**[](http://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&frm=1&source=images&cd=&cad=rja&docid=oOMt0HhPYsWWuM&tbnid=hlYbr3M5p7NIcM:&ved=0CAUQjRw&url=http://algoritmizando.com/desenvolvimento/site-com-curso-de-java-gratis-na-web/&ei=svvjUcn9K9H54AOsuYHQDQ&psig=AFQjCNHyppF0q2v0VL5I5dFHrztHJWGL1A&ust=1373981987566417)**

**Introdução à Programação Orientada a Objetos**

**1.1 Paradigmas de Programação**

**- Programação Procedural – Cobol**

**- Programação Estruturada – Pascal, C**

**- Programação Orientada a Objetos – C++, Java**

**- Programação Orientada a Eventos – Visual Basic**

**Programação Estruturada – organiza o código através de subprogramas (procedimentos e funções), recorrendo a apenas 3 estruturas de controle: seqüenciação, decisão e repetição.**

**Programação Orientada a Objetos – organiza o código em objetos, que são autônomos e trocam mensagens entre si durante a execução do programa, imitando o comportamento dos objetos do mundo real.**

**1.2 Tecnologia Java**

**A tecnologia Java compreende quatro domínios:**

**- uma linguagem de programação**

**- um ambiente de desenvolvimento**

**- um ambiente de execução**

**- um ambiente de distribuição**

**Java é uma linguagem de programação criada em 1991 por James Gosling da Sun Microsystems e esta empresa foi entretanto adquirida pela Oracle.**

**É a tecnologia que capacita muitos programas da mais alta qualidade, como utilitários, jogos e aplicativos corporativos, entre muitos outros.**

**O Java é executado em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e dispositivos de televisão.**

**1.3 Características da Linguagem**

**- Orientada a Objetos desde o início**

**- Familiar às linguagens C e C++**

**- Simples – fácil aprendizado**

**- Robusta – para aplicações confiáveis**

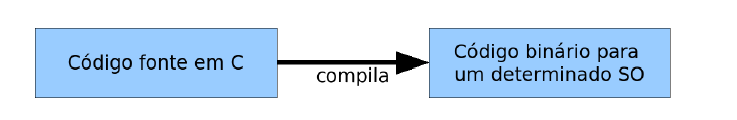
**- Segura – são executadas em ambiente próprio JRE (Ambiente de Execução Java) o que impede a intrusão de código malicioso.**

**- Portátil – programas podem ser executados em qualquer máquina, desde que instalado o JRE.**

**- Independência do Sistema Operacional**

**1.4 Máquina Virtual Java**

**Em uma linguagem de programação como C e Pascal, temos o seguinte quadro quando vamos compilar um programa.**

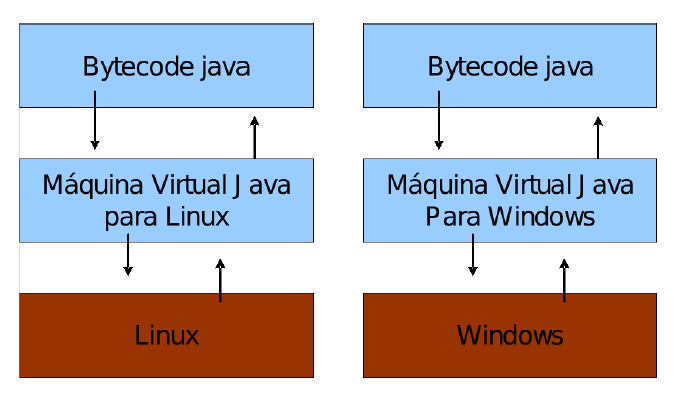
****

**O código fonte é compilado para uma plataforma e sistema operacional específicos. Muitas vezes, o próprio código fonte é desenvolvido visando uma única plataforma!**

**Esse código executável (binário) resultante será executado pelo sistema operacional e, por esse motivo, ele deve saber conversar com o sistema operacional em questão.**

**Isto é, temos um código executável para cada sistema operacional. É necessário compilar uma vez para Windows, outra para o Linux, etc...**

