

Faculdade Senac Maringá

Programação Orientada a Objetos

Prof. Rafael Florindo



Aula 01
Bimestre 01

Cronograma da aula de hoje

❖ Plano de Ensino

- Ementa
- Sistema de Avaliação
- Cronograma
- Ambiente de Produção

❖ Aula 01

- ❖ Introdução ao paradigma orientado a objetos;
- ❖ Conceitos da Orientação a Objetos
 - Classes,
 - Objetos,
 - Atributos e
 - Métodos).

Ementa

- ❖ Introdução e motivação do paradigma orientado a objetos. Introdução à linguagem de programação Java.
- ❖ Conceito de Classes e Objetos.
- ❖ Ciclo de Vida dos Objetos.
- ❖ Construtores e Destrutores.
- ❖ Atributos e Métodos.
- ❖ Encapsulamento.
- ❖ Herança Simples e Múltipla.
- ❖ Classes e Métodos Abstratos.
- ❖ Polimorfismo.
- ❖ Interface.
- ❖ Desenvolvimento de aplicações desktop usando a linguagem Java.
- ❖ Criação de arquivos com extensão “.jar”.
- ❖ Execução de aplicações Java utilizando linha de comando.

Sistema de Avaliação

1º BIMESTRE

- ❖ Avaliação bimestral 4,0 (podendo ser, teórica ou prática)
- ❖ Atividades em sala e trabalhos 4,0
- ❖ Prova integradora 2,0
- ❖ **Total do bimestre 10,0**

Média Bimestral 7,0

2º BIMESTRE

- ❖ Avaliação bimestral 4,0 (podendo ser, teórica ou prática)
- ❖ Atividades em sala e trabalhos 4,0
- ❖ Prova integradora 2,0
- ❖ **Total do bimestre 10,0**

Média Bimestral 7,0

Conteúdo Programático

Aula 1 - Introdução ao paradigma orientado a objetos e Conceitos da Orientação a Objetos (Conceitos de Classes, Objetos, Atributos e Métodos).

Aula 2 - Introdução à linguagem de programação desktop Java com aplicação na Orientação a Objetos.

Aula 3 - Pilares da Orientação a Objetos: Abstração, Encapsulamento

Aula 4 - Aplicação na prática sobre os Pilares da Orientação a Objetos: Abstração, Encapsulamento

Aula 5 - ATIVIDADE PRÁTICA (4 PONTOS)

Aula 6 - Pilares da Orientação a Objetos: Herança e Polimorfismo (Reutilização de Código, Generalização e Especialização e, Sobrescrita e Sobrecarga de Métodos).

Aula 7 - Aplicação na prática sobre os Pilares da Orientação a Objetos: Herança e Polimorfismo (Reutilização de Código, Generalização e Especialização e, Sobrescrita e Sobrecarga de Métodos).

Aula 8 - Construtores, Destrutores e Ciclo de Vida dos Objetos com aplicação na prática

Aula 9 - AVALIAÇÃO DA DISCIPLINA (4 PONTOS)

Conteúdo Programático

Aula 10 - Tratamento de Exceções.

Aula 11 - Classes Abstratas e Interfaces (Classes Abstratas, Métodos Abstratos, Padronização e Contratos)

Aula 12 - Collections (Listas, Conjuntos e Coleções, e, Laço Foreach)

Aula 13 - Testes de unidade e documentação de código (Documentação com JavaDOC)

Aula 14 – Trabalhando com projetos (SEMANA ACADEMICA)

Aula 15 - **ATIVIDADE PRÁTICA (4 PONTOS)**

Aula 16 - Relacionamentos (Dependência, Agregação e Composição)

