Ambiente de Desenvolvimento Java - *Java Development Kit* (Java 2 SDK ou JDK)

⇒ Código-Fonte (arquivo que conterá uma Classe): extensão **.java**. Uma Classe pode conter atributos (campos) e métodos mas somente isto não é necessário para que a Classe se torne executável. Para que isto ocorra, é necessário criar em uma Classe um método especial denominado **main**.

⇒ Arquivo-Objeto (*Bytecode*): extensão **.class.**

⇒ Ferramentas do Ambiente JDK para Plataforma Windows

- Edição de Código-Fonte: Bloco de Notas (*Notepad*).
- Compilação de Classes: *Prompt* de comando. No *Prompt* de comando, usar o comando **javac** (executa o Compilador) que recebe como argumento o nome de um arquivo-fonte, **com** a extensão .java, e cria como resultado um ou mais arquivos com extensão .class. Ex: javac Primeiro.Java
- Execução de Classes: *Prompt* de comando. No *Prompt* de comando, usar o comando **java** (executa o Interpretador) seguido do nome do arquivo-objeto **sem** a extensão .class. Ex: java Primeiro

OBS1: Antes de tentar compilar e executar Classes, fazer o seguinte:

a.) Adicionar ao *Path* o diretório das ferramentas do JDK (set path=%path%;c:\j2sdk1.4.1\bin;)
b.) Criar um Diretório no computador: c:\Programas

OBS2: É possível escrever várias Classes em um único arquivo .java. Porém, isto não é muito aconselhável pois, para cada Classe no arquivo .java, será criado um arquivo .class correspondente.

⇒ Primeiro Programa (Primeiro.java)
(Nós: editar, compilar e executar)

```
/*
   Isto é um comentário em bloco.
   Primeiro programa em Java.
*/

class Primeiro
  {
   public static void main (String[] argumentos)

     {

       System.out.println("Primeiro Programa em Java!");

     } // Isto é um comentário de uma única linha. Fim de main.
```

```
} // Fim da Classe Primeiro.
```

OBS3:

- Java diferencia caracteres maiúsculos de minúsculos.
- Conteúdo das Classes e Blocos de Código dos Métodos → delimitados por chaves - { }.

⇒ Método **main:** obrigatoriamente declarado como

public static void main (String[] argumentos)

Explicação

- public: faz com que o método seja visível de qualquer outra Classe (exigência da Java VM para executá-lo). (Nós: Visibilidade; Modificador de Acesso)
- static: um método declarado como static dispensa a criação de uma instância de sua Classe para que possa ser invocado. (Nós: modificador)
- void: "tipo de retorno". O método main nada deve retornar.
- String[]: array de strings. O método main deve receber como argumento um array de strings, que deverá ter um nome (**argumentos** ou outro). (Nós: esse array de strings conterá, para main, os args passados para a Classe executável via linha de comando do terminal)

Atributos de Classes em Java

⇒ Tipos de Dados Nativos (parte da linguagem)

Tipo	Faixa de valores	Notas
boolean	true ou false	Valor booleano, podendo assumir uma das constantes true (verdadeiro) ou false (falso).
char	0 a 65535	Caracteres representados em 16 bits, com suporte multilíngüe, podendo ser usados como tipos inteiros de 16 bits, com sinal.
byte	-128 a 127	Inteiro de 8 bits de precisão, com sinal.
short	-32768 a 32767	Inteiro de 16 bits de precisão, com sinal.
int	-2147483648 a 2147483647	Inteiro de 32 bits de precisão, com sinal.
long	-9223372036854775808 a 9223372036854775807	Inteiro de 64 bits de precisão, com sinal.
float	1.40129846432481707e-45 a 3.40282346638528860e+38	Ponto flutuante de precisão simples, armazenado em 32 bits, com sinal.
double	4.94065645841246544e-324 a 1.79769313486231570e+308	Ponto flutuante de precisão dupla, armazenado em 64 bits, com sinal.
String	Tamanho limitado à memória disponível.	Cadeia de caracteres que usam dois bytes por caractere. Strings podem ser vazias (zero caractere) e conter qualquer tipo de caractere.

