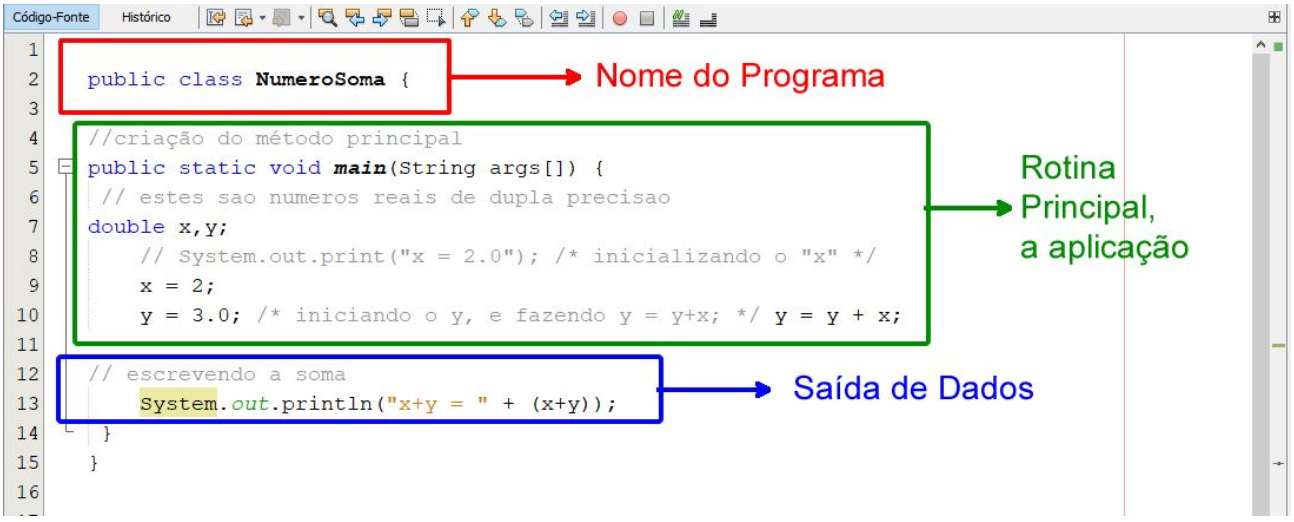
Aula de Revisão II - Lógica - Java

Estrutura do Programa Java

As primeiras coisas que devem ser abordadas para começar a desenvolver em qualquer linguagem são:

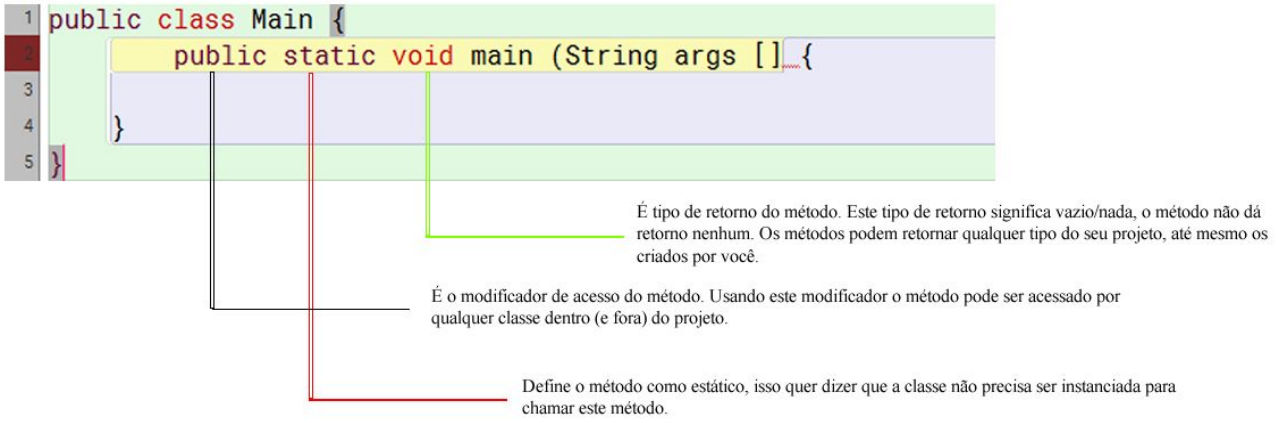
- Bloco do programa;
- Declarar variáveis;
- Sintaxes dos comandos;



