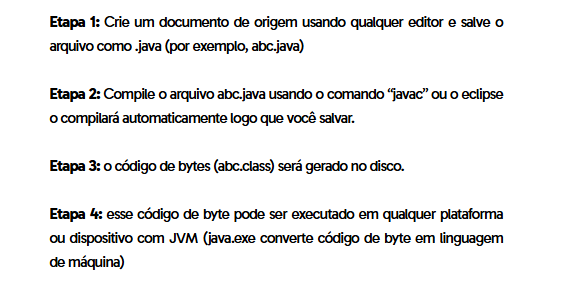
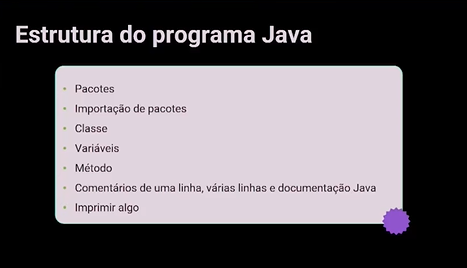
**Java**: Base para aprender qualquer linguagem, é a linguagem mais utilizada, multiplataforma (funciona em qualquer lugar).



**Estrutura da classe**:



* **Pacote:** seria o caminho do recurso (ex: .descomplica.programaçãoI.aula1).
* **Importação de Pacotes:** Importar recursos de outra pasta para a que ta usando.
* **Classe:** “Protótipo” de como o objeto vai ser. Delimitador do que vai falar.
* **Variáveis:** Repositório de memoria aonde consigo jogar valores.
* **Método:** São funções internas da classe.
* **Comentários:** Para deixar uma memoria do que significa o código.
* **Imprimir algo**

**Ex:**

public class Aula{

public static void main(String[] args){

System.out.println("Gabriel");

}

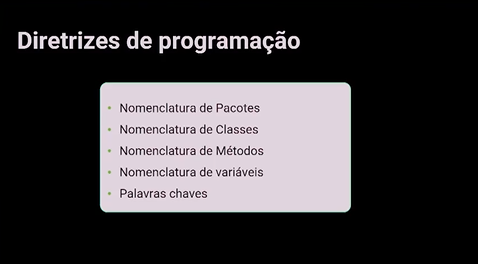
}

Public é aonde define os escopos

Class é palavra reservada

Aula é o nome do objeto (existe uma convenção, não pode ser qualquer jeito o nome)

**Diretrizes de Programação**:



* **Nomenclatura de Pacotes**: Convencionou-se que todos os pacotes devem ser escritos com letra minúscula. Ex:

package descomplica.aula;

A instrução package define um espaço para nome no qual as classes são armazenadas. O pacote é usado para **organizar as classes com base na funcionalidade**. Se você omitir a instrução do pacote, os nomes das classes serão colocados no pacote padrão, que não tem nome. A declaração do pacote não pode aparecer em nenhum lugar do programa. **Deve ser a primeira linha do seu programa ou você pode omiti-lo**.

* **Nomenclatura de Classes**:

public class AulaExemplo001{

}

public: Esta é a palavra-chave modificadora de acesso que informa o acesso do compilador à classe. Vários valores de modificadores de acesso podem ser públicos, protegidos, privados ou padrão (sem valor).

class: esta palavra-chave usada para declarar uma classe. Nome da classe (AulaExemplo001) seguido por esta palavra-chave.

* **Nomenclatura de método**: Dentro da Classe

public class AulaExemplo001{

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

}

}

Seu método (Função) chamado main com array de strings como argumento.

public: Modificador de acesso

static: static é uma palavra-chave reservada, o que significa que um método é acessível e utilizável, mesmo que não exista nenhum objeto da classe.

void: palavra-chave declara que nada seria retornado do método. O método pode retornar qualquer primitivo ou objeto.

Conteúdo do método dentro de chaves. {}

\**Se essa classe fosse algo que fizesse uma animação ou algo do tipo: a pessoa iria colocar:* ***import descomplica.aula.AulaExemplo001****\**

* **Nomenclatura de Variáveis**: Java é tipado, você precisa colocar o tipo das variáveis.

public class AulaExemplo001{

public static void main (String[] args){

int idadeProfessor;

idadeProfessor= 5;

}

}

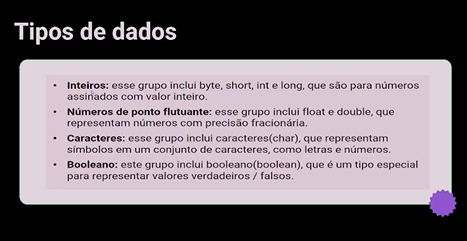
Note na variável a primeira letra é minúscula e apartir dela é maiúscula! isso é convenção! é recomendado usar sempre!

