



PROGRAMAÇÃO B - JAVA



LIANDERSON FRANCO BRUM




EMAIL: PROF. LIANDERSON@GMAIL.COM





FORMAÇÃO

- ESCOLA TÉCNICA SANTO INÁCIO - 2000
(Processamento de Dados)
 - FACULDADE DOM BOSCO - 2010
(Sistema de Informação)
- 



PROFISSIONAL

- ADVANCEDIT - Analista Projetista de Teste
- 

PROFISSIONAL

- ESCOLA TÉCNICA ALCIDES MAYA

Disciplinas :

Banco de Dados (Mysql)


ASP.NET

PHP

Especialização - Banco de Dados




Orientação a Objetos

- Classe
 - Objetos
 - Encapsulamento
- 




Orientação a Objetos

Por que “Programação Orientada a Objetos”
e não a tradicional Programação Estruturada?






Orientação a Objetos

- Mas por que “Programação Orientada a Objetos” ao invés da tradicional Programação Estruturada?
 - Compreender que em algum momento o código Orientado a Objetos utiliza-se do paradigma Estruturado, a grande diferença é a forma como a aplicação é idealizada.
- 




Orientação a Objetos

- Um paradigma de programação, seja ele Estruturado ou Orientado a Objetos é a forma como a solução para um determinado problema é desenvolvida.
- 



Orientação a Objetos


- Em Orientação a Objetos, os problemas são resolvidos pensando-se em interações entre diferentes objetos.
 - Em Paradigma Estruturado, procura-se resolver os problemas decompondo-os em funções e dados que somados formarão um programa.
- 



Orientação a Objetos

- Por que surgiu a Programação Orientada?

Porque os programas foram tornando-se mais complexos, surgiu assim a necessidade de resolver os problemas de uma maneira diferente.






Orientação a Objetos

- Quais os benefícios da abordagem orientada a objetos?
- 




Orientação a Objetos

- **Modularidade:** Uma vez criado um objeto pode ser passado por todo o sistema.
 - **Encapsulamento:** Detalhes de implementação ficam ocultos externamente ao objeto.
 - **Reuso:** Uma vez criado um objeto pode ser utilizado em outros programas.
 - **Manutenibilidade:** Manutenção é realizada em pontos específicos do seu programa (objetos).
- 



Classes e Objetos

- Objetos são “coisas” que temos no mundo real e abstraímos no mundo virtual para que possamos manipulá-los na resolução de problemas. Um objeto no mundo real sempre possui estado e comportamento, isto é, ele possui características e ações que são pertinentes a sua natureza.
- 


Classes e Objetos

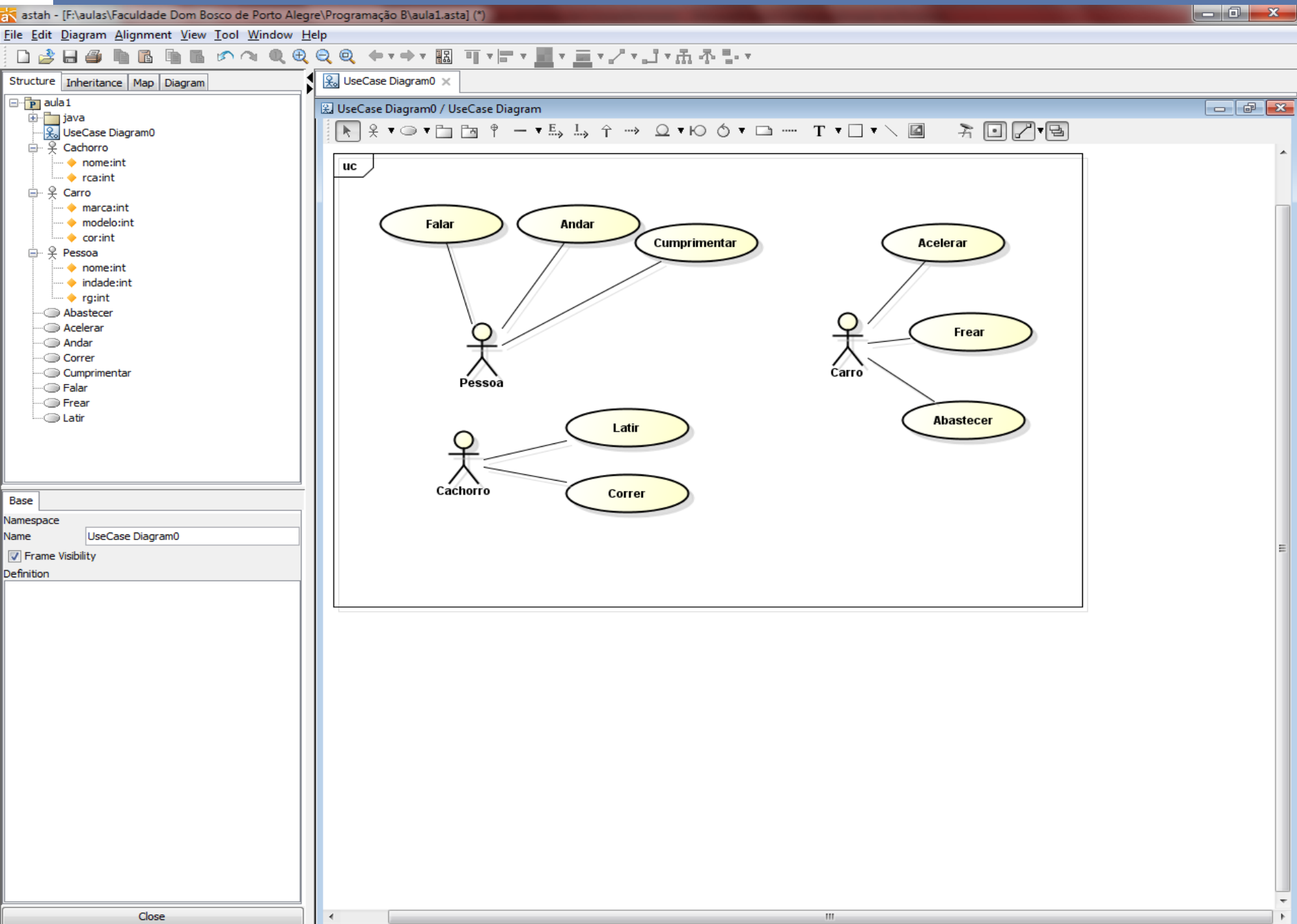
Objeto	Estado	Comportamento
Pessoa	Nome, idade, RG	Falar, andar, cumprimentar
Cachorro	Nome, raça	Latir, correr
Conta bancária	Saldo, agência, número	Creditar, debitar
Carro	Cor, marca, modelo	Acelerar, frear, abastecer