Aula 17 - Desenvolvimento de aplicações de linha de comando com Java

Aula 18 - Compilação e execução de arquivos .jar utilizando linha de comando

Aula 19 - **AVALIAÇÃO DA DISCIPLINA (4 PONTOS)**

Aula 20 - Padrões de Projetos

Ambiente de Produção

Algumas IDEs que podem ser utilizada

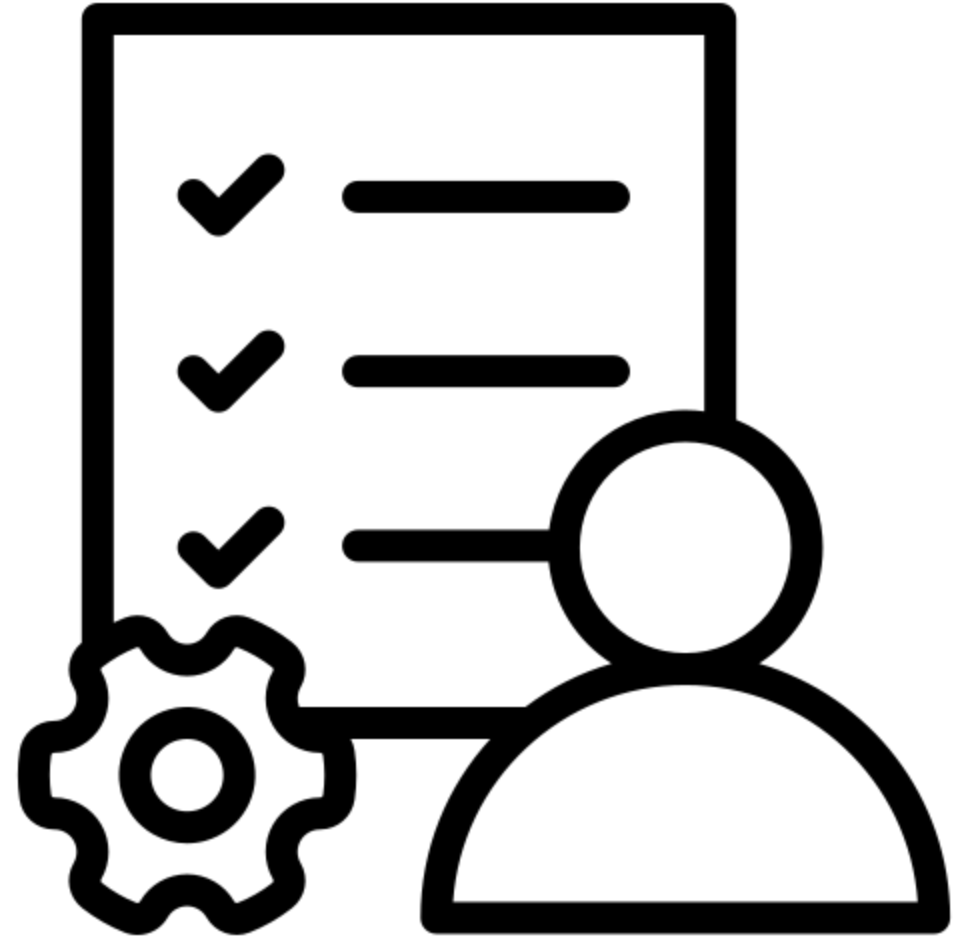
- ❖ IntelliJ IDEA Versão Community (<https://www.jetbrains.com/pt-br/idea/download/?section=windows>)
- ❖ Eclipse (<https://www.eclipse.org/downloads/>)
- ❖ Vscode (<https://code.visualstudio.com/download>)

Referências Bibliográficas

- ❖ DEITEL, Harvey M. **Java: Como Programar**. 6 ed. Editora Pearson. Prentice-Hall, 2005.
- ❖ SIERRA, Kathy; BATES, Bert. **Use a cabeça Java**. 2 ed. Editora Alta Books, 2007.
- ❖ BARNES, David J.; KÖLLING, Michael. **Programação orientada a objetos com Java**. 4 ed. Editora Pearson/Prentice-Hall, 2009.
- ❖ SANTOS, Rafael. **Introdução à Programação Orientada a Objetos Usando Java**. Editora Campus, 2003.
- ❖ HORSTMANN, CAY S. **Conceitos de Computação com Java**. Editora Bookman. 2009.
- ❖ ARNOLD Ken. **A linguagem de Programação JAVA**. Editora Bookman. 2008.
- ❖ ANSELMO, Fernando. **Aplicando Lógica Orientada a Objeto em Java**. Editora Visual Books. 2005.
- ❖ RODRIGUES FILHO, Renato. **Desenvolva Aplicativos com Java 2**. Editora Érica LTDA. 2007.

Aula 1

- ❖ Introdução ao paradigma orientado a objetos;
- ❖ Conceitos da Orientação a Objetos
 - Classes,
 - Objetos,
 - Atributos e
 - Métodos).



