Aula 02 - Lógica - Parte 2

Alguns conceitos

Computadores são máquinas que não tem iniciativa, elas executam as instruções que são passadas a elas. Informamos essas instruções por meio de algoritmos, porém para que os computadores entendam os algoritmos escritos é necessário transformar a sequência de passos que escreve-se em linguagem natural para uma linguagem que possa ser "entendida" pelo computador. Essas linguagens são chamadas de linguagens de programação.

Memória

O primeiro passo para um programa ser executado em um computador é o carregamento do programa para a memória. Memórias são todos os componentes internos do computador que armazenam informações como a memória RAM e o disco rígido (HD). Esse armazenamento pode ser temporário, quando as informações são perdidas ao desligar o computador ou permanente, quando ao desligar o computador a informação permanece no dispositivo para os próximos usos.
Quando um programa é executado (tornando-se o que chamamos de processo), todo o seu conteúdo é colocado na memória RAM (*Random Access Memory – Memória de Acesso Randômico*) do computador. Sendo que é somente nela que o mesmo é executado. Essa memória é utilizada para armazenar as instruções dos programas, que são os comandos dados ao computador pelos algoritmos que serão construídos.

Como um programa é tratado na memória?
Todo o programa que é posto da memória RAM do computador é dividido em três áreas.

Área de Dados: É o local destinado para o armazenamento dos dados vindos do usuário, cálculos, saídas. Enfim, é o espaço que o algoritmo tem para trabalhar.
Área de Código: É o local onde está o algoritmo propriamente dito.

Pilha: É quem garante a continuidade do programa. Se o programa efetua "saltos" dentro do código, é a pilha que mantém a informação de onde o código estava antes do salto consequentemente garantindo um retorno correto ao final do salto. Pilhas utilizam uma estrutura de dados manipulada pelo que chamamos de **FILO (*First Input Last Output – Primeiro que entra é o último que sai*).**
Para guardar os dados na memória em sua área de dados, criam-se "rótulos" para cada informação a ser armazenada e posteriormente acessada pelos programas. Esses rótulos são chamados de **variáveis**.

Variáveis

Basicamente, uma variável é um tipo de dado armazenado em computador. Chama-se variável, pois o valor contido nesse espaço de memória do computador é mutável ao longo da execução do programa. Esse valor pode ser lido do teclado, poderá ser obtido de um *hardware*, poderá ser o resultado de um cálculo dentro do programa, etc. Como bem se sabe, **o computador utiliza memória RAM** para armazenar dados temporários e essa memória é utilizada também pelos programas.

Quando se declara uma variável em um programa, está na verdade se definindo e reservando um espaço na memória para armazenar o valor que aquela variável conterá em determinado tempo de execução do programa. Para que o computador saiba qual o tamanho da informação que deverá ser armazenada e possa reservar na memória a quantidade necessária de espaço, é necessário se definir o tipo dessa variável.

Além do tipo, a variável precisará de um rótulo ou nome que a identifique dentro do programa e através desse nome a variável será referenciada em todo o programa.
A variável, antes de ser utilizada, precisa ser declarada. Na declaração, define-se o nome e o tipo da variável. Supondo que seja necessário fazer um programa que solicite ao usuário dois números, some esses dois números e apresente o resultado da soma para o usuário ao final, seriam necessárias as variáveis e seus respectivos tipos definidos.

Entendendo o código fonte

Na **variável *número1*** armazena-se o primeiro número digitado, na variável ***número2*** o segundo número e na variável *soma* o resultado da soma desses dois números. Importante notar que foi utilizado o tipo inteiro nessa declaração de variáveis, dessa forma o computador sabe que precisa reservar um espaço na memória suficiente para armazenar um número inteiro negativo ou positivo sem as casas decimais.

Tipos primitivos

O **tipo inteiro** utilizado no exemplo anterior **é um tipo primitivo no Java**, ou seja, ele é fornecido por esta linguagem de programação. Cada linguagem possui um conjunto de tipos primitivos "nativos" da linguagem.

Tipos primitivos normalmente incluem a possibilidade de armazenar valores numéricos **inteiros, valores com vírgula (decimais ou também chamados de números de ponto flutuante) , texto, valores lógicos (verdadeiro ou falso) entre outros.**

Alguns exemplos:

Alguns tipos primitivos em Java:

int: Toda e qualquer informação numérica que pertença ao conjunto dos números inteiros (negativo, zero ou positivo).

Exemplos:
filhos = 3
casas = 2
degraus = 14

double: Todo e qualquer valor numérico que pertença ao conjunto dos números reais (negativos, zero ou positivos).

Exemplos:
altura = 1,71
peso = 75,5
saldo = -50,20
temperatura = -4,2

string: Toda informação composta de um conjunto de caracteres alfanuméricos, numéricos (0..9), alfabéticos (A..Z, a..z) e ainda caracteres especiais (*, ?, !, @).

Exemplos:
nome = "José da Silva"
cidade = "Porto Alegre"
bairro = "Vila Velha"

Exemplos:
int n = 4;
long numero = 456L;
float pi = 3.14f;
double tamanho = 3.873457486513793;
string = nome;

Resumindo: **Variáveis são utilizadas para representar valores desconhecidos, porém necessários para a resolução de um problema e que poderão ser alterados de acordo com a situação.**

Constantes

Um determinado valor é constante quando não sofre alteração no decorrer do tempo, da mesma forma ocorre no desenvolvimento de software. O valor definido em uma constante será igual desde o início até o final do algoritmo.
Como as variáveis, as constantes possuem um nome e um tipo, porém se define um valor inicial para essa constante e ela não poderá mais ser alterada. Constantes são importantes para referenciar valores úteis ao longo do nosso programa, como o "Pi" para o cálculo da circunferência (3,1416).

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)

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 declararemos 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

```
public class MyClass{
}
```

- 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.

```
1 public class MyClass{
2 public static void main(String[] args){
3 }
4 }
```

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.