**Já o Java, se utiliza do conceito de máquina virtual (JVM – Java Virtual Machine – Máquina Virtual Java), onde existe, entre o sistema operacional e a aplicação, uma camada extra, responsável por “traduzir” - mas não apenas isso - o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento:**

****

**Dessa forma, a maneira com a qual você abre uma janela no Linux ou no Windows é a mesma: você ganha independência de sistema operacional. Ou, melhor ainda, independência de plataforma em geral: não é preciso se preocupar em qual sistema operacional sua aplicação está rodando, nem em que tipo de máquina, configurações etc.**

**Repare que uma máquina virtual é um conceito bem mais amplo que o de um interpretador. Como o próprio nome diz, uma máquina virtual é como um computador de mentira: tem tudo que um computador tem, é um programa que carrega e executa os aplicativos** [**Java**](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_Java)**, convertendo os** [**bytecodes**](https://pt.wikipedia.org/wiki/Bytecode_Java) **em código executável de máquina.**

**Sua aplicação roda sem nenhum envolvimento com o sistema operacional! Sempre conversando apenas com a Java Virtual Machine (JVM).**

**1.5 Instalação da JVM**

**Para ter a Máquina Virtual Java em seu sistema é necessário instalar o JRE (Java Runtime Environment), um programa gratuito que permite ao usuário rodar aplicativos Java em seu computador.**

**Para aqueles que desejam desenvolver aplicações, é necessário instalar o JDK (Java Development Kit), pacote que inclui tudo o que é necessário para escrever aplicações e também o JRE para poder rodá-los após finalizá-los.**

**Para fazer o download do JDK:**

[**http://www.oracle.com/technetwork/java/javase/downloads/index.html**](http://www.oracle.com/technetwork/java/javase/downloads/index.html)

**1.6 Fases de um programa Java:**

**Compreende quatro fases:**

**- Criação do código fonte (programa.java)**

**- Compilação do código, gerando o bytecode (programa.class)**

**- Interpretação do bytecode pela JVM**

**- Conversão do bytecode em linguagem de máquina**

**IDE – Integrated Development Environment – Ambiente de Desenvolvimento Integrado)**

**Programas que reúnem características e ferramentas de apoio ao desenvolvimento de software como o objetivo de aplicar este processo.**

**- BlueJ (www.bluej.org)**

**- NetBeans (www.netbeans.org)**

**- Eclipse (**[**www.eclipse.org**](http://www.eclipse.org)**)**

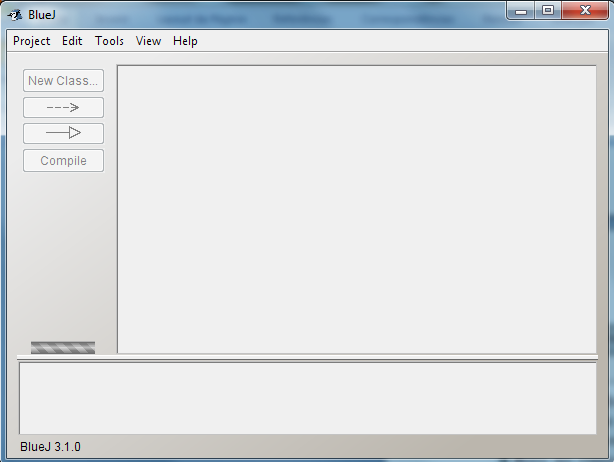
****

**O BlueJ é um ambiente gratuito de desenvolvimento de aplicações orientadas a objetos utilizando a linguagem de programação Java. Uma de suas características principais é a facilidade de aprendizagem de programação OO, através de recursos visuais interativos. Por exemplo, classes e relacionamentos podem ser definidos visualmente, e é possível verificar o comportamento dos objetos em memória durante a execução.**

**O ambiente BlueJ**

**Interface principal do BlueJ (figura 1). A grande área central é o Class Browser, e a parte inferior é chamada de ObjectBench.**

