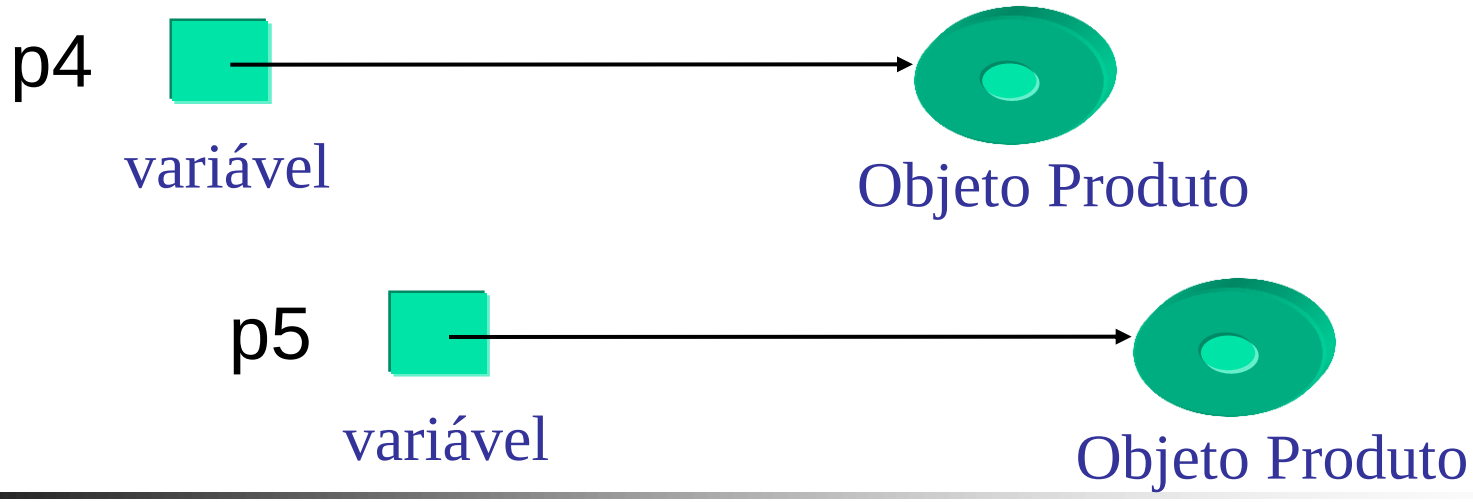
```
? = p1.getNome();
```

Entendendo melhor...

- Outro exemplo

```
Produto p4 = new Produto(3, "Borracha", 15, 2.5f);

Produto p5 = new Produto(3, "Borracha", 15, 2.5f);
```