Introdução ao paradigma orientado a objetos

- ❖ Orientação a objetos é um paradigma aplicado na programação que consiste na interação entre diversas unidades chamadas de objetos.
- ❖ Usamos a orientação a objetos para nos basear na vida real e resolver problemas de software, ou pelo menos tentamos. Ela acaba sendo uma base inclusive para outros paradigmas.
- ❖ Orientação a objetos é algo atemporal e não está ligado a uma linguagem.

Introdução ao paradigma orientado a objetos

- ❖ Na orientação a objetos, devemos trabalhar constantemente com um **dos conceitos pilares da abstração** ou abstrair, **significa esconder os detalhes de uma implementação, ou seja, quanto menos souberem sobre nossas classes, mais fácil de consumi-las será.**

Paradigma extruturado X Paradigma orientado a objetos

Programação Estruturada

❖ Composição dos Programas

- Um programa é composto por um conjunto de rotinas
- A funcionalidade do programa é separada em rotinas
- Os dados do programa são variáveis locais ou globais

❖ Fluxo de Execução

- O programa tem início em uma rotina principal
- A rotina principal chama outras rotinas
- Estas rotinas podem chamar outras rotinas, sucessivamente
- Ao fim de uma rotina, o programa retorna para a chamadora

Paradigma extruturado X Paradigma orientado a objetos

Programação Orientada a Objetos

❖ Composição do programa:

- A funcionalidade do programa é agrupada em objetos
- Os dados do programa são agrupados em objetos
- Os objetos agrupam dados e funções correlacionados

❖ Fluxo de Execução:

- Similar ao anterior
- Os objetos colaboram entre si para a solução dos objetivos
- A colaboração se realiza através de chamadas de rotinas

Paradigma extruturado X Paradigma orientado a objetos

Paradigma estruturada

❖ Vantagens

- Controle mais eficaz quanto ao fluxo de execução do programa
- Facilidade de compreender o código quando o mesmo é analisado

❖ Desvantagens

- Código confuso, caso o desenvolvedor não faça a modularização do código
- Dificuldade em realizar o reuso

Obs: Adequada para construir programas para realização de operações lógicas e aritméticas.

Paradigma extruturado X Paradigma orientado a objetos

Programação OO


❖ Vantagens

- Reutilização de código
- Melhor organização do código no programa
- Mais próxima do Ser Humano

❖ Desvantagens

- Desempenho do código menor que linguagens estruturadas
- Confusão do código caso os conhecimentos de OO não sejam corretamente aplicados

Obs: Adequada para construir Sistemas de Informação.

A diagram on the left side of the slide. It consists of a large dark green semi-circle on the left. To its right is a smaller, light green semi-circle that is partially overlapping the dark green one. The entire diagram is enclosed within a thin green rectangular border.

OO

- Encapsulamento
- Herança
- Polimorfismo

Estruturada

- Seqüencia
- Decisão
- Repetição

Linguagens do Paradigma orientado a objetos



Paradigma orientado a objetos

O ser humano se relaciona com o mundo através do conceito de **objetos**, onde um objeto é uma entidade que combina estrutura de dados e comportamento funcional.

Estamos sempre **identificando** objetos ao nosso redor.

Para isso:

atribuímos nomes
classificamos em grupos – classes.

Conceitos da orientado a objetos: Classes

As Classes descrevem um **conjunto de objetos com as mesmas propriedades** (atributos e associações) e **o mesmo comportamento** (operações). Em outros termos, uma classe descreve os serviços providos por seus objetos e quais informações eles podem armazenar.

A classe é a forma mais básica de se definir apenas uma única vez **como devem ser todos os objetos criados a partir dela**, em vez de definir cada objeto separadamente e até repetidamente.