**O class browser possibilita definir visualmente uma estrutura de classes (usando um subconjunto simplificado da linguagem de modelagem UML), e também instanciar objetos interativamente, que são mostrados no ObjectBench. Os atributos dos objetos podem ser examinados através do Object Inspector (não mostrado na figura), e métodos podem ser executados via cliques do mouse.**



**Figura 1. Interface Principal do BlueJ**

**Construindo um projeto no BlueJ**

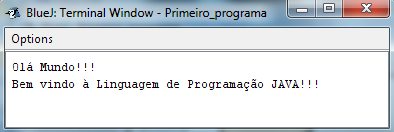
**Para criar um projeto, no menu principal deve-se acessar Project||New Project.**

**Exercício 1: Desenvolva uma aplicação em Java para mostrar as mensagens:**

**Olá Mundo!!!**

**Bem vindo à Linguagem de Programação JAVA!!!**

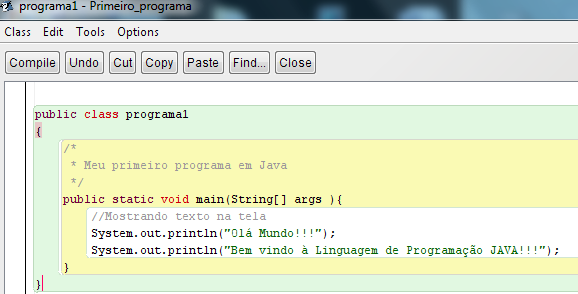
**Conforme a tela:**



**Nome do Projeto: Primeiro\_programa**

**Nome da Classe: programa1**

Teremos o código:



**public class programa1**

**{**

**/\***

**\* Meu primeiro programa em Java**

**\*/**

**public static void main( String[] args ){**

**//Mostrando texto na tela**

**System.out.println("Olá Mundo!!!");**

**System.out.println("Bem vindo à Linguagem de Programação JAVA!!!");**

**}**

**}**

**Entendendo nosso primeiro programa em Java**

**O nome da classe é Hello**

**● Em Java todo e qualquer código deve pertencer a uma classe**

**● Esta classe usa um identificador de acesso public. Indica que está acessível para outras classes de diferentes pacotes (pacotes são coleções de classes).**

**Quando executamos um programa escrito em Java, o sistema tem que ter um ponto de início, como haverá muitas classes e métodos, ele tem que executar a partir de algum lugar.**

**A JVM está programada para executar o método main, portanto se você compilar e executar um arquivo, será o método main (Principal) que primeiro executará, desta forma, tudo que estiver dentro das chaves {} do método será executado pela JVM.**

**A linha 2 que contém uma chave { indica o início de um bloco de instruções.**

**As próximas 3 linhas indicam um comentário em Java**

**Um comentário é uma explicação usada para a documentação do código.**

**- Não faz parte do programa em si, mas sim de sua documentação.**

**- É uma boa prática de programação adicionar comentários aos programas.**

**Java possui 3 tipos de comentários:**

**// código**

**/\* código \*/**

**/\*\***

**\*código**

**\*/**

**A instrução System.out.println(), mostra, na saída padrão, o texto descrito entre as aspas.**

**As últimas duas linhas, que contêm somente uma chave em cada, simbolizam, respectivamente, o fechamento do método main e da classe.**

**Uma instrução é composta de uma ou mais linhas terminadas por ponto-e-vírgula.**

**● Exemplo:**

**System.out.println(“Hello world”);**

**Um bloco é formado por uma ou mais instruções agrupadas entre chaves { } indicando que formam uma só unidade.**

**● Blocos podem ser organizados em estruturas aninhadas**

**Indefinidamente.**

**● Qualquer quantidade de espaços em branco é permitida**

**Exemplo:**

**public static void main (String[] args) {**

**System.out.println("Hello");**

**System.out.println("world”);**

**}**

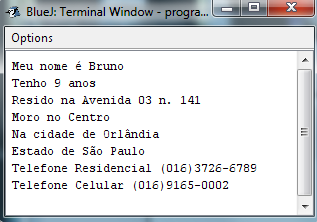
**Exercício 2:**

**Desenvolva uma aplicação (projeto) em Java utilizando-se do Bluej para exibir suas informações pessoais, conforme exemplo abaixo, considerando:**

