



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
ESTRUTURA DE DADOS ORIENTADA A OBJETOS
ADS1232
PROF. MSC. ANIBAL SANTOS JUKEMURA

**PROGRAMAÇÃO
ORIENTADA A OBJETOS
[JAVA]**

Agenda



- Orientação a Objetos com Java
- Classes / Objetos
- Exemplo
- Exercícios

- Orientação a objetos é uma maneira de programar que ajuda na organização e resolve muitos problemas enfrentados pela programação procedural.
- O problema do paradigma procedural é que não existe uma forma simples de criar conexão forte entre dados e funcionalidades. No paradigma orientado a objetos é muito fácil ter essa conexão através dos recursos da própria linguagem.

- Orientação a objetos vai te ajudar em muito em se organizar e escrever menos, além de concentrar as responsabilidades nos pontos certos, flexibilizando sua aplicação, **encapsulando** a lógica de negócios.
- Outra enorme vantagem, onde você realmente vai economizar montanhas de código, é o **polimorfismo das referências**.

Introdução: Dicionário OO

- Objeto: pode ser definido como uma unidade de *software* constituído de atributos (dados) e de métodos (códigos de funções) que atuam sobre os dados, sendo os representantes das classes.
- Atributos: qualquer propriedade, qualidade ou característica que possa ser atribuída, podendo ser acessado pelo serviços.
- Funções: Blocos de códigos para especificar comportamentos e/ou ações

Introdução: Dicionário OO

- **Serviços:** atividade executada para permitir o acesso a alguns recursos, em OO é um comportamento específico que um objeto deve exibir.
- **Encapsulamento:** restrição de escopo ou visibilidade dos dados do aplicativo que poderão ser acessados pelos serviços.

Introdução: Dicionário OO

- Passagem de mensagem: Possibilita a comunicação entre objetos.
- Herança: mecanismo para trabalhar com similaridades entre as classes como, por exemplo, mamíferos são os homens e o macacos, podendo considerá-los como herdeiros dos atributos da classe mamíferos.

Introdução: Dicionário OO

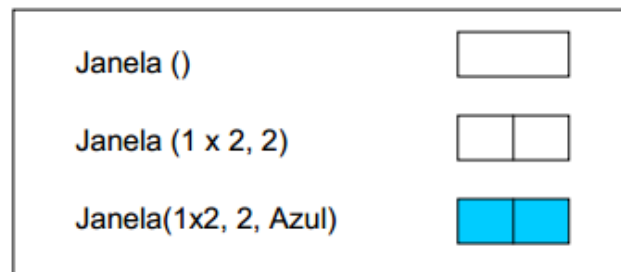
- **Classes:** Em linguagem Java, a unidade de programação é a classe, a partir da qual os objetos são **instanciados** (criados).

Introdução: Dicionário OO

- Superclasse: Classe base para as subclasses (Mamíferos).
- Subclasse: herdeiros das superclasses (Homens e Macacos)
- Superclasse Abstrata: Finalidade apenas de determinação de uma superclasse, não permitindo o manuseio pelas subclasse.

Introdução: Dicionário OO

- Associação: Conexão de idéias através das classes como, por exemplo, diferenças finitas podem resolver problemas de eletromagnetismo.
- Polimorfismo: Relaciona as diferentes formas de um objeto.



Paradigma de Orientação a Objetos



```
6 package javaapplication1;
7
8  /**...4 linhas */
12
13 class OlaMundo{
14     public void mensagem()
15     {
16         System.out.println("HELLO WORLD!");
17     }
18 }
19
20 public class JavaApplication1 {
21
22      public static void main(String[] args) {
23         OlaMundo objOlaMundo = new OlaMundo();
24         objOlaMundo.mensagem();
25     }
26
27 }
```

Problema: listagem de alunos

- Uma certa instituição de ensino está incentivando os seus alunos a participarem de eventos acadêmicos em troca de créditos para obter desconto nas mensalidades.
 - Para participar, o aluno deve comparecer em algum dos eventos cadastrados na instituição e depois escrever um relatório sobre o conteúdo apresentado no evento. Este relatório será avaliado por um professor e receberá uma pontuação de 0.0 a 10.0.
 - A instituição quer manter uma listagem dos alunos que entregaram relatórios. Cada relatório entregue por um aluno corresponde a uma entrada na lista.
- Notas de 3.0 a 5.0 recebem desconto de 10% na mensalidade.
 - Notas de 5.1 a 7.0 recebem desconto de 30% na mensalidade.
 - Notas de 7.1 a 10.0 recebem desconto de 50% na mensalidade.