Tabela 2.2: Tipos básicos de Java

⇒ Todos os tipos numéricos em Java preservam suas características e limitações independentemente de Sistema Operacional ou arquitetura de computador. Ex: int ocupa 4 Bytes independentemente da plataforma.

⇒ Dos tipos Inteiros (byte, short, int, long): int é o padrão para valores literais (valores numéricos constantes). Nas atribuições, os valores literais de Inteiro são convertidos automaticamente para os outros tipos. Ex: d = 25; (d é do tipo byte; 25 é int por padrão → compilador converte 25 para tipo byte)

⇒ Dos tipos em Ponto Flutuante (float, double): double é o padrão para valores literais. Para o caso de uma atribuição de uma variável do tipo float, deve-se usar um f ao final do valor literal. Ex: g = 5.0f; (g é do tipo float; 5.0 é double por padrão → f diz ao compilador que o literal deve ser armazenado como um float)

⇒ Operadores

Operador Aritmético	Ação
+	Adição
-	Subtração, Inversão de sinal
*	Multiplicação
/	Divisão
%	Extração do Módulo ou Resto da Divisão

OBS4: o resultado de um operador binário será sempre do mesmo tipo do maior operando. Exceção: int é o menor resultado para inteiros.

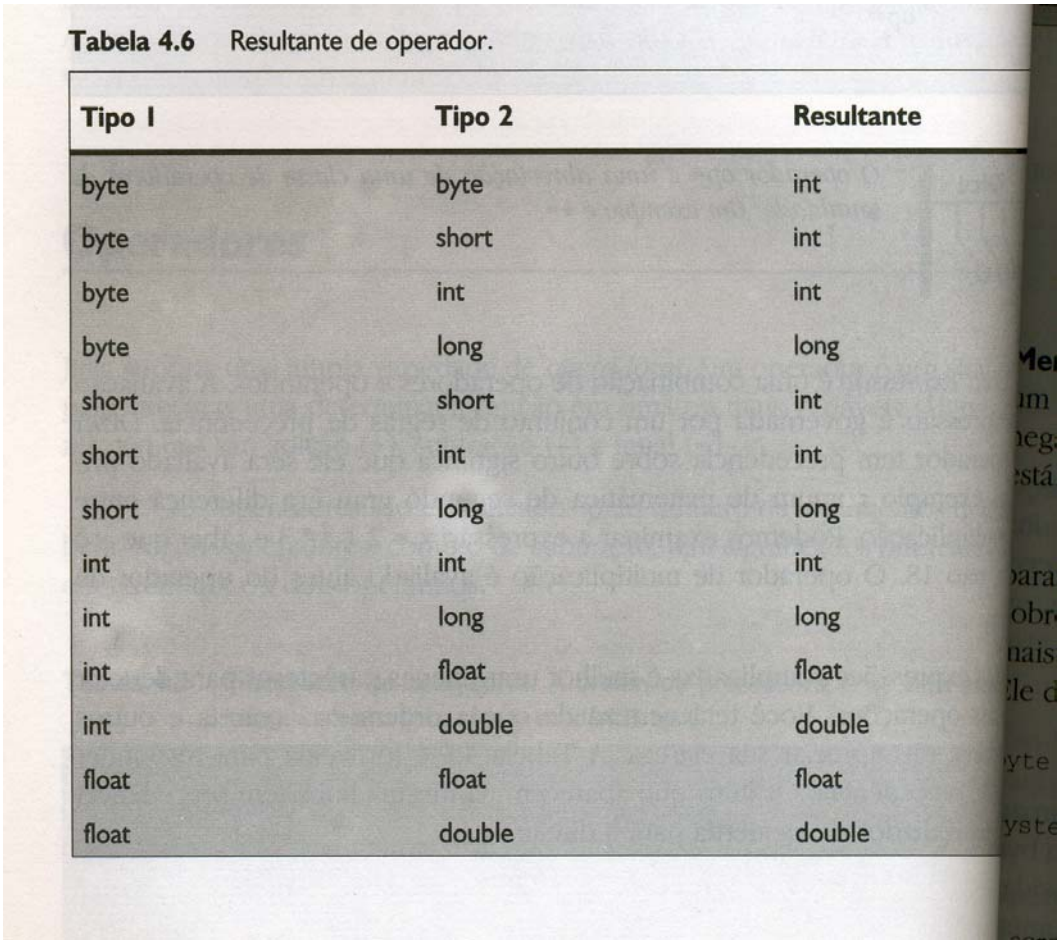

Tabela 4.6 Resultante de operador.