**- Nome do Projeto: Prj\_Exercicio2**

**- Classe: DadosPessoais**

**Exemplo de Saída:**



**Usando a Formatação na Saída com a Instrução System.out.println**

**Trabalhando com “tela” – Saída Formatada com a instrução System.out.println()**

**Exercício 3:**

**Nome do Projeto: Prj\_Formatacao**

**Nome da Classe: Formatos**

**public class Formatos {**

**public static void main(String[] args) {**

**System.out.println( );**

**System.out.printf("Um caractere: %c",'A');**

**System.out.println( );**

**System.out.printf("Um texto: %s","Java em todos os lugares");**

**System.out.println( );**

**System.out.printf("Um número inteiro: %d",123);**

**System.out.println( );**

**System.out.printf("Um número decimal: %f",4.55);**

**System.out.println( );**

**System.out.printf("Um valor booleano: %b",true);**

**System.out.println( );**

**System.out.println( );**

**}**

**}**

**Tipos de Dados em Java**

**O Java compreende 2 grupos de tipos de dados: Primitivos e Referência:**

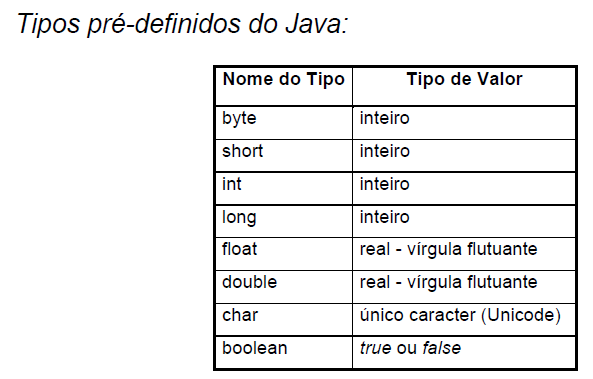
**- Tipos Primitivos:**

**boolean**

**char**

**byte, short, int, long**

**float, double**



**- Tipos de Referência:**

**Arrays**

**Classes**

**Interfaces**

**Obs.: Em Java, String não é um tipo de dado, mas sim uma classe. E suas variáveis são, na verdade, objetos dessa classe, por isso é digitada com a inicial maiúscula.**

**Concatenação**

**Concatenação nada mais é do que juntar strings numa só.**

**De uma forma mais simples, podemos usar o + para juntar várias strings ao mesmo tempo.**

**Trabalhando com Variáveis e Estruturas de Programação**

**- declaração, atribuição de valores, comparação de variáveis;**

**- controle de fluxo através de if e else;**

**- instruções de laço for e while, controle de fluxo com break e continue.**

**Variáveis**

**Uma variável é um espaço na memória usado para armazenar o estado de um objeto.**

**Uma variável deve ter um nome e um tipo. O tipo da variável indica o tipo de dado que ela pode conter. Os nomes das variáveis devem seguir as mesmas regras de nomenclatura que**

**os identificadores.**

**Nome da Variável**

**Nomes de variáveis são sensitivos a minúsculas e maiúsculas. Isso quer dizer que "cliente" é diferente de "Cliente". O nome de uma variável pode ser qualquer identificador legal: Uma seqüência sem limites de tamanho de letras e dígitos, começando com uma letra, "$" ou o caractere de sublinhado "\_".**

**A convenção, contudo, é sempre começar um nome de variável com uma letra e não "$" ou "\_". Além disso, por convenção, o caractere $ nunca é usado. Você pode encontrar situações em que nomes gerados automaticamente podem conter o sinal $, mas evite usá-lo sempre que possível. O mesmo acontece com "\_". Embora permitido, evite começar nomes de variáveis com "\_".**

**Espaços não são permitidos em nomes de variáveis.**

**Caracteres subsequentes podem ser letras, dígitos, $ ou "\_".**

**Quando escolher nomes para suas variáveis, use nomes completos em vez de abreviações confusas. Isso tornará seu código mais legível.**

