**Exercício de Estruturas**

Exercício 1

Defina dois números inteiros. Atribua um valor a cada um. Verifique se um número é divisível pelo segundo e indique o resultado obtido por meio de uma mensagem.

**NOTA:** O fato de ser divisível significa que ao dividir precisa dar um valor exato, ou seja, sem decimais.

A linguagem Java

O núcleo do Java são as classes, mais adiante veremos o que são e como construí-las. Mas para começar a trabalhar, devemos começar a usar as próprias classes do Java.

**Variáveis**

Em Java, encontramos **tipos primitivos** como uma ferramenta de desenvolvimento e o chamamos de tipos de dados por apenas nos permitirem armazenar um valor.

Por exemplo, int, float, double e char. Quando definimos uma variável com esses tipos primitivos, só podemos armazenar valores.

**Classes**

Neste caso teremos um elemento que, além de armazenar um valor, nos permite realizar certas operações que já estão programadas, chamamos esses métodos de operações.

Por exemplo, String é uma classe, portanto, é inicializada em maiúsculas. Nomeamos todas as classes com a inicial maiúscula, se definirmos:

String nome;

Veremos que ao usar **nome** nos oferece os métodos disponíveis.

nome.

Estas são funções que já estão resolvidas e que só podemos utilizá-las com a classe a que pertence, ou seja, cada classe em Java possui seus próprios métodos.Texto

Descrição gerada automaticamente

Para entender como funcionam as próprias classes da linguagem, vamos nomear 3 classes que são úteis para nós - String já usamos em nossa primeira classe: **String, Integer, Float,** note que todas elas começam com a inicial maiúscula.

As classes **Integer** e **Float** são equivalentes aos tipos de dados primitivos, ou seja, nos permitem armazenar valores dos tipos indicados e também fornecem certas funcionalidades. Costuma-se dizer que eles envolvem os tipos primitivos.

Tenha em mente que quando usamos essas classes é que não podemos usar operadores como "==", para fazer uma comparação usamos o .equals (), por exemplo:

nome.equals ("Juan") retornará verdadeiro se a string "Juan" for salva no nome e falso caso contrário.

O **equals** é usado para comparar por igualdade, desde que estejamos trabalhando com classes.

Se quisermos comparar se um valor é maior ou menor que outro, devemos usar **.compareTo ()** .

Outra coisa a notar é que uma String à qual não atribuímos nada terá o valor **null**. Isso acontece com todas as classes e, se definirmos um elemento (objeto) de uma classe, inicialmente ele terá o valor null.

**Packages**

Para organizar as classes, existem os pacotes, onde as classes podem ser agrupadas. Posteriormente, iremos usá-los para nossas classes, mas por enquanto devemos saber que as classes Java também são agrupadas em pacotes, ou como seu nome em inglês: **package.**

**Prática Integradora**

**Objetivo**

O objetivo é poder praticar o uso da sintaxe da linguagem de programação Java.

**Exercício 1**

Digite seu nome e sobrenome separadamente, a partir dessas variáveis, você deve obter uma terceira com as suas iniciais

Funções

As funções em Java são semelhantes às  vistas em JavaScript. No entanto, há algumas coisas a se ter em mente, pois é uma linguagem digitada, portanto teremos que definir mais algumas coisas.

**Como definimos uma função?**

Para defini-la, vamos considerar três coisas:

* O que a função retorna.
* Que nome tem.
* Os parâmetros da qual precisaremos.

Quando dizemos "o que ela retorna", nos referimos ao tipo de dados que a função retorna. Portanto, a definição seria mais ou menos assim:

|  |
| --- |
| Tipo de retorno nome (parâmetros) |

**Nomes**

O nome deve ser o mais descritivo possível, não importa se precisamos juntar duas ou mais palavras; por exemplo, para nomes descritivos podem ser:

calcularTotal, inserirData, buscarNúmeroMaisRepetido, soma.

**Parâmetros**

Não há muitas diferenças em termos de parâmetros, apenas de que é necessário indicar o tipo de cada um; portanto, para que uma função receba valores, iremos indicá-la da seguinte forma:

|  |
| --- |
| (**int** num1, **int** num2) |

|  |
| --- |
| (**double** número, String descrição da string) |

|  |
| --- |
| (**int** quantidade, **int** posição, String nome da string) |

**Tipos de retorno**

Até agora, as diferenças não eram tantas, talvez o tipo de retorno seja a maior diferença da forma como aprendemos em JavaScript. Essas funções podem retornar um valor de algum tipo específico, por exemplo, int, double, Integer, String, etc. Na verdade, eles podem retornar qualquer coisa, não apenas valores, mas também estruturas inteiras, (veremos isso mais adiante), mas devemos indicar que tipo de retorno existem.