Classe Main

Quando escrevemos o código fonte de um programa em Java e o compilamos, ele é transformado em ***bytecode***, que nada mais é que uma linguagem intermediária interpretada por uma máquina virtual, a **JVM (Java Virtual Machine)**. É a JVM que faz a ponte entre seu código fonte e a máquina, convertendo os *bytecodes*. Alguns criticam esse reprocessamento porque a linguagem perderia desempenho se comparada com outras, como o C, em que compilamos o código diretamente para linguagem de máquina.
Um projeto em Java pode ser composto por **diversas classes**. Mas uma delas deve ser **responsável pela execução do programa, um ponto de partida**.

Um início. Normalmente chamamos essa classe de **Programa**, **Principal, Main, Teste, etc**. **Essa classe não possui atributos, e normalmente possui apenas um método chamado de main (principal, em inglês). Nesta classe inserimos os comandos para o programa "funcionar". Esta é a classe que será "executada". E que nos retornará finalmente algo na tela.**



(String[ ] args)

Toda a classe Principal possuirá uma linha com o seguinte comando: public static void main(String args[]) - Este comando especifica que a classe é estática, sem retorno e principal. Abaixo deste comando devemos escrever todos os passos que devem ser seguidos pelo programa.Define que o método deve receber como parâmetro um array de String (nomeado args). Nesse caso específico: este parâmetro serve para caso seu programa precise receber algum valor como argumento, isso é muito comum quando o programa é iniciado por outro programa ou pelo terminal (CMD, Shell, Bash, etc.).

MÉTODOS DE SAÍDA DE DADOS (IMPRESSÃO NA TELA)

Método System.out.println()

A instrução System.out.println(), gera uma saída de texto entre aspas duplas significando uma String, criando uma nova linha e posicionando o cursor na linha abaixo, o que é identificado pela terminação **"ln"**.

Sempre temos a possibilidade de mandar mensagens ao usuário, essas mensagens podem conter somente texto, somente respostas, textos e respostas, texto, respostas e mais texto... Isso é possível através de um método de saída de dados, o método out, este método é da classe do sistema, **class System**, portanto o comando fica: **System.out.print("mensagem")**;

Exemplos:

```java
public class Main {
    public static void main (String args[]) {
        System.out.println("Aqui via o texto que aparece para o usuário!");
    }

}
```

```java
public class Main {
    public static void main (String args[]) {
        System.out.println("Aqui via o texto que aparece para o usuário!");
        System.out.println("Sempre quando eu quero colocar quebra de linha ....");
        System.out.println("Eu coloco -ln- junto do print");
    }

}
```

Senão ....

```java
public class Main {
    public static void main (String args[]) {
        System.out.print("Aqui vai o texto que aparece para o usuário!");
        System.out.print("Posso deixar sem o -ln- ...");
        System.out.print("Que fica tudo na mesma linha");
    }

}
```

Entendendo a estrutura de um código Java

Fonte: https://www.devmedia.com.br/entendendo-a-estrutura-de-um-codigo-java/24622

Arquivo Fonte(Source File) :

Em java, cada classe(class) é colocada em um arquivo source, esses arquivos representam partes de uma aplicação ou toda a aplicação(no caso de programas muito pequenos).

Arquivos source são gerados com a extensão .java e podem ser facilmente criados utilizando o notepad por exemplo. Esses arquivos devem possuir o mesmo nome da classe que possuem.

Classe(Class)

```
1 public class MyClass{
2 }
```

Em uma classe java são colocados os métodos(methods), funções ou procedimentos. Todo o código deve estar em alguma classe, pois quando executamos algum aplicativo java nós estamos, na verdade, executando uma classe.