**Nomes tais como "cliente", "nota", "quantidade", etc, são bem mais fáceis de entender e lembrar que "cl", "n", "qt", etc. Tenha também em mente que palavras-chaves e palavras reservadas da linguagem não podem ser usadas como nomes de variáveis.**

**Se o nome de variável que você escolher consistir de apenas uma palavra, escreva-a toda em letras minúsculas. Ex.: cliente. Se consistir de mais de uma palavra, escreva a primeira letra da segunda palavra usando letra maiúscula.**

**Exemplo: codCliente**

**Identificadores em Java**

**● Identificadores são representações de nomes de variáveis,**

**métodos, classes, etc**

**● Exemplos de identificadores podem ser: Hello, main, System, out**

**● Os identificadores são case-sensitive.**

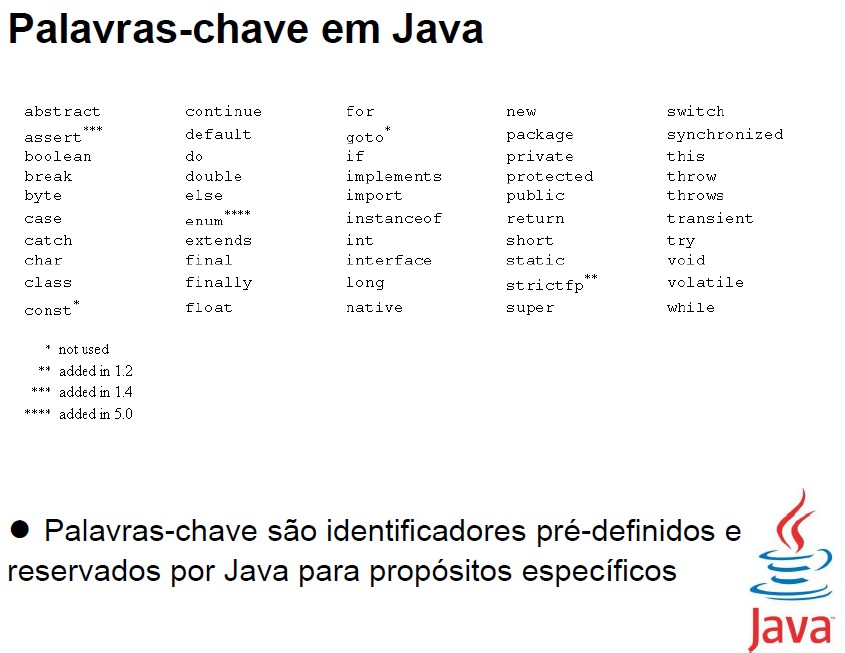
**- Isto significa que o identificador Hello não é o mesmo que hello.**

**● Iniciam com Letra (A-Z, a-z), Underscore “\_”, ou Sinal de**

**cifrão “$”.**

**● Aos caracteres subseqüentes adicionam números (0-9).**

**● Não pode utilizar nomes iguais às palavras-chave.**



**Declarando e inicializando Variáveis**

**Dentro de um bloco, podemos declarar variáveis e usá-las.**

**Em Java, toda variável tem um tipo que não pode ser mudado, uma vez que declarado:**

**tipoDaVariavel nomeDaVariavel;**

**Por exemplo:**

**int idade;**

**A linha a seguir é a tradução de “idade deve valer agora quinze”.**

**idade = 15;**

**Além de atribuir, você pode utilizar esse valor. O código a seguir declara novamente a variável idade com valor 15 e imprime seu valor na saída padrão através da chamada a System.out.println.**