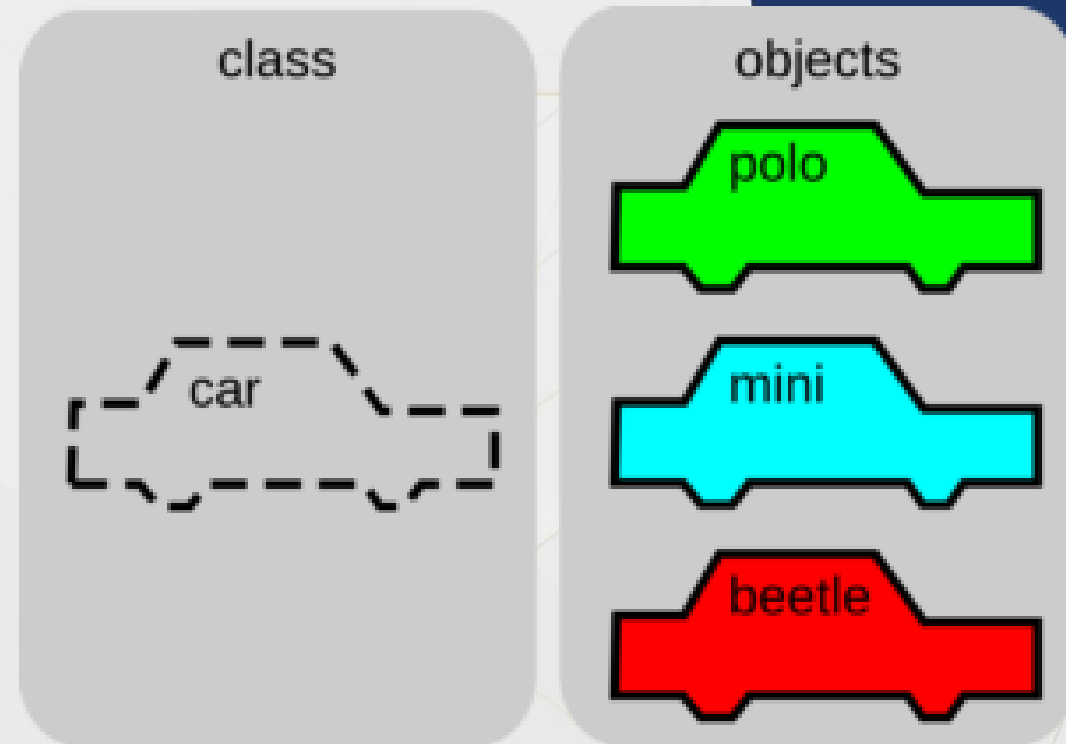


Conceitos da orientado a objetos: Classes

A classe é a forma mais básica de se definir apenas uma única vez como devem ser todos os objetos criados a partir dela, em vez de definir cada objeto separadamente e até repetidamente.

Uma classe também pode ser definida como uma abstração de uma entidade, seja ela física (carro, avião, pessoa etc.) ou conceitual como (viagem, venda, estoque etc.) do mundo real.

Fonte: (CARVALHO, 2022).



1º Passo na orientação a objetos

Identificar as características e o comportamento de objetos do mundo real é o **primeiro passo da programação OO**.

Observe um objeto e pergunte:

1. Quais as **possíveis características** deste objeto e quais estados elas assumem?
2. Quais **comportamentos (ações)** que ele pode executar?

Conceitos da orientado a objetos: Classes



Carro



Pessoa



Turma

Conceitos da orientado a objetos: Classes

Por exemplo:

Imagine que precisamos desenvolver um site de vendas online. Assim, aparecerão entidades como Vendas, Cliente, Fornecedor, Produto entre outras. Vemos que todos estes substantivos fazem parte do contexto de um site de vendas como este. Logo, é possível especificar classes para assim manipular tais entidades.

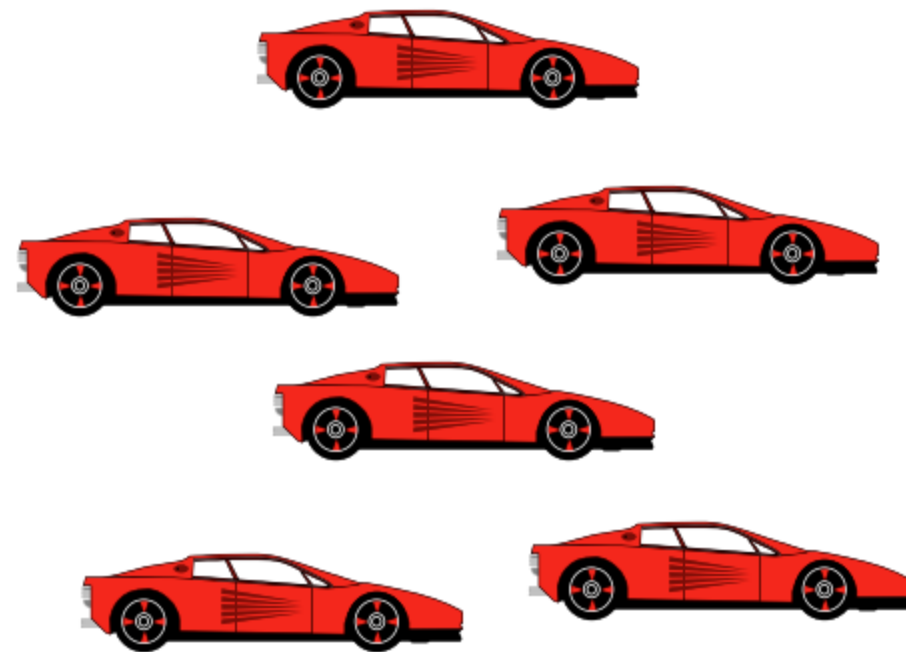
Conceitos da orientado a objetos: Classes