* **Mostrar os valores:**

System.out.println(idadeProfessor);

Nesse caso ele vai mostrar o conteúdo da variável.

**Tipos de Dados:**





public class Aula003{

public static void main(String[] args){

**int** idade = 18;

**float** peso = 90.1;

**double** peso2 = 90.2;

System.out.println(idade);

System.out.println(peso);

System.out.println(peso2);

}

}

● **byte:**

O menor tipo inteiro é byte. Tem um valor mínimo de -128 e um valor máximo de 127 (inclusive). O tipo de dados de bytes pode ser útil para economizar memória em matrizes grandes, onde a economia de memória realmente importa. As variáveis ​​de bytes são declaradas pelo uso da palavra-chave byte. EX:

byte b = 100;

● **short**:

O tipo de dados short é um inteiro de complemento de dois de 16 bits e assinado. Tem um valor mínimo de -32.768 e um valor máximo de 32.767 (inclusive). Assim como no byte, as mesmas diretrizes se aplicam: você pode usar um pequeno para economizar memória em matrizes grandes, em situações em que a economia de memória realmente importa. EX:

short s= 123;

● **int:**

O tipo **inteiro mais comumente usado é int**. É um tipo assinado de 32 bits que tem um intervalo de –2.147.483.648 a 2.147.483.647. Além de outros usos, variáveis do tipo int são comumente empregadas para controlar loops e indexar matrizes. Esse tipo de dados provavelmente será grande o suficiente para os números que seu programa usará, mas se você precisar de um intervalo maior de valores, use long.

int v = 123543;

int calc = -9876345;

● **long**:

Long é um tipo assinado de 64 bits e é útil para aquelas ocasiões em que um tipo **int não é grande o suficiente** para armazenar o valor desejado. Possui um valor mínimo de -9.223.372.036.854.775.808 e um valor máximo de 9.223.372.036.854.775.807 (inclusive). O uso desse tipo de dados pode estar no aplicativo bancário quando uma quantidade grande deve ser calculada e armazenada.

long longoVal = 1234567891;

● **float:**

Os números de ponto flutuante, também conhecidos como números reais, são usados ​​ao avaliar expressões que requerem precisão fracionária. Por exemplo, cálculo da taxa de juros ou cálculo da raiz quadrada. O tipo de dados flutuante é um ponto flutuante IEEE 754 de 32 bits e precisão única. Como nas recomendações para byte e short, use um **ponto flutuante (em vez de duplo) se precisar economizar memória em grandes matrizes de números de ponto flutuante**. O tipo float especifica um valor de precisão único que usa 32 bits de armazenamento. A precisão única é mais rápida em alguns processadores e ocupa metade do espaço da precisão dupla. EX:

float taxaJuros = 12.25f;

● **double**:

A precisão dupla, como indicado pela palavra-chave double, usa 64 bits para armazenar um valor. A precisão dupla é realmente mais rápida do que a precisão única em alguns processadores modernos que foram otimizados para cálculos matemáticos de alta velocidade. Todas as funções matemáticas transcendentais, como sin (), cos () e sqrt (), retornam valores duplos. EX:

double duplo = 12345.234d;

● **boolean**:

O tipo de dados booleano possui apenas dois valores possíveis: **verdadeiro e falso**. Use esse tipo de dados para sinalizadores simples que rastreiam condições verdadeiras / falsas. Este é o tipo retornado por todos os operadores relacionais, como no caso de a < b. booleano também é o tipo exigido pelas expressões condicionais que governam as instruções de controle, como if ou while.

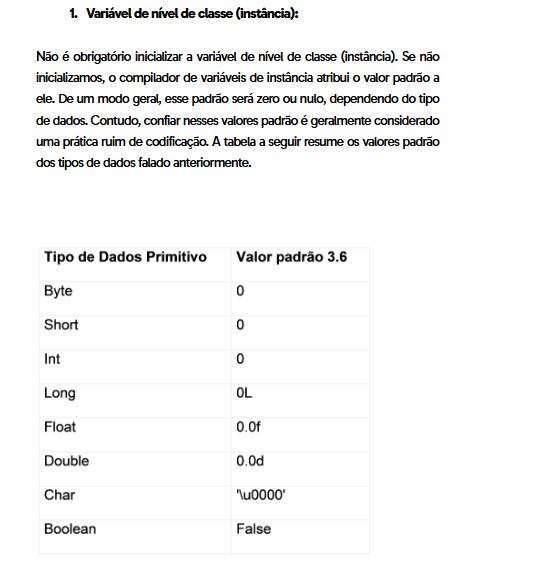
boolean val = false;

● **Caracteres:**

Em Java, o tipo de dados usado para armazenar **caracteres é char**. O tipo de dados char é um único caractere Unicode de 16 bits. Ele tem um valor mínimo de ‘\ u0000’ (ou 0) e um valor máximo de ‘\ uffff’ (ou 65.535 inclusive). Não há caracteres negativos.