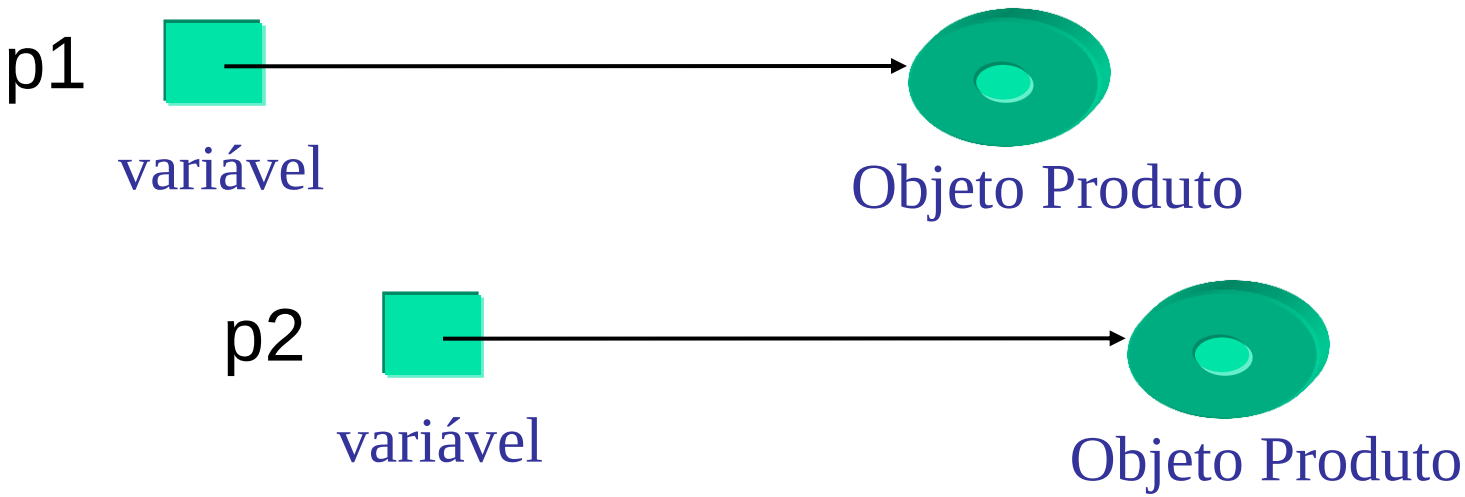
p4 == p5 ? ✗



Entendendo melhor...

- Outro exemplo

```
Produto p1 = new Produto(1, "Resma Papel", 10, 14.5f);

Produto p2 = new Produto(2, "Lápis", 15, 1.0f);
```



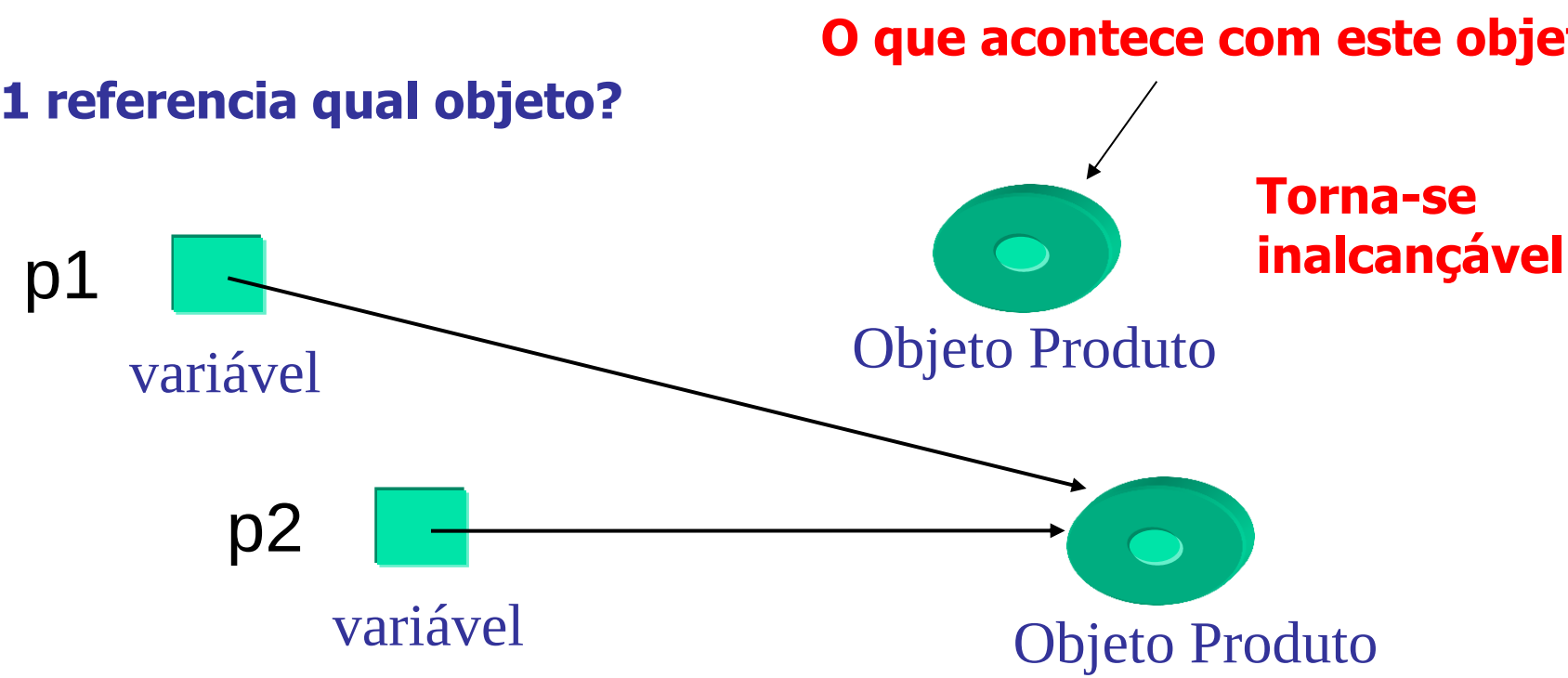
Entendendo melhor...