Quando se cria uma classe, estamos definindo o que chamamos de tipo abstrato de dado. **Esse termo é bem parecido com o Struct da programação estruturada**, onde é possível apenas definir os dados que a entidade virá a ter. Já em uma classe, também podemos definir as operações juntas a estes dados, ou seja, é possível observa que a OO tem como premissa representar melhor o mundo real.

Conceitos da orientado a objetos: Classes

```
class Carro{  
    //atributos  
    //métodos  
}
```

Carro
Número de Rodas Cor Cor Lateral
Anda Para Acelera Estaciona



Aluno

Nome

Matrícula

Nota Média

João

193.31.098-7

7,6

Maria

195.31.022-5

8,7

As classes proveem a estrutura para a construção de objetos - estes são ditos instâncias das classes.



As classes proveem a estrutura para a construção de objetos - estes são ditos instâncias das classes.



Conceitos da orientado a objetos: Classes

Pessoa
- Nome
- Telefone
- Endereço
- Email
- Sexo
- Data de nascimento
+ Andar
+ Falar
+ Dormir
+ Pensar

propriedades

métodos

Animal
- Tipo
- Raça
- Idade
- Sexo
+ Andar
+ Correr
+ Morder
+ Comer

propriedades

métodos

Conceitos da orientado a objetos: Atributo

O Atributo é o elemento de uma classe responsável por definir sua estrutura de dados. O conjunto destes será responsável por representar suas características e fará parte dos objetos criados a partir da classe.

Exemplo de atributos:

nome, dataNascimento (e não idade), genero, telefone, cpf, cnpj entre outros.

Nome que deve ser evitados: qtd, data, vlr, ou seja, escreva o substantivo por completo: quantidade, dataCadastro, valor.

Conceitos da orientado a objetos: Atributo

```
class Paciente{  
    String Nome;  
    Float altura;  
    Date dataCadastro;  
    Boolean possuiPlanoSaude;  
}
```

Conceitos da orientado a objetos: método

O **método** é uma **porção do código** (sub-rotina) que é disponibilizada pela classe. Esta é **executada quando é feita uma requisição** a ela. Um método serve para identificar quais serviços, ações que a classe oferece. Eles são responsáveis por definir **e realizar um determinado comportamento**.

Para definir o nome do método, ou seja, a assinatura do método, deve-se pensar em verbos que denotam uma ação, **tais como, cadastrar, alterar, validar entre outros**. Mas **cuidado, novamente, cadastrar o que?** Voltamos ao exemplo do Hospital, poderíamos definir a assinatura de um método como **cadastrarPaciente**.

Vale salientar que o ideal é ter apenas uma única funcionalidade dentro do método.

Conceitos da orientado a objetos: método

Os métodos devem possuir um indicador de retorno ou não, ou seja, ele pode ser considerado um procedimento quando não retorna valor, ou uma função que deve retornar, isso na programação estruturada. Na programação OO todo método deve ter sua assinatura completa, ou seja, utiliza void para indicar que aquele método não retorna valor, ou os tipos dos dados primitivos, string ou reference, quando deve ser retornar valores.

Void cadastrarPaciente(){ //...}

Boolean cadastrarPaciente(){ //...}

String cadastrarPaciente(){ //...}

UTI cadastrarPaciente(){ //...}

Conceitos da orientado a objetos: método

```
class Paciente{  
    String Nome;  
    Float altura;  
    Date dataCadastro;  
    Boolean possuiPlanoSaude;  
  
    Void CadastrarPaciente(){  
        //instruções do método  
    }  
}
```

Conceitos da orientado a objetos: método

O **método pode receber parâmetros**, ou seja, valores. Contudo, ao declarar uma assinatura de um método e colocar parâmetros, ao invocar o método, deve ser passado a quantidade de parâmetros bem como o tipo correto dos dados.