Vejamos alguns exemplos:

|  |
| --- |
| **int** **soma**(**int** num1, **int** num2) |

|  |
| --- |
| **double** **calcularTotal**(**double** número, **int** quantidade) |

Desta forma, teremos uma função definida com um valor de retorno.

Porém, existem outros tipos de funções das quais não retornam nada, e nesse caso devemos indicar o tipo de retorno colocando a palavra reservada void. Vejamos um exemplo:

|  |
| --- |
| **void** **mostrarMensagem**(String mensagem) |

Usamos funções do tipo void quando queremos que nossa função execute apenas uma série de etapas ou ações e não retorne nada.

**Desenvolvendo funções**

Até agora, vimos como definir uma função, a seguir veremos o que varia na implementação, por isso vamos ter duas situações. Uma em que a função tem um valor de retorno e outra onde não retorna nada.

Neste primeiro caso, devemos incluir um retorno com o valor esperado, este tipo de valor deve corresponder ao tipo de dado indicado como o tipo de retorno.

|  |
| --- |
| **int** **soma**(**int** num1, num2) {  **return** num1 + num2; } |

**O valor retornado deve ser do mesmo tipo que é indicado!**

Caso tenha uma função que n

**Exercício de Sintaxe Java**

# Objetivo

Praticar a sintaxe usando a linguagem de programação Java.

**Exercício 1**

Digite seu nome e sobrenome separadamente, a partir dessas variáveis ​​obtenha em uma terceira, suas iniciais e sua data de nascimento, para a data digite 3 valores que representam dia, mês e ano.

Em seguida, exiba uma mensagem, como se fosse um token, com seu nomes, as iniciais do seu nome e sobrenome e sua data de nascimento no formato "dd / mm / aaaa"

**Nota:** Use alguma função para resolver, por exemplo, dados os 3 valores, retorne a data como uma string.

**Até a próxima!**

**Exercício Funções**

**Exercício 1**

Faça uma função que, dado um número, indica se é um número primo ou não. Um número primo é aquele que só pode ser dividido por 1 e ele mesmo. Por exemplo: 25 não é primo, pois 25 é divisível por 1, 5 e 25. 17 é primo porque só pode ser dividido por 1 e por 17.

Para usar esta função que iremos criar em um programa, podemos permitir a entrada de apenas um número para verificar se o número é primo ou não.

 Para resolvê-lo, use a função obtida no exercício desenvolvido na aula assíncrona, Boolean e Divisível (int n, int divisor).

**Exercício 2**

Escreva uma função que receba três números e retorne o máximo dos três números.

**int maximoEntreTresNumeros(int umNumeroA, int umNumeroB, int umNumeroC)**

Em seguida, faça um programa que permita a entrada de 3 valores, use a função e exiba o resultado.

**Exercício Funções**

**Exercício 1**

Faça uma função que, dado um número, indica se é um número primo ou não. Um número primo é aquele que só pode ser dividido por 1 e ele mesmo. Por exemplo: 25 não é primo, pois 25 é divisível por 1, 5 e 25. 17 é primo porque só pode ser dividido por 1 e por 17.

Para usar esta função que iremos criar em um programa, podemos permitir a entrada de apenas um número para verificar se o número é primo ou não.

 Para resolvê-lo, use a função obtida no exercício desenvolvido na aula assíncrona, Boolean e Divisível (int n, int divisor).

**Exercício 2**

Escreva uma função que receba três números e retorne o máximo dos três números.

**int maximoEntreTresNumeros(int umNumeroA, int umNumeroB, int umNumeroC)**

Em seguida, faça um programa que permita a entrada de 3 valores, use a função e exiba o resultado.

**Vamos nos exercitar com desafios extras**

Tente resolver cada desafio criando o código com o IDE IntelliJ. Teste o resultado e discuta-o com seus colegas de mesa.