**//declara a idade**

**int idade;**

**idade = 15;**

**// imprime a idade**

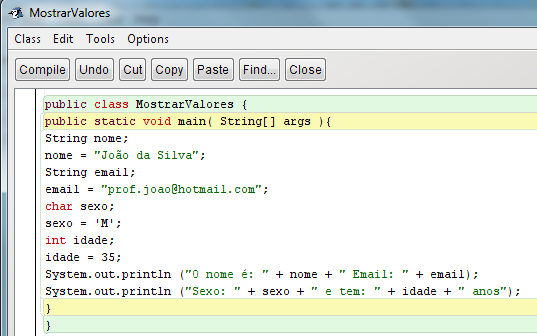
**System.out.println(idade);**

**Exercício 4: Desenvolva um projeto em Java para exibir suas informações pessoais, através de variáveis.**

**Teremos: nome, email, sexo, idade.**

**Projeto: Prj\_Variaveis**

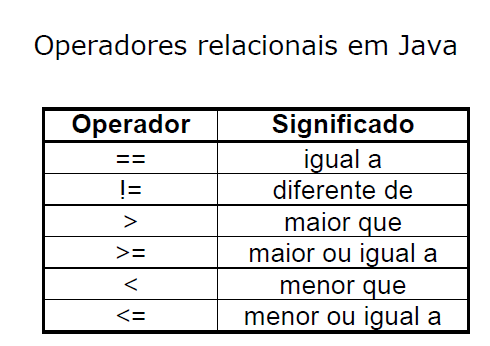
**Classe: Mostrar\_Valores**

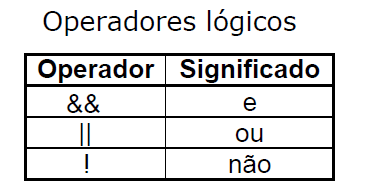
****

**Operadores em Java**

**Operadores Aritméticos**

|  |  |
| --- | --- |
| **Operador** | **Descrição** |
| **++** | **Incremento** |
| **--** | **decremento** |
| **+** | **Adição** |
| **-** | **Subtração** |
| **\*** | **Multiplicação** |
| **/** | **Divisão** |
| **%** | **Módulo (resto)** |

****

****

**Exercício 5: Desenvolva um projeto em Java para somar 4 números inteiros. O usuário entrará com 4 números e o resultado da soma desses números será apresentado na tela.**

**Nome do Projeto: Prj\_Somar\_Numeros**

**Nome da Classe: SomarNumeros**

**Recebendo dados do usuário**

**Para receber dados do usuário, temos que usar a classe Scanner, que faz parte do pacote 'java.util'.**

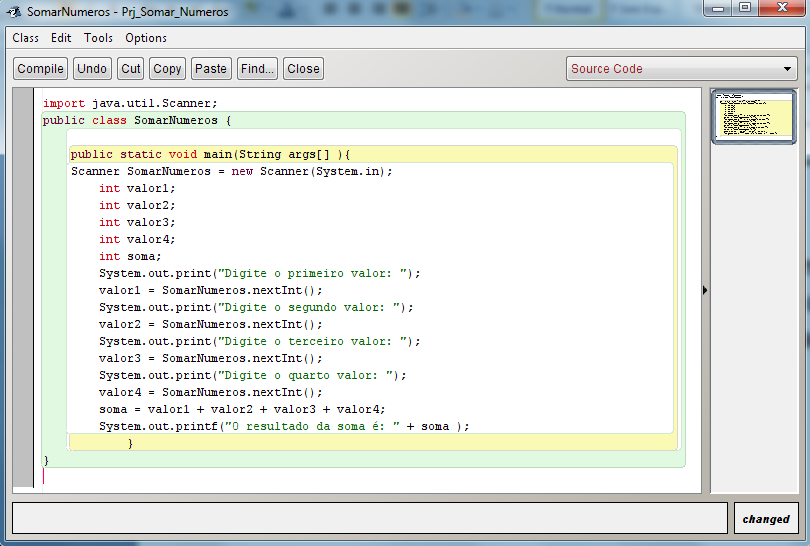
**Vamos dizer ao Java que usaremos essa classe na nossa aplicação.**

**Para isso, adicione essa linha no começo do programa:**

**import java.util.Scanner;**

**Bom, temos a classe. Vamos declarar o nosso objeto do tipo Scanner.  
Vamos chamá-lo de 'SomarNumeros'. Sua declaração é feita da seguinte maneira:**

**Scanner SomarNumeros = new Scanner(System.in);**

****