Classes e Objetos

O Estado e Comportamento,
respectivamente, são transformados em
dados e procedimentos quando
programamos de forma estruturada e
atributos e métodos quando utilizamos
orientação a objetos.





Classes e Objetos

- Objetos são instâncias das classes.
- Uma **classe** é uma especificação para um determinado tipo de objeto, isto é, para que o objeto seja de determinada classe ele, obrigatoriamente, terá que respeitar a especificação.

Por exemplo, vamos especificar que todo documento deve possuir, ao menos, foto, código, nome e data de nascimento.

Classes e Objetos

Documento	Documento 1	Documento 2
Foto:	Img1.png	Img4.png
Código:	123456	789012
Nome:	Alfredo	Juliana
Data de nascimento:	20/05/1990	30/09/1987

Classes e Objetos

- Vamos implementar:


```
public class documento
{
    //Estado
    String foto; //Nome do arquivo de imagem
    String nome; //Nome da pessoa
    Integer codigo; //Codigo deste documento
    String dataNascimento; //Data de nascimento
}
```

Métodos

- As ações que são realizadas sobre os atributos são chamados de **métodos** e o conjunto de métodos define o **comportamento** de um objeto.
- Imagine que agora o sistema a ser modelado é o de uma corrida de carros. O principal objeto de uma corrida de carros são os próprios carros para os quais nós devemos criar uma especificação (classe) para ser utilizada neste sistema imaginário.



Métodos

- O nosso carro de corrida terá seu estado definido pelo conjunto de atributos número de identificação, velocidade atual e velocidade máxima. O comportamento será definido pelo conjunto de métodos acelerar, frear, ligar e desligar.
 - Vamos criar a nossa classe apenas com a definição do estado, inicialmente:
- 

Métodos

```
public class CarroCorrida {  
    Integer numeroIdentificacao;  
    Double velocidadeAtual;  
    Double velocidadeMaxima;  
    void ligar()  
    {  
        System.out.println("VRUUUMmmmmmmmmmm");  
    }  
    void desligar()  
    {  
        System.out.println("MMMnnnnnnn");  
    }  
}
```


Métodos

Observamos que os dois métodos não retornam nenhuma informação para quem os aciona (invoca), isto é informado através da palavra reservada **void**.

```
void acelerar()
{
    //velocidadeAtual = 10;
    velocidadeAtual = 10.0;
    System.out.println(velocidadeAtual+" KM/h");

}
```

Métodos

Vamos Implementar o metodo Frear

```
void frear(Integer intensidadeFreada)
```

```
{
```

```
    if(intensidadeFreada > 100)
```

```
{ intensidadeFreada = 100; }
```

```
    else if(intensidadeFreada < 0)
```

```
{ intensidadeFreada = 0; }
```

```
    velocidadeAtual -= intensidadeFreada*0.25;
```

```
    if(velocidadeAtual < 0)
```

```
{ velocidadeAtual = 0.0; }
```

```
}
```

Métodos

O método frear recebe um parâmetro que significa a intensidade com que o pedal de freio foi acionado. Esta intensidade pode ser um valor entre 0 e 100, a velocidade após a freada é o resultado da intensidade da freada multiplicada pelo fator 0.25, tudo isso diminuído da velocidade atual, caso o resultado desta operação seja menor do que 0 (zero), então o valor final será zero.

Métodos

- Agora vamos adicionar um Piloto
- Na classe CarroCorrida adicionar

class Piloto

{

String nome;

Integer habilidade;

Integer idade;

}