Exemplo

Problema: listagem de alunos

???



Aluno	Nota
Rafael	10
Ana	7
Paulo	5.5
Douglas	4.5
José	6
Mauro	8
Sérgio	9
Daniel	6.5
Maria	8.5
Pedro	3.5

Problema: como fazer um programa em Java que represente cada desconto da lista de alunos avaliada pelo professor?

Exemplo

Solução Homer Simpson !

- Usar VETORES!
- **Vetor Alunos** = guarda o nome de cada aluno
- **Vetor Notas** = guarda nota de cada aluno
- **Vetor Desconto** = guarda desconto para cada um conforme as notas



Exemplo



PUC GOIÁS

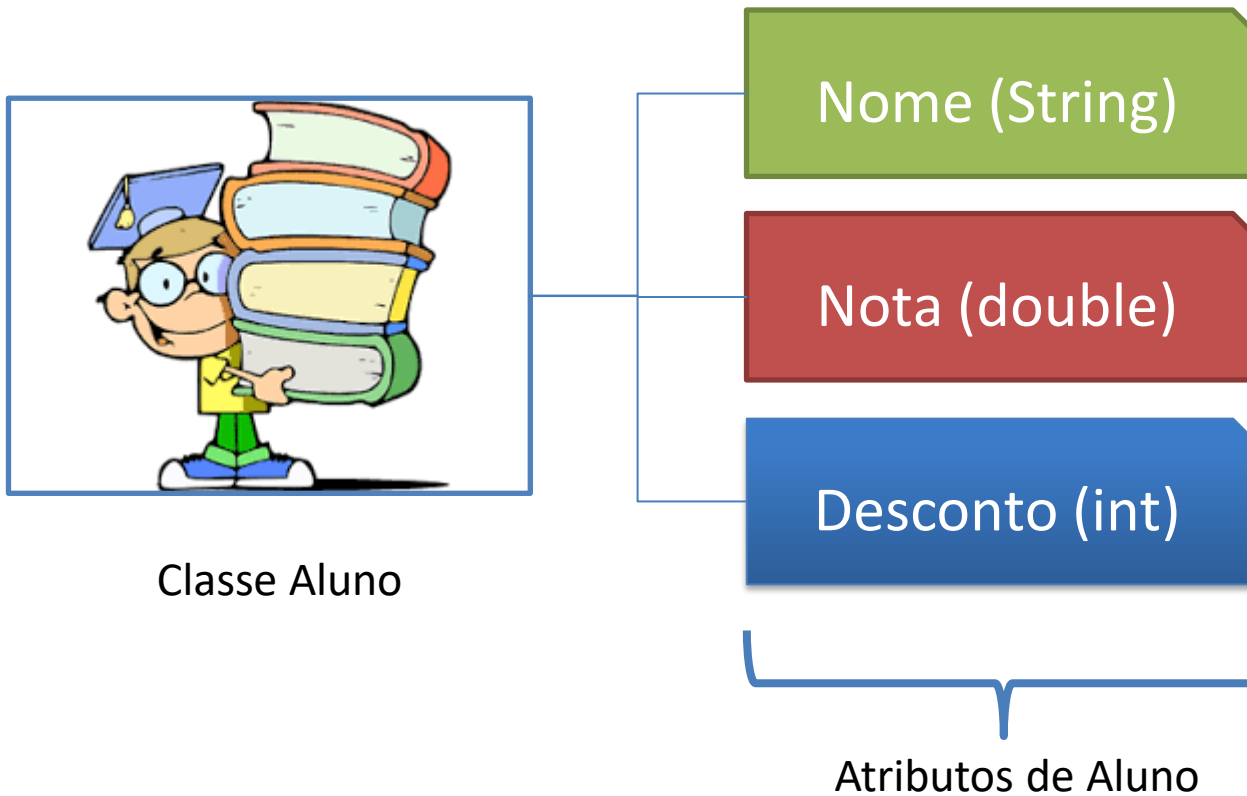
Solução Homer Simpson !

```
public class JavaApplication1 {  
  
    public static void main(String[] args) {  
        String Nome[] = {"Rafael", "Ana", "Paulo", "Douglas", "Jose", "Mauro",  
                        "Sergio", "Daniel", "Maria", "Pedro"};  
  
        double Nota[] = new double[] {10, 7, 5.5, 4.5, 6, 8, 9, 6.5, 8.5, 3.5};  
  
        int Desconto[] = new int[10];  
  
        int i;  
        for (i=0; i<10; i++)  
        {  
            if (Nota[i] >= 3 && Nota[i] <= 5)  
                Desconto[i] = 10;  
            else if (Nota[i] > 5 && Nota[i] <= 7)  
                Desconto[i] = 30;  
            else if (Nota[i] > 7 && Nota[i] <= 10)  
                Desconto[i] = 50;  
        }  
        System.out.println("Relação de Descontos");  
        System.out.println("Aluno\tNota\tDesconto");  
        for (i=0; i<10; i++) {  
            System.out.println(Nome[i] + "\t" + Nota[i] + "\t" + Desconto[i]);  
        }  
    }  
}
```