Tipo 1	Tipo 2	Resultante
byte	byte	int
byte	short	int
byte	int	int
byte	long	long
short	short	int
short	int	int
short	long	long
int	int	int
int	long	long
int	float	float
int	double	double
float	float	float
float	double	double

Operador Relacional	Ação
<	Menor que
>	Maior que
<=	Menor ou Igual
>=	Maior ou Igual
==	Igual
!=	Diferente

Operador Lógico	Ação
&&	E (AND) lógico
||	OU (OR) lógico
!	NÃO (NOT) lógico

⇒ Programa DecAtributos.java

```
class DecAtributos
  {
```

```
  public static void main (String[] argumentos)

   {

    //Declarando Atributos

    boolean valor1;
    byte valor2 = 100;
    short valor3 = 50;
    int valor4 = 750;
    float valor5;
    double valor6 = 20.2;
    String texto;


    // Início
    valor1 = false; // poderia iniciar na declaração da variável
    valor3 = (short)(valor2 + valor3);// byte + short => int
    valor4 = valor4 + valor3; //int + short => int
    valor5 = 7.8f;
    valor2 = 5;
    valor2  = (byte)(valor2*6);
    valor6 = valor4 + valor6;
    texto = "Testando!";

    System.out.println("Valor 1: " + valor1); // false
    System.out.println("Valor 2: " + valor2); // 30
    System.out.println("Valor 3: " + valor3); // 150
    System.out.println("Valor 4: " + valor4); // 900
    System.out.println("Valor 5: " + valor5); // 7.8
    System.out.println("Valor 6: " + valor6); // 920.2
    System.out.println("Texto: " + texto);    // Testando!


   } // Fim de main.

 }  // Fim da Classe DecAtributos.
```

OBS5: Perceber o uso da conversão explícita de tipo (cast) para operações em valor2 e valor3.

Métodos em Classes em Java

⇒ Métodos não podem ser criados dentro de outros métodos, nem fora da Classe a qual pertencem.

⇒ Declaração:
modificador-de-acesso tipo-ou-classe-de-retorno nome-do-método(lista-de-argumentos)

Ex: public boolean calcula(int a)

⇒ void: "tipo de retorno" para quando o método nada tiver para retornar.

⇒ Métodos que retornam algum valor diferente de void devem ter, em seu corpo, a palavra-chave return seguida de uma constante ou variável do tipo ou classe que foi declarada como sendo a de retorno do método. Ex: return true;

⇒ Programas ExempMetodo.java e Maior.java (calcula o maior entre 3 números. Compilar: javac *.java)

```
class ExempMetodos
  {
    public static void main (String[] argumentos)

      {


       // instância da Classe Maior. A instanciação é feita pelo operador new.
       Maior maiorNumero = new Maior();



       int num1, num2, num3, nm;
       num1 = 20; num2 = 30; num3 = 40;
       nm=maiorNumero.encontraMaior(num1, num2, num3);
       System.out.println("O maior numero e: " + nm);


      } // Fim de main.

  }  // Fim da Classe ExempMetodos.
```

```
class Maior
  {


       //Atributo
       int numMaior;


       // Método
       int encontraMaior(int num1, int num2, int num3)
       {
         if (num1 > num2)
            {                    //pode retirar as chaves; bom deixar.
              if (num1 > num3)
                 numMaior = num1;
              else
                numMaior = num3;

            }
          else
            {
```

```
            if (num2 > num3)
               numMaior = num2;
            else
               numMaior = num3;
          }
        return numMaior;

      }

} // Fim da Classe Maior.
```