Exemplo de um método:

```
Void    cadastrarPaciente(String    nome,    Float    altura,    Date  
dataCadastro,    Boolean possuiPlanoSaude)  
{  
    //  
}
```

Conceitos da orientado a objetos: método

```
class Paciente{  
    String Nome;  
    Float altura;  
    Date dataCadastro  
    Boolean possuiPlanoSaude  
  
    Void cadastrarPaciente(String nome, Float altura,  
    Date dataCadastro, Boolean possuiPlanoSaude)  
    {  
        //  
    }  
}
```



Conceitos da orientado a objetos

Objetos: entidades que interagem entre si, onde cada uma delas desempenha um papel específico.

É uma **instância de um objeto** é uma unidade de programação que é armazenada em uma variável.



Carro

GOI
Vermelho
4 portas

Fusca
Preto
1300



Pessoa

Rafael
Maringá
Senac

Juliana
Sarandi
SEED



Turma

ADS
3º Sem.
POO

ADS
4º Sem.
framework

Colaboração entre objetos

- ❖ Um programa OO é um conjunto de objetos que colaboram entre si para a solução de um problema.
- ❖ Objetos colaboram através de trocas de mensagens.
- ❖ A troca de mensagens representa a chamada de um método

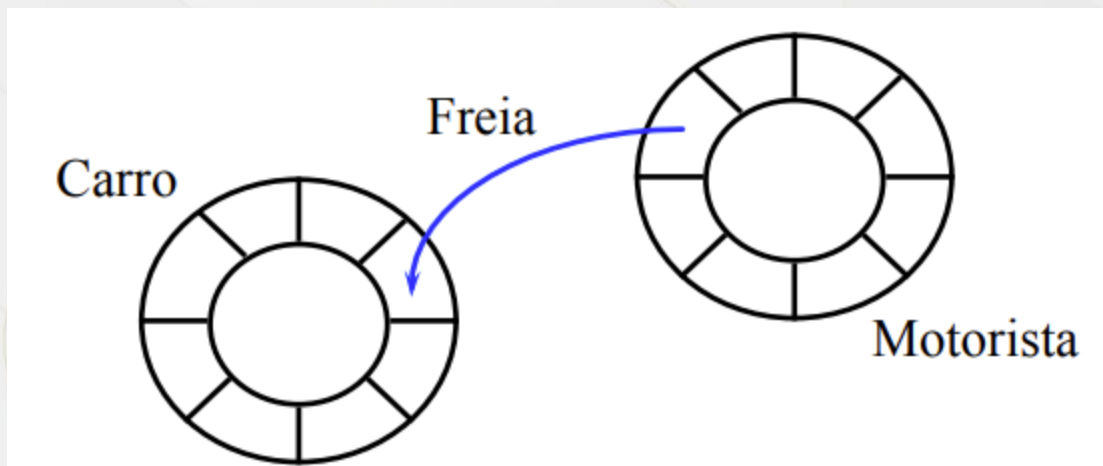


Diagrama de Classe

- ❖ Em programação, um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos.
- ❖ O DC é um conjunto de objetos com as mesmas características, assim saberemos identificar objetos e agrupá-los, de forma a encontrar suas respectivas classes.
- ❖ Na *Unified Modeling Language* (UML) em diagrama de classe, uma classe é representada por um retângulo com três divisões, são elas: O nome da classe, seus atributos e por fim os métodos

