description: Uma classe bem estruturada não quer guerra com ninguém

Anatomia das classes

A escrita de códigos de um programa é feito através da composição de palavras pré-definidas pela linguagem com as expressões que utilizamos para determinar o nome do nossos arquivos, classes, atributos e métodos.

É muito comum mesclarmos expressões no idioma americano com o nosso vocabulário. Existem projetos que recomendam que toda a implementação do seu programa seja escrita na língua inglesa.

Sintaxe de declaração de uma nova classe:

!

- 99,9% das nossas classes iniciarão com `public class;`
- Toda classe precisa de nome, exemplo `MinhaClasse;`
- O nome do arquivo deve ser idêntico ao nome da classe pública;
- Após o nome, definir o corpo `{ }` , onde iremos compor nossas classes com atributos e métodos.

!

- É de suma importância que agora você consiga se localizar dentro do conjunto de chaves `{ }` existentes em sua classe.
- Dentro de uma aplicação, **recomenda-se que somente uma classe possua o método** `main`, responsável por iniciar todo o nosso programa.
- O método main recebe seu nome `main`, sempre terá a visibilidade `public`, será difinido como `static`, não retornará nenhum valor com `void` e receberá um parâmetro do tipo array de caracteres `String[]`.

Padrão de nomenclatura

Quando se trata de escrever códigos na linguagem Java, é recomendado seguir algumas convenções de escrita. Esses padrões estão expressos nos itens abaixo:

- **Arquivo .java**: Todo arquivo .java deve começar com letra MAIÚSCULA. Se a palavra for composta, a segunda palavra deve também ser maiúscula, exemplo:

  `Calculadora.java`, `CalculadoraCientifica.java`

- **Nome da classe no arquivo**: A classe deve possuir o mesmo nome do arquivo.java, exemplo:

```
// arquivo CalculadoraCientifica.java

public class CalculadoraCientifica {

}
```

- **Nome de variável**: toda variável deve ser escrita com letra minúscula, porém se a palavra for composta, a primeira letra da segunda palavra deverá ser MAIÚSCULA, exemplo: `ano` e `anoFabricacao`. O nome dessa prática para nomear variáveis dessa forma se chama "camelCase".

{% hint style="info" %} Existe uma regra adicional para variáveis quando na mesma queremos identificar que ela não sofrerá alteração de valor, exemplo: queremos determinar que uma variável de nome **br** sempre representará **"Brasil"** e nunca mudará seu valor, logo, determinamos como escrita o código abaixo: {% endhint %}

```
 String BR = "Brasil"
double PI = 3.14
int ESTADOS_BRASILEIRO = 27
int ANO_2000 = 2000
```

{% hint style="danger" %} Recomendações: Para declarar uma variável nós podemos utilizar caracteres, números e símbolos, porém devemos seguir algumas regras da linguagem. {% endhint %}

- Deve conter apenas letras, _ (underline), $ ou os números de 0 a 9
- Deve obrigatoriamente se iniciar por uma letra (preferencialmente), _ ou $, jamais com número
- Deve iniciar com uma letra minúscula (boa prática – ver abaixo)
- Não pode conter espaços
- Não podemos usar palavras-chave da linguagem
- O nome deve ser único dentro de um escopo

```
 // Declação inválida de variáveis

int numero&um = 1; //Os únicos símbolos permitidos são _ e $
int 1numero = 1;    //Uma variável não pode começar com númerico
int numero um = 1; //Não pode ter espaço no nome da variável
int long = 1; //long faz parte das palavras reservadas da linguagem

 // Declaração válida de veriáveis
int numero$um = 1;
int numero1 = 1;
int numeroum = 1;
int longo = 1;
```

Declarando variáveis e métodos

Como identificar entre declaração de variáveis e métodos em nossa programa? Existe uma estrutura comum para ambas as finalidades, exemplo:

- Declarar uma variável em Java segue sempre a seguinte estrutura:

```
 // Estrutura

Tipo NomeBemDefinido = Atribuição (opcional em alguns casos)

// Exemplo

int idade = 23;
double altura = 1.62;
Dog spike; //observe que aqui a variável spike não tem valor, é normal
```

- Declarando métodos em Java segue uma estrutura bem simples:

```
// Estrutura

TipoRetorno NomeObjetivoNoInfinitivo Parametro(s)

//Exemplo

int somar (int numeroUm, int numero2)

String formatarCep (long cep)
```

{% hint style="warning" %} Como parte da estrutura de declaração de variáveis e métodos também temos o aspecto da **visibilidade**, mas ainda não é necessário nesta etapa de estudos. {% endhint %}

Identação

Basicamente **indentar** é um termo utilizado para escrever o código do programa de forma hierárquica, facilitando assim a visualização e o entendimento do programa.