- Outro exemplo

```
p1 = p2;
```

O que acontece com este objeto?

P1 referencia qual objeto?



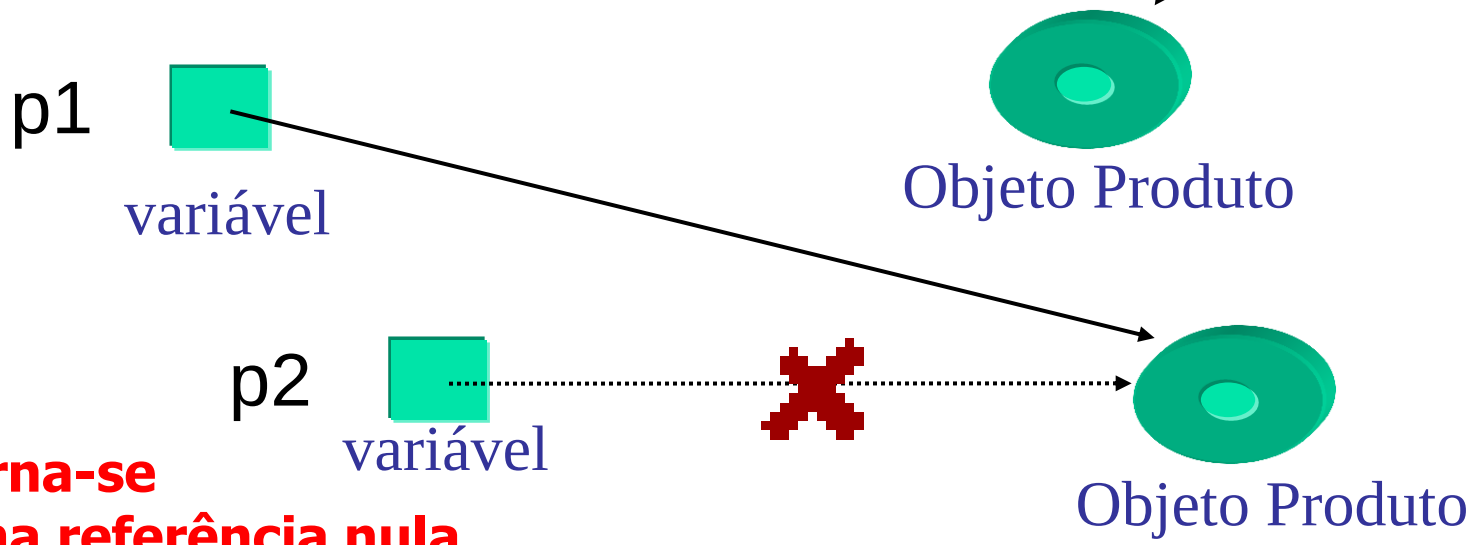
Entendendo melhor...

- Se fizermos...

```
p2 = null;
```

Objeto anteriormente abandonado

P2 referencia qual objeto?