Diferentemente de um arquivo fonte que só pode conter uma classe, uma classe pode conter vários métodos. Em java a classe deve estar em um Arquivo Fonte(Source File) e deve ir com um par de chaves “{}”, são nessas chaves que serão colocados os métodos. Lembrando que uma classe sempre inicia com letra maiúscula.

```
1 public class MyClass{
2 // código vai aqui
3 }
```

Métodos(Methods)

Os métodos, funções ou procedimentos são onde declaramos o código das nossas aplicações java. Assim como classes, os métodos em java devem ser

escritos acompanhados de um par de chaves “{}” no final. Lembrando que um método sempre inicia com letra minúscula.

```
public class MyClass{
public void meuMetodo(/*argumentos*/){
}
 }
```

Anatomia de uma Classe (imagem acima)

- public = Refere-se a visibilidade desta classe. Quando dizemos que uma classe é de visibilidade “public” , estamos dizendo que esta classe poderá ser acessada por outras classes.
- class = Mostramos que estamos criando uma classe.
- MyClass = Refere-se ao nome da classe que estamos criando. Nesse caso , o nome da minha classe será “MyClass”.
- { } = As chaves indicam até onde certa classe ou método se estende. O código que queremos inserir nesta classe deverá ser escrito dentro do espaço das chaves.

Anatomia de uma Método

```
public class MyClass{
public void meuMetodo(/*argumentos*/){
}
 }
```

- public = Do mesmo modo que uma classe, refere-se a visibilidade deste método. Quando dizemos que o método é de visibilidade “public” ,

estamos dizendo que este método poderá ser acessado por outras classes.

- void = Refere-se ao tipo de retorno que esse método terá. Nesse caso, como o tipo de retorno é “void”, ou seja , “vazio”, esse método não retornará valor nenhum.
- meuMetodo = Assim como numa classe, refere-se ao nome do método que estamos criando. Nesse caso, o nome do meu método será “meuMetodo”.
- (/*argumentos*/) = Refere-se aos argumentos que serão passados para esse método, sendo opcional. Caso não seja necessário passar argumentos, simplesmente deixaríamos os parênteses vazios “( ) “. De contrário é necessário escrever o tipo da variável a ser passada e um nome para essa variável “(int valor)”.
- { } = As chaves indicam até onde certa classe ou método se estende. O código que queremos inserir nesta classe deverá ser escrito dentro do espaço das chaves.

O Método main

Quando o java virtual machine(JVM) inicia, ele procura na sua classe principal por um método muito específico, chamado de método main.

Uma aplicação java obrigatoriamente deverá possuir pelo menos uma classe e um método main, pois é por esse método main que o JVM começará a executar. Como o método main é padrão para qualquer aplicação java, há algumas regras que devem ser cumpridas para o funcionamento desse método. Por regra , todo método main deverá ser : Público, estático , sem retorno(void), com nome de “main”, e deverá receber como argumento um array do tipo String.

```
public class MyClass{
public static void main(String[] args){
}
}
```

Anatomia do método main (imagem acima)

- public = Do mesmo modo que um método comum, refere-se a visibilidade deste método. Quando dizemos que o método é de visibilidade “public” , estamos dizendo que este método poderá ser acessado por outras classes.
- static = Nos garante que somente haverá uma, e não mais que uma, referência para nosso método main, ou seja, todas as instâncias da classe irão compartilhar a mesma cópia do método main.
- void = Assim como um método comum, refere-se ao tipo de retorno que esse método terá. Nesse caso, como o tipo de retorno deve ser “void”, ou seja , “vazio”, esse método não retornará valor nenhum.
- (String[] args) = Refere-se aos argumentos que serão passados para esse método, sendo obrigatório no caso do método main
- {} = Assim como um método comum , As chaves indicam até onde certa classe ou método se estende. O código que queremos inserir neste método deverá ser escrito dentro do espaço das chaves.

Condicionais

Executa um texto Lógico, sempre dentro de um retorno boleano (verdadeiro ou falso)

Condição, (verdadeiro; falso)

A média da disciplina é 60
Se a média do aluno for **maior ou igual** a 60, **então** o aluno está aprovado, **senão**, o aluno está reprovado.

if (variavel >= 60)
Aprovado
else
Reprovado