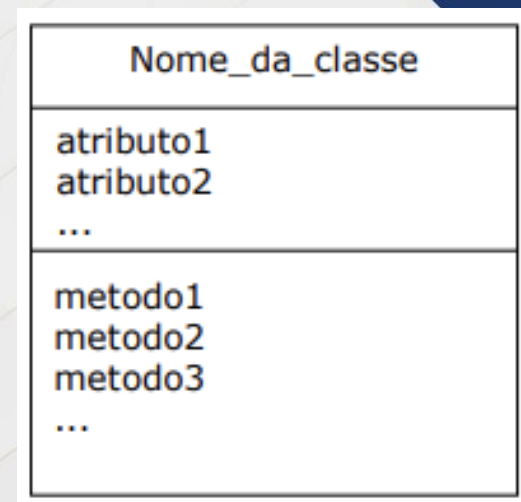
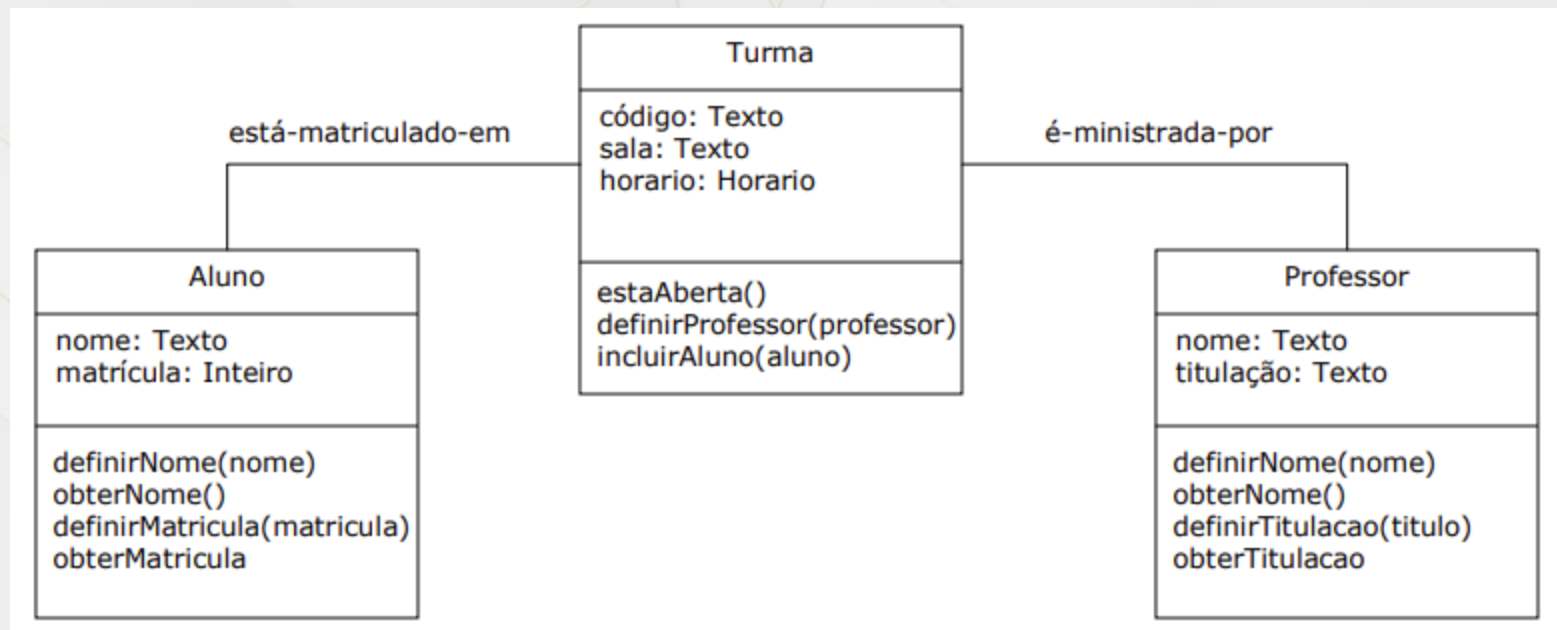


Diagrama de Classe

Por que é tão importante encontramos as classes para o desenvolvimento de um sistema?

- ❖ **Normalmente cada classe do diagrama representa uma tabela do banco de dados, por esse motivo é tão importante encontrarmos.**
- ❖ **Observe também que para identificarmos uma classe, precisamos antes identificar seus objetos com características semelhantes.**



Elementos de um Diagrama de Classe

- ❖ Classes

- ❖ Relacionamentos

- Associação
 - Agregação
 - Composição
- Generalização
- Dependência

Diagrama de Classe: Classe

- ❖ Atributos – Representam o conjunto de características (estado) dos objetos daquela classe
 - Visibilidade:
 - + público: visível em qualquer classe de qualquer pacote
 - # protegido: visível para classes do mesmo pacote
 - - privado: visível somente para classe

Exemplo:

+ nome : String

Diagrama de Classe: Classe

- ❖ Métodos – Representam o conjunto de operações (comportamento) que a classe fornece.
 - Visibilidade:
 - + público: visível em qualquer classe de qualquer pacote
 - # protegido: visível para classes do mesmo pacote
 - - privado: visível somente para classe