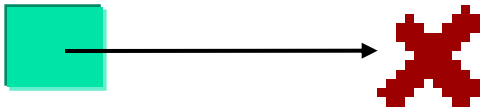


Torna-se
Uma referência nula

Entendendo melhor...

- null
  - Indica uma referência para nenhum objeto

```
Produto p6 = null;
```



Variável de Referência

  - Qualquer objeto do tipo definido para a referência pode ser vinculado à mesma

```
p6 = new Produto();
```

Entendendo melhor...

- null

```
Produto p7 = null;
```

```
p7.getNome();
```

NullPointerException

Entendendo melhor...

- Variável Local x Variável de Instância

```
int idade = 15;

boolean amigoDoDono = true;

if ((idade < 18) && !amigoDoDono)

  System.out.println("Não pode entrar");

else

  System.out.println("Pode entrar");
```

```
public class Aluno {

    String nome;
    int idade;
    Curso curso;
    int matricula;

    public Aluno(...){
       ...
    }

    ...
}
```

Entendendo melhor...

- Inicialização de variáveis
  - Variáveis de instância do tipo primitivo são automaticamente inicializadas pelo default
    - Boolean → false
    - Demais tipos → 0
  - Variáveis locais ou temporárias não são automaticamente inicializadas por default
  - Variáveis que contêm referências não são automaticamente inicializadas por default

Implementando Classes

- Outro exemplo

Conta
numero: int saldo: double
Conta (numero: int) getNumero(): int getSaldo(): double creditar(valor: double) debitar(valor: double)

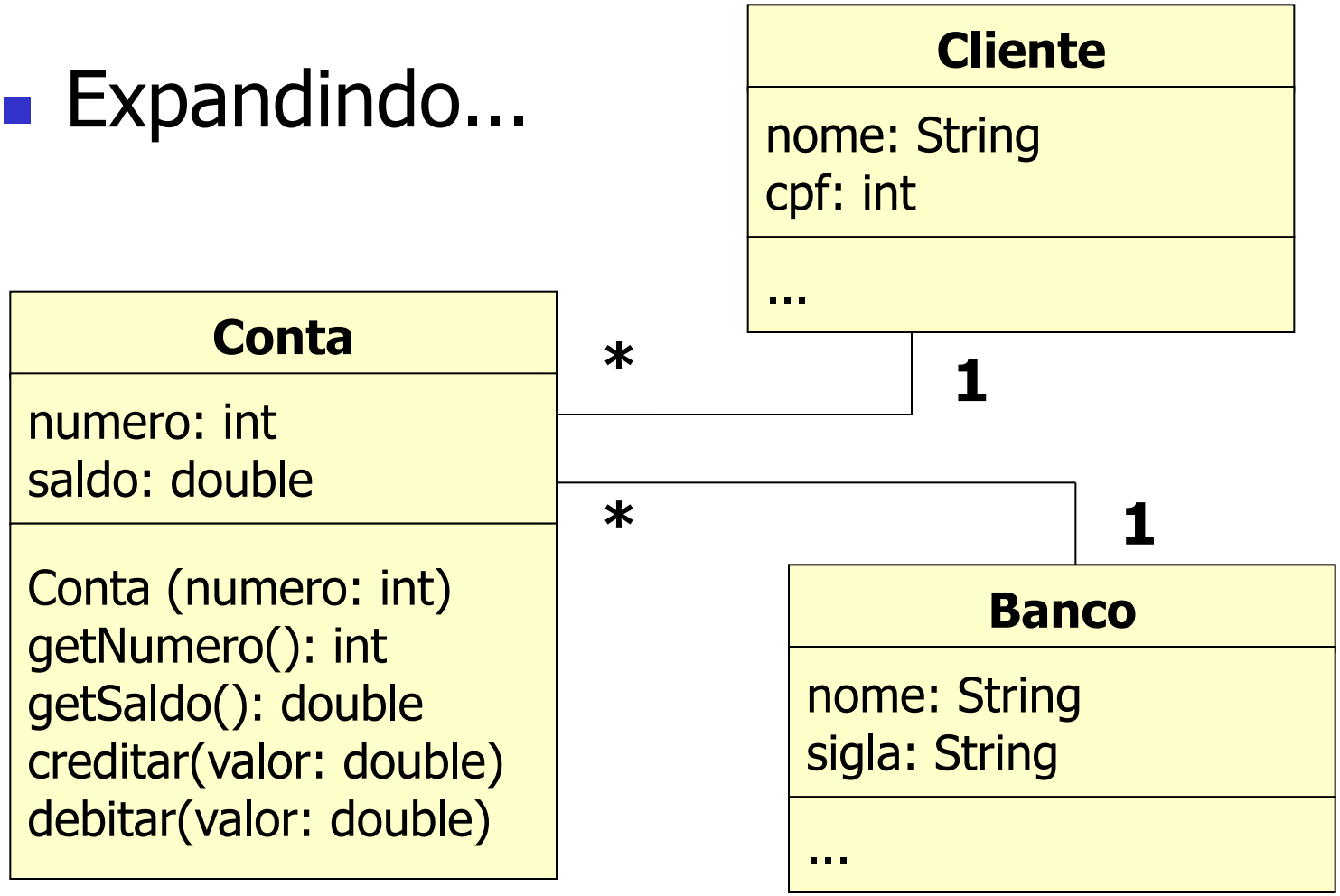
```
public class Conta {

   private int  numero;
   private double   saldo;

   public Conta (int num) {
        numero = num; saldo = 0;
   }

   public int getNumero() {return numero;}

   public double getSaldo() {return saldo;}

   public void creditar (double valor) {
       saldo = saldo + valor;
   }

   public void debitar (double valor) {
       int novoSaldo = saldo - valor;
       if (novoSaldo >= 0) saldo = novoSaldo;
   }
}
```