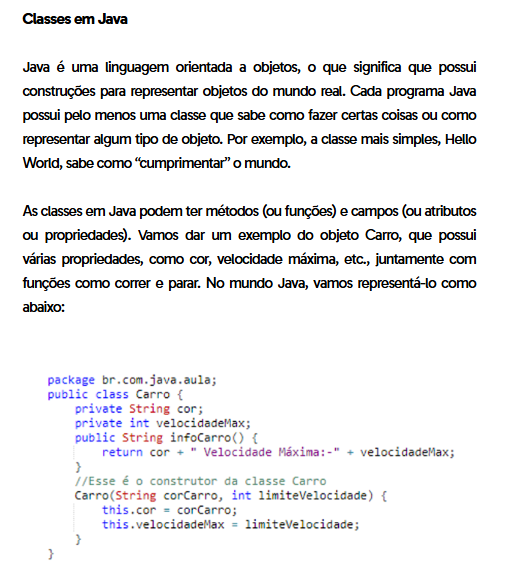
char ch1 = 88; // código para X

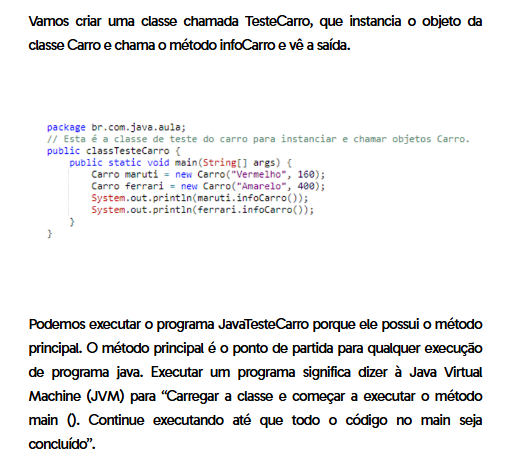
char ch2 = ‘Y’;



**Variável local do método**:

As variáveis ​​locais do método precisam ser inicializadas antes de usá-lo. O compilador nunca atribui um valor padrão a uma variável local não inicializada. Se você não conseguir inicializar sua variável local onde ela é declarada, certifique-se de atribuir um valor a ela antes de tentar usá-la. O acesso a uma variável local não inicializada resultará em um erro em tempo de compilação.





**Diretrizes de programação:**

Os identificadores Java devem começar com **uma letra**, **um caractere de moeda** ($) ou um **caractere de conexão**, como o sublinhado (\_). Identificadores não podem começar com um número. Após o primeiro caractere, os identificadores podem conter qualquer combinação de letras, caracteres monetários, caracteres de conexão ou números. Por exemplo:

int variável1 = 10; // Isso é válido

int 4var = 10; // isso é inválido, o identificador não pode começar com um dígito.

Identificadores, nomes de métodos e nomes de classes **diferenciam maiúsculas de minúsculas**; var e Var são dois identificadores diferentes. Você não pode usar palavras-chave Java como identificadores.

Para **classes e interfaces** a primeira **letra deve ser maiúscula** e**, se várias palavras estiverem vinculadas para formar o nome, a primeira letra das palavras internas deverá estar em maiúsculas** (um formato que às vezes é chamado de “camelCase”).

Para os **métodos** a **primeira letra deve estar em minúscula** e, em seguida, devem ser usadas camelCase. Veja os exemplos de nomes de métodos: getBalance; doCalculation; setCustomerName.

Para as **variáveis** o formato camelCase deve ser usado, **começando com uma letra minúscula.** É recomendado nomes curtos e significativos, o que nos parece bom. Alguns exemplos: buttonWidth; saldoDaConta; empName.

Para definir uma **constante** é criada uma variável ​​estática e final. As constantes devem ser **nomeadas usando letras maiúsculas com caracteres sublinhados como separadores**: MIN\_HEIGHT

**Pontos importantes** a serem observados, pode haver **apenas uma** classe pública por arquivo de código-fonte. Além disso, os comentários podem aparecer no início ou no final de qualquer linha no arquivo de código-fonte; eles são independentes de qualquer uma das regras de posicionamento discutidas aqui.

Outro ponto é que se houver uma **classe pública em um arquivo, o nome do arquivo deverá corresponder ao nome da classe pública**. Por exemplo, uma classe declarada como “public class Cachorro {}” deve estar em um arquivo de código-fonte chamado Cachorro.java.