**Desafio 1**

Modifique a função que você escreveu no Desafio 2 do "Pedra-Papel-Tesoura" para jogar a variável Spock: Há mais uma possibilidade, o Sr. Spock - formado pela mão Vulcan de Jornada nas Estrelas - tem outros poderes de acordo com:

* Papel desaprova Spock
* Spock quebra uma tesoura
* Spock vaporiza pedra

**Desafio 2**

Por motivos trabalhistas, você pode trabalhar até 40 horas semanais, se trabalhar mais horas são consideradas horas extras cujo valor é 50% a mais que a jornada normal de trabalho. Crie um programa que possa relatar o salário semanal da programadora Julieta. Ela trabalhou na última semana "x" número de horas e o valor por semana  é de 875.

Agora é a sua vez!

# Diagramas UML

A seguir, a ***Unified Modeling Language*** ou **UML** é apresentada. Vamos usar a UML a partir de agora como principal meio de modelar nossas soluções em diagramas.

## Especificando uma Classe

Em nosso diagrama, cada classe é representada por um retângulo que conterá três divisões:

##### **Nome da Classe**

##### **Atributos**

##### **Métodos**

Tabela

Descrição gerada automaticamente

### Especificando o nome de uma classe

Você deve escolher um **nome descritivo** para a classe que está diagramando. O nome da classe sempre começa com uma letra maiúscula. Para nomes compostos, usamos o estilo de escrita em ***upper camel case.***

### Especificação de atributo

**Diagrama

Descrição gerada automaticamente**

### Especificação do método

**Diagrama

Descrição gerada automaticamenteEsquema Geral**

**Uma imagem contendo Aplicativo

Descrição gerada automaticamente**

### Exemplos

**Tabela

Descrição gerada automaticamente**

**Tabela

Descrição gerada automaticamente**

# Diagramas UML

A seguir, a ***Unified Modeling Language*** ou **UML** é apresentada. Vamos usar a UML a partir de agora como principal meio de modelar nossas soluções em diagramas.

## Especificando uma Classe

Em nosso diagrama, cada classe é representada por um retângulo que conterá três divisões:

##### **Nome da Classe**

##### **Atributos**

##### **Métodos**

Tabela

Descrição gerada automaticamente

### Especificando o nome de uma classe

Você deve escolher um **nome descritivo** para a classe que está diagramando. O nome da classe sempre começa com uma letra maiúscula. Para nomes compostos, usamos o estilo de escrita em ***upper camel case.***

### Especificação de atributo

**Diagrama

Descrição gerada automaticamente**

### Especificação do método

**Diagrama

Descrição gerada automaticamenteEsquema Geral**

**Uma imagem contendo Aplicativo

Descrição gerada automaticamente**

### Exemplos

**Tabela

Descrição gerada automaticamente**

**Tabela

Descrição gerada automaticamente**

# Diagramas UML

A seguir, a ***Unified Modeling Language*** ou **UML** é apresentada. Vamos usar a UML a partir de agora como principal meio de modelar nossas soluções em diagramas.

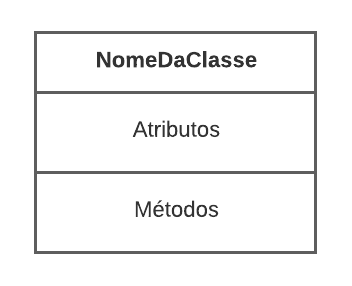
## Especificando uma Classe

Em nosso diagrama, cada classe é representada por um retângulo que conterá três divisões:

##### **Nome da Classe**

##### **Atributos**

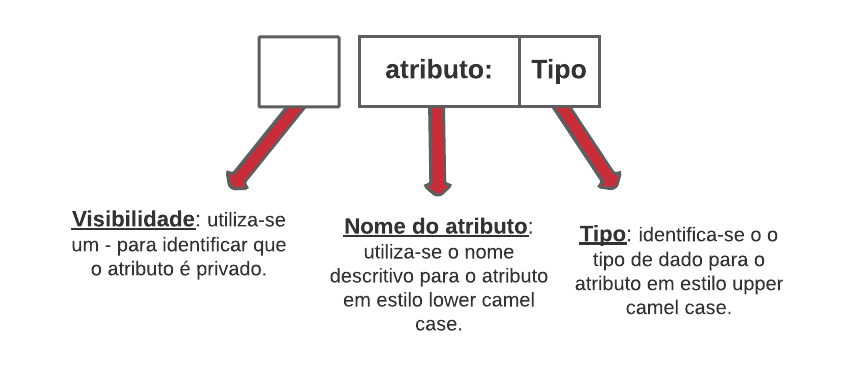
##### **Métodos**



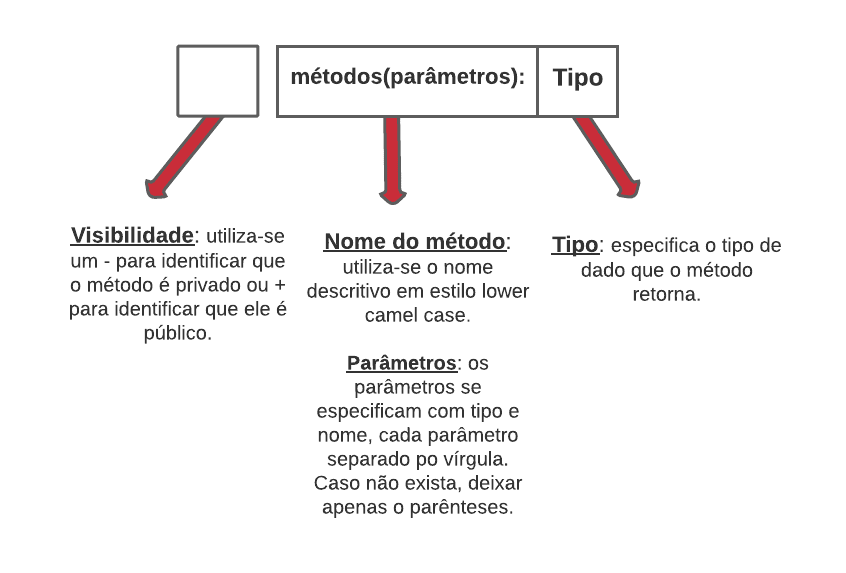
### Especificando o nome de uma classe

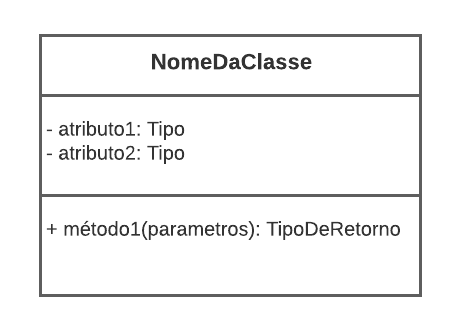
Você deve escolher um **nome descritivo** para a classe que está diagramando. O nome da classe sempre começa com uma letra maiúscula. Para nomes compostos, usamos o estilo de escrita em ***upper camel case.***

### Especificação de atributo

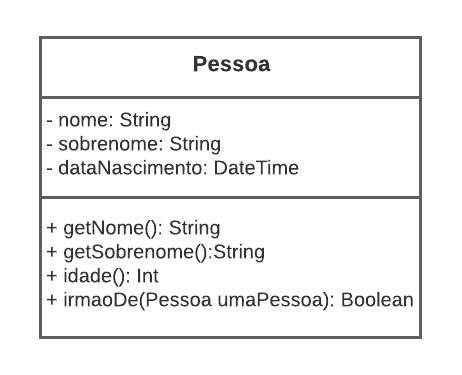
****

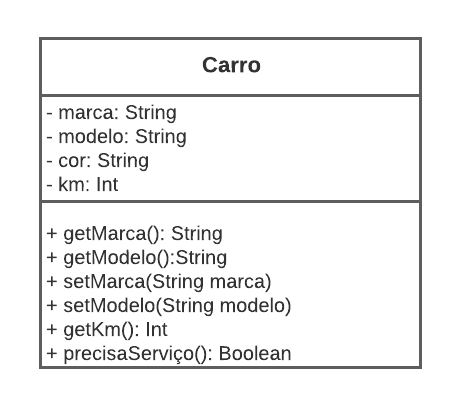
### Especificação do método

**Esquema Geral**

****

### Exemplos

****

****

Modelar - questões bancárias

Banco digital

Normalmente, um banco oferece diversos serviços a pessoas, empresas e instituições.

Esses serviços bancários geralmente incluem pacotes ou combinações para seus

clientes. Focamo-nos particularmente na análise dos chamados cartões de crédito e

cartões de débito.