Manipulando objetos

```
Conta c1 = new Conta(26789);

c1.creditar(100);

System.out.println(c1.getSaldo());

c1.debitar(150);

System.out.println(c1.getSaldo());

c1.creditar(100);

c1.debitar(150);

System.out.println(c1.getSaldo());
```

Implementando Classes

- Expandindo...



Implementando Classes

- Novas variáveis de instância são criadas

```
public class Conta {

    private int  numero;
    private double   saldo;

    private Cliente titular;
    private Banco banco;
    ...
```

Métodos acessores e mutantes...

```
public class Conta {
   ...
   public Cliente getTitular( ) {
            return this.titular;
   }

   public Banco getBanco( ) {
            return this.banco;
   }

   public void setTitular(Cliente cliente) {
            this.titular = cliente;
   }

   public void setBanco(Banco banco) {
          this.banco = banco;
  }
   ...
```

Manipulando objetos

```
Conta c4 = new Conta(26789);

c4.creditar(300);

System.out.println(c1.getSaldo());

Banco b1 = new Banco ( “Banco do Brasil”, “BB”);

c4.setBanco(b1);

System.out.println(c4.getBanco().getNome());

Cliente cl = new Cliente(“Pedro”, 123123123);

c4.setCliente(cl);

System.out.println(c4.getCliente().getNome());
```