!

Abaixo, veja um exemplo de um algoritmo de validação de aprovação de estudante. Em uma aba, temos um código sem identação nenhuma, e na outra aba, temos o mesmo código seguindo um padrão de identação. Observe como é muito mais fácil entender a hierarquia do código na segunda aba.

{% tabs %} {% tab title="Sem Identação" %}

```
// arquivo BoletimEstudantil.java

public class BoletimEstudantil {
public static void main(String[] args) {
int mediaFinal = 6;
if(mediaFinal<6)
System.out.println("REPROVADO");
else if(mediaFinal==6)
System.out.println("PROVA MINERVA");
else
System.out.println("APROVADO");
}
}
```

{% endtab %}

{% tab title="Com Identação" %}

```
public class BoletimEstudantil {
    public static void main(String[] args) {
        int mediaFinal = 6;
        if (mediaFinal < 6)
            System.out.println("REPROVADO");
        else if (mediaFinal == 6)
            System.out.println("PROVA MINERVA");
        else
            System.out.println("APROVADO");
    }
}
```

{% endtab %} {% endtabs %}

Organizando arquivos

À medida que nosso sistema vai evoluindo, surgem novos arquivos (código fonte) em nossa estrutura de arquivos do projeto. Isso exige que seja realizado uma organização destes arquivos através de pacotes (packages).

!Ilustração de uso de pacotes

Com o uso de pacotes as nossas classes (.java) passam a possuir duas identificações, o nome simples e nome qualificado:

- **Nome Simples**: Nome do arquivo, exemplo `ContaBanco`.
- **Nome Qualificado**: Nome do pacote + nome do arquivo, exemplo: `com.suaempresa.ContaBanco`.

Java Beans

Umas das maiores dificuldades na programação é escrever algoritmos legíveis a níveis que sejam compreendidos por todo seu time ou por você mesmo no futuro. Para isso a linguagem Java sugere, através de convenções, formas de escrita universal para nossas classes, atributos, métodos e pacotes.

Variáveis

Mais cedo já aprendemos algumas regras de declaração de variáveis, mas agora iremos conhecer algumas sugestões de de nomenclatura:

- Uma variável deve ser clara, sem abreviações ou definição sem sentido;
- Uma variável é sempre no singular, **exceto quando se referir a um array ou coleção**.
- Defina um idioma único para suas variáveis. Se você for declarar variáveis em inglês, defina todas em inglês.

Não recomendado

```
double salMedio = 1500.23  //variável abreviada, o que dificulta a compreensão
String emails = "aluno@escola.com" //confuso se a variável seria um array ou único e-mail
String myName = "JOSEPH" //se idioma pt-BR, o valor poder ser de outro idioma mas o nome da variável n
```

Recomendado

```
 double salarioMedio=1500.23;
String email ="aluno@escola.com";
String [] emails = {"aluno@escola.com","professor@escola.com"}
String meuNome = "JOSEPH"
```

Métodos

Os métodos deverão ser nomeados como verbos, através de uma mistura de letras minúsculas e maiúsculas. Em princípio todas as letras que compõem o nome devem ser mantidas em minúsculo, com exceção da primeira letra de cada palavra composta a partir da segunda palavra.

Exemplos sugeridos para nomenclatura de métodos:

```
 somar(int n1, int n2){}

abrirConexao(){}

concluirProcessamento() {}

findById(int id){} // não se assuste, você verá muito método em inglês em sua jornada

calcularImprimir(){} // há algo de errado neste método, ele deveria ter uma única finalidade
```

# #