Exemplo:

- getNome() : String

Prática



❖ <https://online.visual-paradigm.com/drive/#diagramlist:proj=0&new=ClassDiagram>



❖ <https://app.creately.com/d/start/dashboard>

Exemplos

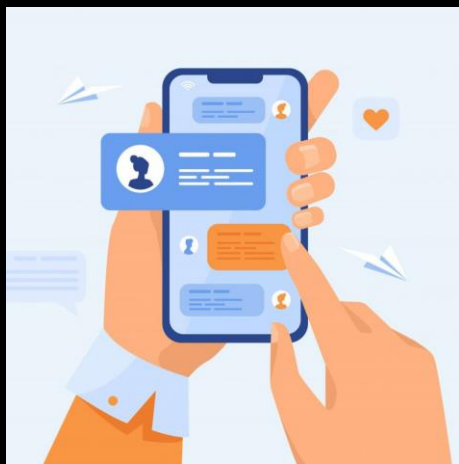
Diagrama de Classe



1 - Construir um Diagrama de Classe, onde tenha a classe Pessoa com os seguintes atributos: nome, idade, salario, peso e casado.

Exemplos

Diagrama de Classe

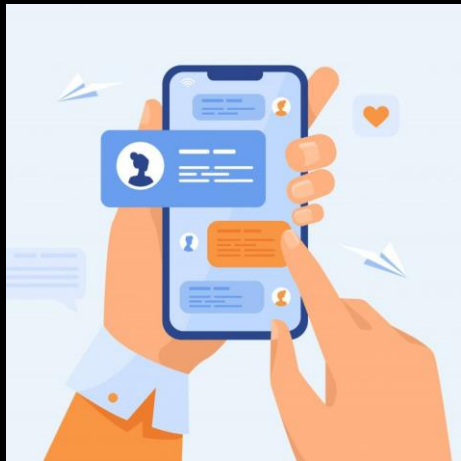


2 – Construir um Diagrama de Classe, onde tenha a classe Pessoa com os seguintes atributos: **código, nome, espécie e idade.**



Exemplos

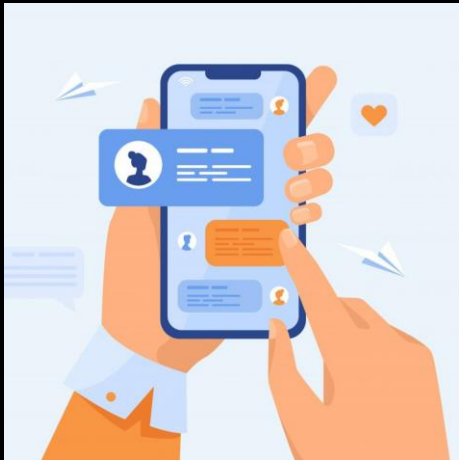
Diagrama de Classe



3 – Construir um Diagrama de Classe, onde tenha a classe Fornecedor. Atribua os atributos necessários.

Desafio

Diagrama de Classe



1 – Identificar as classes, e seus possíveis atributos e métodos.

A Faculdade SENAC deseja informatizar seu sistema de matrículas:

- A Faculdade oferece dois cursos.
- O Coordenador de um curso define as disciplinas que serão oferecidas pelo seu curso num dado semestre.
- Várias disciplinas são oferecidas em um curso.
- Várias turmas podem ser abertas para uma mesma disciplina, porém o número de estudantes inscritos deve ser entre 3 e 10.
- Estudantes selecionam 4 disciplinas.
- Quando um estudante matricula-se para um semestre, o Sistema de Registro Acadêmico (SRA) é notificado.
- Após a matrícula, os estudantes podem, por um certo prazo, utilizar o sistema para adicionar ou remover disciplinas.
- Professores usam o sistema para obter a lista de alunos matriculados em suas disciplinas. O Coordenador também.
- Todos os usuários do sistema devem ser validados.

Cronograma da próxima aula

- ❖ Retomada dos conceitos de POO
 - Classe
 - Atributos
 - Métodos
 - Objetos
- ❖ Introdução à linguagem de programação desktop Java com aplicação na Orientação a Objetos.

Faculdade Senac Maringá

Programação Orientada a Objetos

Prof. Rafael Florindo



Aula 02
Bimestre 01