Manipulando objetos

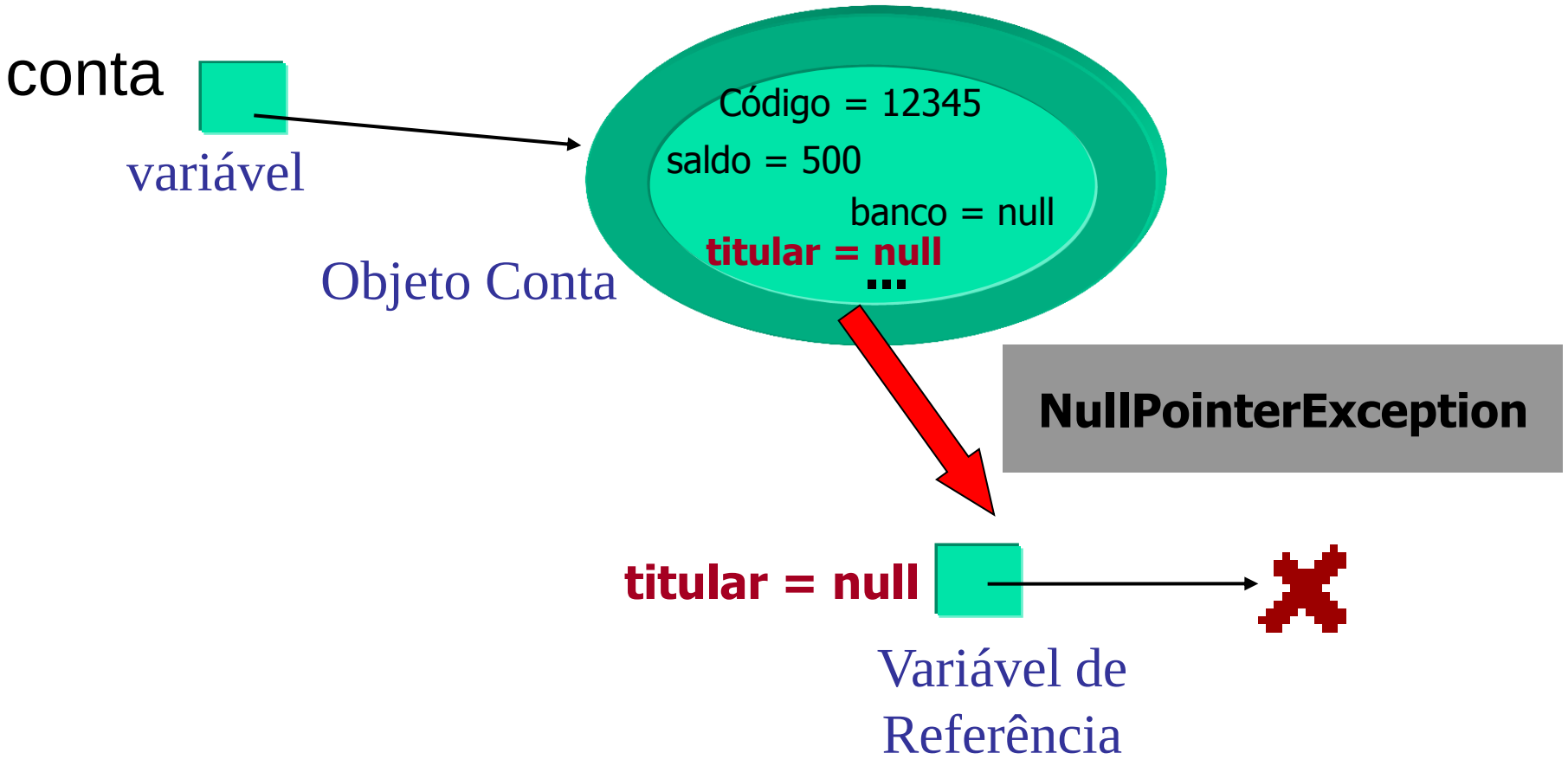
- O que acontece ao executar o código?

```
Conta conta = new Conta(12345);

conta.creditar(500);

System.out.println(conta.getCliente().getNome());
```

- Por que não funcionou?