Exemplo

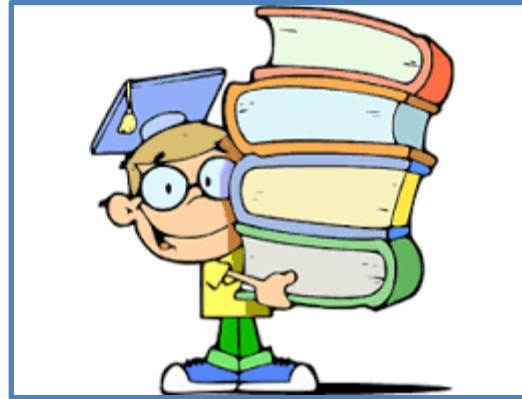
Solução através de Orientação a Objetos:



Exemplo

Solução através de Orientação a Objetos:

Classe Aluno e seus Métodos



Nome (String)

Gravar nome:
setNome (String name)

Ler nome:
String getNome ()

Nota (double)

Gravar nota:
setNota (double nota)

Ler nota:
double getNota ()

Desconto (int)

Gravar desconto:
setDesconto (int desconto)

Ler desconto:
int getDesconto ()

Exemplo

Solução através de Orientação a Objetos:

```
public class Aluno{  
    private String Nome;  
    private double Nota;  
    private int Desconto;  
  
    public void setNome(String nome) {  
        this.Nome=nome;  
    }  
    public void setNota(double valor){  
        this.Nota=valor;  
    }  
    public void setDesconto(int desconto){  
        this.Desconto=desconto;  
    }  
    public String getNome(){  
        return this.Nome;  
    }  
    public double getNota(){  
        return this.Nota;  
    }  
    public int getDesconto(){  
        return this.Desconto;  
    }  
}
```