Cartão de débito: permite transações vinculadas ao nosso saldo em conta bancária, é

como se estivéssemos utilizando o dinheiro economizado na conta sem ter que sacar o

dinheiro físico de um caixa eletrônico. Na frente, podemos ver que tem o nome do

titular, um número, dia e mês de validade.

Cartão de crédito: nos permite fazer compras sem a necessidade de ter a quantia exata

em conta, o banco nos autoriza um valor limite de financiamento, que é reduzido a

cada vez que utilizamos o cartão de crédito. As compras são parceladas e acrescidas de

juros, cada cartão tem juros e um valor máximo de parcelamento. Quando pagamos o

resumo do cartão, uma parte do limite do financiamento é recuperada, que se

completa quando terminamos de pagar nossas dívidas, então sempre temos um saldo

consumido e um saldo não pago.

Atividade

Crie um diagrama UML para o cartão de crédito e outro para o cartão de débito, com

seus atributos e métodos incluindo um construtor adequado para cada um.

Agora é a sua vez!

Modelar - questões bancárias

Banco digital

Normalmente, um banco oferece diversos serviços a pessoas, empresas e instituições.

Esses serviços bancários geralmente incluem pacotes ou combinações para seus

clientes. Focamo-nos particularmente na análise dos chamados cartões de crédito e

cartões de débito.

Cartão de débito: permite transações vinculadas ao nosso saldo em conta bancária, é

como se estivéssemos utilizando o dinheiro economizado na conta sem ter que sacar o

dinheiro físico de um caixa eletrônico. Na frente, podemos ver que tem o nome do

titular, um número, dia e mês de validade.

Cartão de crédito: nos permite fazer compras sem a necessidade de ter a quantia exata

em conta, o banco nos autoriza um valor limite de financiamento, que é reduzido a

cada vez que utilizamos o cartão de crédito. As compras são parceladas e acrescidas de

juros, cada cartão tem juros e um valor máximo de parcelamento. Quando pagamos o

resumo do cartão, uma parte do limite do financiamento é recuperada, que se

completa quando terminamos de pagar nossas dívidas, então sempre temos um saldo

consumido e um saldo não pago.

Atividade

Crie um diagrama UML para o cartão de crédito e outro para o cartão de débito, com

seus atributos e métodos incluindo um construtor adequado para cada um.

Agora é a sua vez!

**Diagramação com UML**

Agora que sabemos um pouco mais sobre diagramação, vamos analisar alguns casos. Em cada um devemos pensar nos atributos, responsabilidades e construtor certo para o problema de acordo com o contexto.

**Exercício N° 1**

Qualquer dia no veterinário consiste em receber o velho Labrador Amber, que vem com a sua dona Anita. O Dr. Julio o espera na sala 1 para atendê-lo, enquanto vemos a gatinha Miluz, de 3 meses, saindo da sala 2. Cada veterinário tem um escritório designado. Como sempre, cada animal é pesado, medido e registrado em seu estado geral (razoável, bom ou muito bom). Os clientes pagam pela consulta no final do mês, ou seja, o valor das consultas e quantas foram acumuladas.

**Atividade**

É necessário identificar no texto quais são as classes que se apresentam na narrativa. Desenhe-os em um diagrama de classes com atributos, métodos e construtores apropriados.

**Exercício N° 2**

O veterinário exige um abastecimento constante de diferentes caixas de mantimentos que são adquiridos para posterior revenda ou para serem utilizados durante as operações nas consultas que ocorrem nos consultórios médicos. Essas caixas de suprimentos possuem características diferentes, entre as quais podemos destacar: peso total, procedência, nome do fornecedor, descrição do conteúdo e se deve ser tratado com cuidado no caso de conter material frágil.

**Atividade**

Detecte os atributos e métodos que o objeto de caixa de provisionamento deve ter, considere adicionar um construtor correspondente e fazer a diagramação com a UML.

Agora é a sua vez!

Para resolver um problema, devemos criar uma classe cliente. Um cliente tem um número de cliente, o nome e uma dívida. Para registrar esse cliente, teremos que possuir seu número e seu nome. Um cliente pode aumentar sua dívida ou quitar.

|  |
| --- |
| **Cliente** |
| * numeroCliente: Int * nome: String * divida: Double |
| * Cliente( String numero, String nome) * aumentarDivida(Double valor) * pagarDivida() |