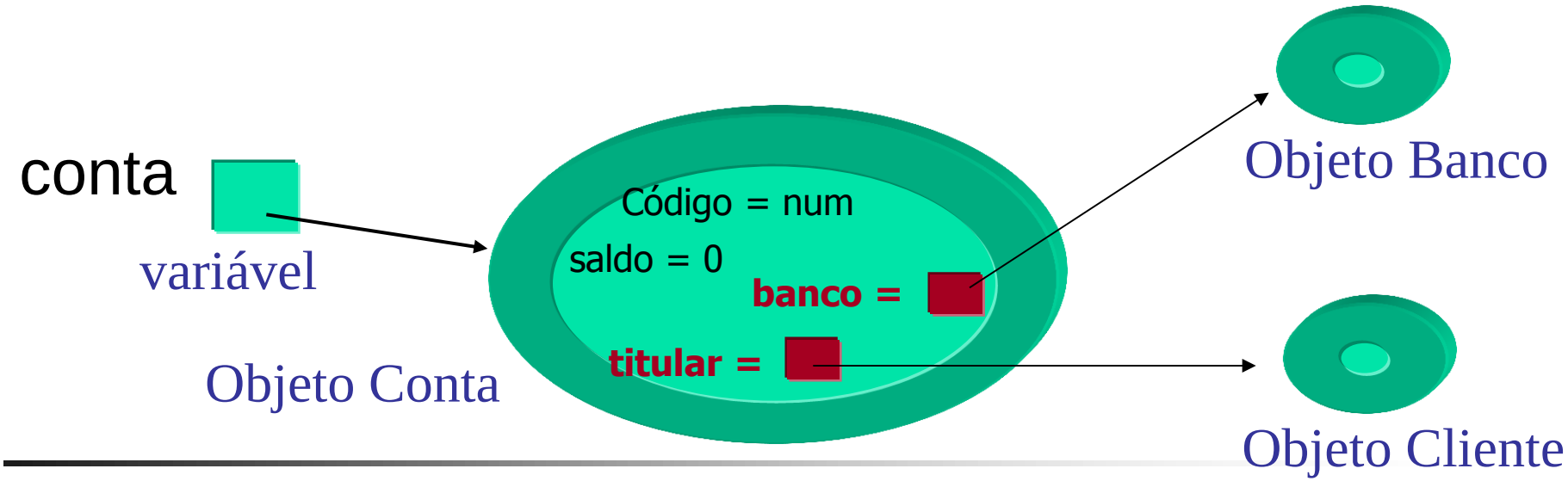
Manipulando objetos

- Qual a situação atual do objeto conta?



Qual uma possível solução?

```
public class Conta {
      ...
      public Conta (int num) {
        numero = num; saldo = 0;
        this.banco = new Banco( );
        this.titular = new Cliente( );
    }
    ...
```



Tempo de vida

- A criação dos objetos é dinâmica (através do *new*)
- O tempo de vida de um objeto termina quando não existe mais nenhuma referência para ele
- Objetos cujos tempos de vida terminam são eliminados automaticamente (*garbage collected*)

Dever de Sala

1) Implemente a Classe Livro com os atributos código, nome e quantidade de exemplares

- Crie o construtor
- Crie os métodos acessores
- Crie os métodos modificadores
- Crie uma nova classe chamada TesteLivro
  - Crie 3 instâncias da classe Livro
  - Imprima o número de exemplares de cada livro
  - Altere as quantidades e imprima novamente o número de exemplares.

Dever de Sala

2) Implemente as Classes Produto e Item com os seguintes atributos:

- Produto: código, nome, quantidade em estoque e preço unitário
- Item: código, Produto, quantidade vendida
  - Crie o construtor
  - Crie os métodos acessores
  - Crie os métodos modificadores
  - Crie uma nova classe chamada TesteProdutoItem
    - Crie 2 instâncias da classe Produto
    - Imprima a quantidade em estoque da cada produto
    - Crie 3 instâncias da classe Item
    - Imprima a quantidade do Estoque dos 2 produtos após a venda dos itens
    - Imprima para cada Item qual foi o produto vendido, a quantidade vendida, preço unitario e o valor pago na venda

Referências

- Tony Sintes - Aprenda Programação Orientada a Objetos em 21 dias
  - Capítulo – Dia 1
- Slides "Introdução OO" Prof. Marcos Dósea. UFS. 2010.
- Slides "Objetos e Classes", Profª. Débora. UFS. 2010.
- "ClassesEObjetos" e "ClassesJava". Prof Giovanny . Java. UFS. 2009.