Atributos

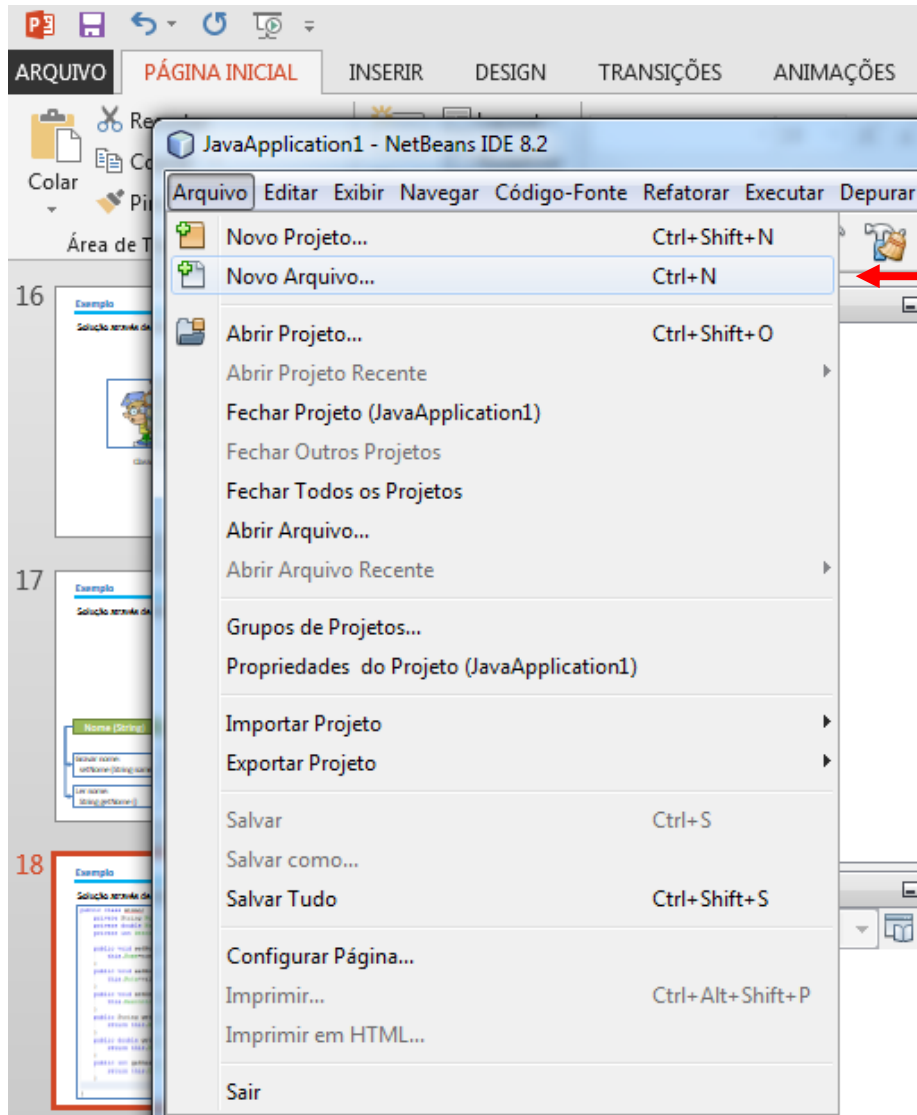
Métodos



Classe Aluno

Exemplo

Solução através de Orientação a Objetos:

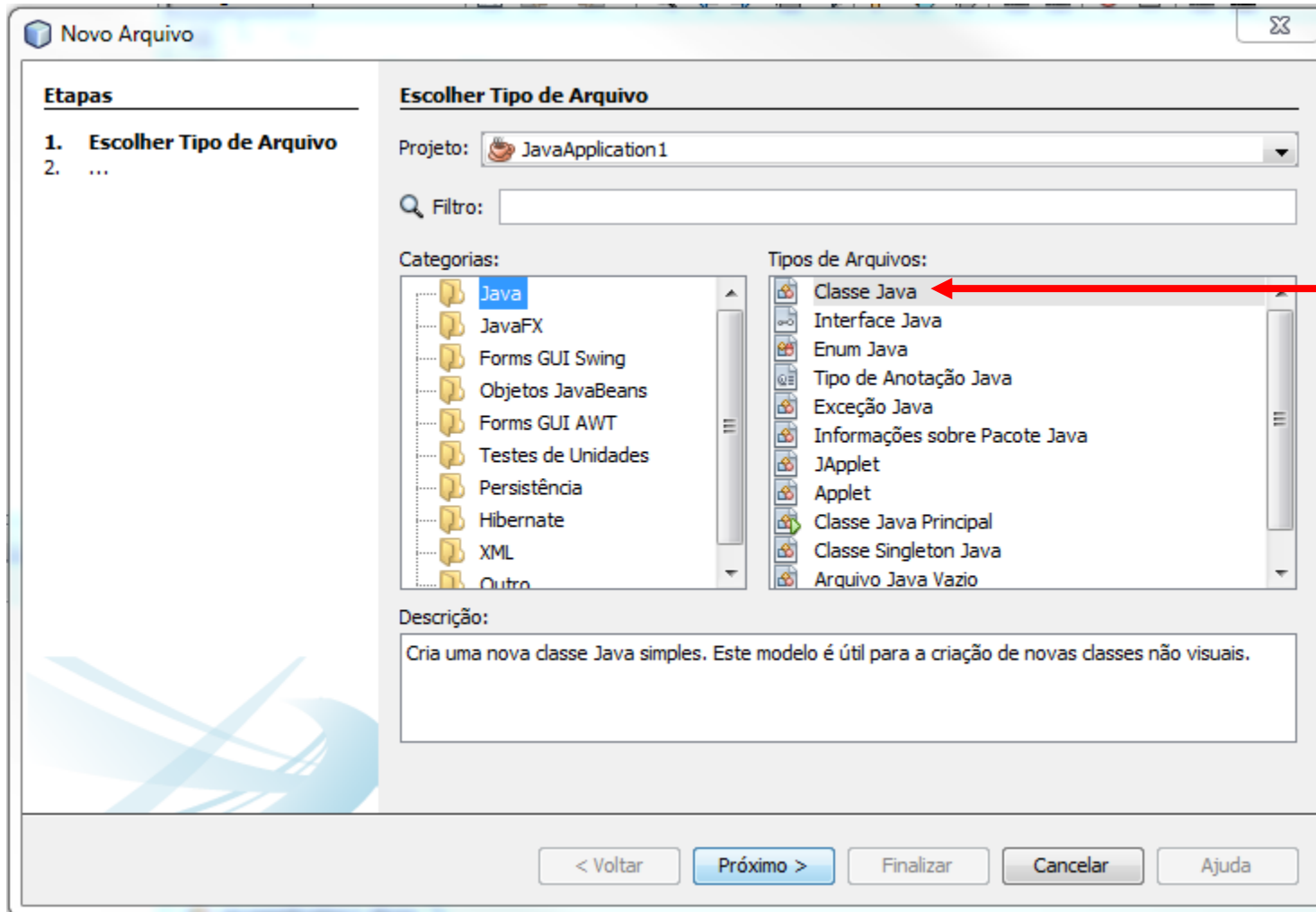


Clicar em Arquivo

↳ Novo Arquivo

Exemplo

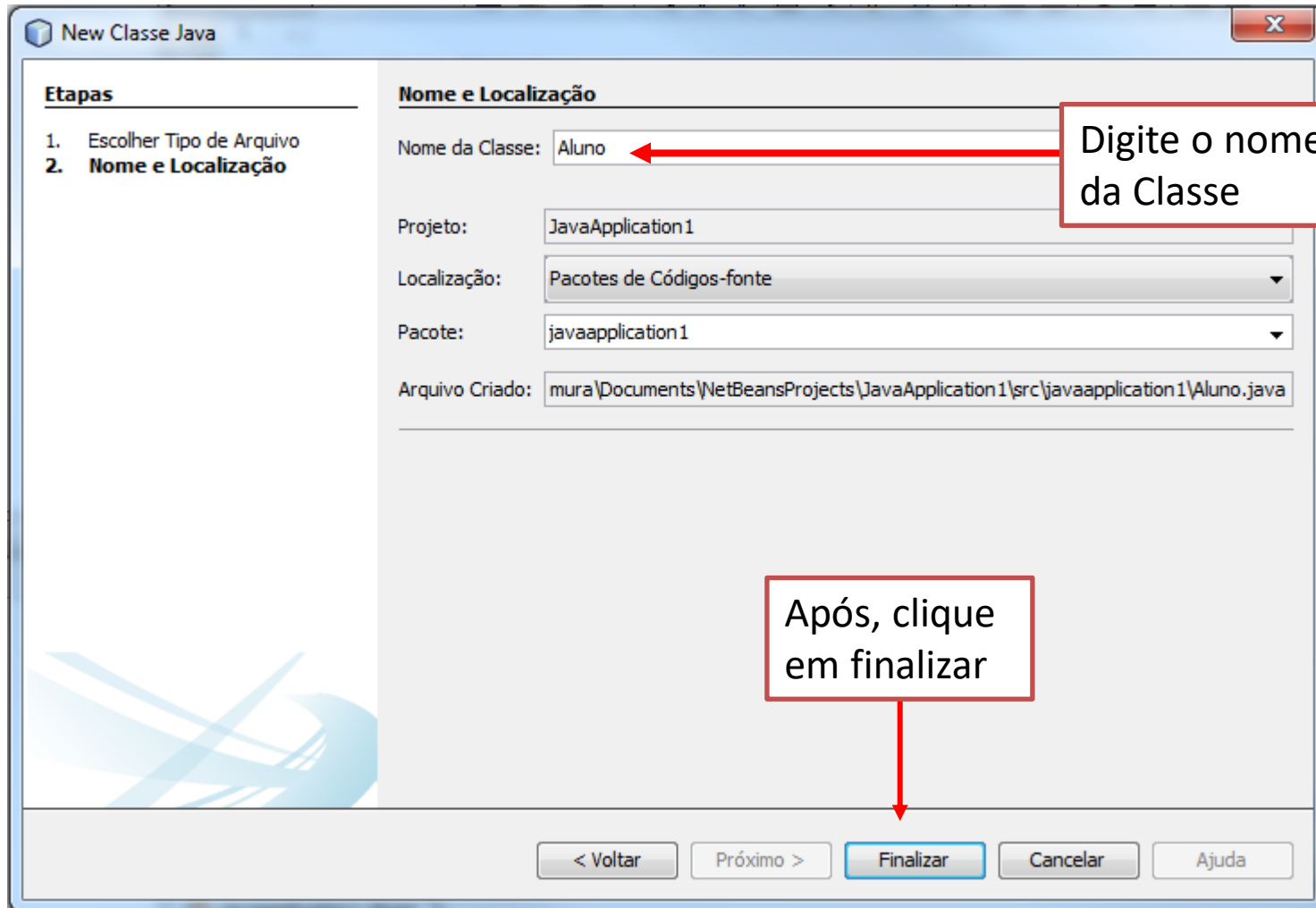
Solução através de Orientação a Objetos:



Clicar em
Classe Java

Exemplo

Solução através de Orientação a Objetos:



Etapas

1. Escolher Tipo de Arquivo
2. **Nome e Localização**

Nome e Localização

Nome da Classe:

Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > **Finalizar** Cancelar Ajuda

Digite o nome da Classe

Após, clique em finalizar

Exemplo

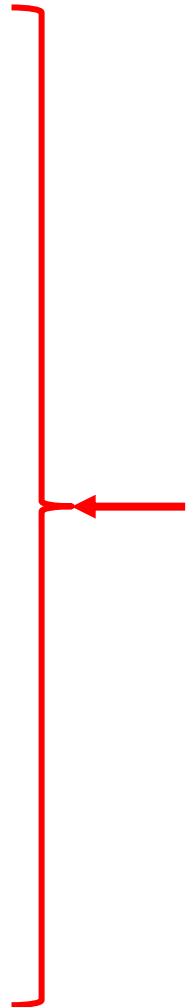
Solução através de Orientação a Objetos:



The screenshot displays an IDE environment with the following components:

- Project Explorer (Left):** Shows a project named 'JavaApplication1' with a package 'javaapplication1' containing the file 'Aluno.java', which is highlighted with a red arrow.
- Code Editor (Center):** Displays the source code for 'Aluno.java'. The code defines a class with three private attributes: 'Nome' (String), 'Nota' (double), and 'Desconto' (int). It includes a constructor and three setter methods: 'setNome', 'setNota', and 'setDesconto'. The 'Desconto' attribute and its corresponding setter and getter methods are highlighted in yellow.
- Class Navigator (Bottom Left):** Shows the class 'Aluno' with its members. The 'setNome(String nome)' method is highlighted in blue.

```
13     private String Nome;
14     private double Nota;
15     private int Desconto;
16
17     Aluno () {
18         this.Nome="";
19         this.Nota=0;
20         this.Desconto=0;
21     }
22
23     public void setNome(String nome) {
24         this.Nome=nome;
25     }
26     public void setNota(double valor){
27         this.Nota=valor;
28     }
29     public void setDesconto(int desconto) {
30         this.Desconto=desconto;
31     }
32     public String getNome () {
33         return this.Nome;
34     }
35     public double getNota () {
36         return this.Nota;
37     }
38     public int getDesconto () {
39         return this.Desconto;
40     }
41 }
```



Exemplo



Solução através de Orientação a Objetos:

Construtor da Classe Aluno

Também conhecidos pelo inglês **constructors**, os construtores são os responsáveis por criar o objeto em memória, ou seja, instanciar a classe que foi definida. Eles são obrigatórios e são declarados conforme declaração:

```
public class Carro{  
  
    /* CONSTRUTOR DA CLASSE Carro */  
    public Carro(){  
        //Faça o que desejar na construção do objeto  
    }  
  
}
```



```
Aluno () {  
    this.Nome="";  
    this.Nota=0;  
    this.Desconto=0;  
}
```

Exemplo



Solução através de Orientação a Objetos: Usando a classe Aluno para um aluno

```
public class JavaApplication1 {  
  
    public static void main(String[] args) {  
        Aluno aluno1=new Aluno();  
  
        aluno1.setNome("Rafael");  
        aluno1.setNota(10);  
        if (aluno1.getNota()>=3.0 && aluno1.getNota()<=5.0)  
            aluno1.setDesconto(10);  
        else if (aluno1.getNota()>5.0 && aluno1.getNota()<=7.0)  
            aluno1.setDesconto(30);  
        else if (aluno1.getNota()>7.0 && aluno1.getNota()<=10.0)  
            aluno1.setDesconto(50);  
  
        System.out.println("Nome: " + aluno1.getNome());  
        System.out.println("Nota: " + aluno1.getNota());  
        System.out.println("Desconto: " + aluno1.getDesconto());  
  
    }  
  
}
```


Exemplo

Solução através de Orientação a Objetos: Lista de Alunos

```
public class JavaApplication1 {  
  
    public static void main(String[] args) {  
        Scanner ler=new Scanner(System.in);  
        Aluno[] alunos = new Aluno[10];  
        int i;  
        for (i=0;i<10;i++){  
            alunos[i]=new Aluno();  
        }  
        for(i=0;i<10;i++){  
            System.out.print("Digite o nome do aluno " + (i+1) + ": ");  
            alunos[i].setNome(ler.next());  
            System.out.print("Digite a nota do aluno " + (i+1) + ": ");  
            alunos[i].setNota(ler.nextDouble());  
        }  
  
        for(i=0;i<10;i++){  
            if (alunos[i].getNota()>=3.0 && alunos[i].getNota()<=5.0)  
                alunos[i].setDesconto(10);  
            else if (alunos[i].getNota()>5.0 && alunos[i].getNota()<=7.0)  
                alunos[i].setDesconto(30);  
            else if (alunos[i].getNota()>7.0 && alunos[i].getNota()<=10.0)  
                alunos[i].setDesconto(50);  
        }  
  
        for (i=0;i<10;i++){  
            System.out.println("Nome: " + alunos[i].getNome()+"\t"+"Nota: " +  
                alunos[i].getNota()+"\t"+"Desconto: " + alunos[i].getDesconto());  
        }  
    }  
}
```

Declaração de um
Vetor de Aluno

Inicialização dos
objetos

Exercícios:

Exercício 01 - Escreva uma aplicação que demonstre o uso de instâncias da classe **ContaBancaria** que deve ser criada com o atributo: saldo. Demonstre como a transferência de valores de uma instância da classe para outra pode ser feita através de chamadas aos métodos **deposita** e **retira**. Crie um método **getSaldo** para retornar o saldo corrente da conta. Tente fazer com que os dados que serão usados nas classes sejam lidos do teclado.
Use a classe com dois vetores para indicar duas agências bancárias diferentes.





Referência Bibliográfica Principal

- DEVMEDIA. Disponível em <https://www.devmedia.com.br> . Acessado em Fevereiro de 2018.