Linguagem Java para aplicativo DeskTop

Curso prático de linguagem Java

Rogério Laureano Martins

Linguagem Java..... 9

História do Java 10
Features Java 10

Preparação para programar em Java.....11

1) Como definir o caminho temporário do JDK no Windows 12
2) Como definir o caminho permanente do JDK no Windows..... 13
Configurando o caminho Java no Linux OS 18

Exibição de mensagem e textos 18

Variáveis 30

Classe Scanner e System.in 36
Operações Numéricas 39
Operações com texto..... 58
Operações com data 63
Transformações de caracter para número 85
Transformações de data para número 90
Transformações de número para data 91
Transformação de String para data..... 93
Transformação de Data para String 94
Transformações de número para número 95
Transformação de String para boolean..... 99
Transformação de boolean para String..... 100
Transformação de String para char 100
Transformação de char para String 101
Transformação de Date para Timestamp..... 101
Transformação de Timestamp para Date..... 101
Transformações de String para Objeto..... 102
Transformações de Objeto para String..... 102
Teste de variáveis e tomadas de decisões 102
Ciclos..... 112
Array de variáveis 116

Array 117
Enumeration 117
Vector 118
ArrayList 120
LinkedList 120
LinkedHashSet 121
TreeMap 122
TreeSet 123
BitSet 123
Stack 124
HashMap 125
HashSet 126
Hashtable 127
Properties 128
Queue 129

Gravação e Leitura de arquivos 130

Gravação e leitura de arquivos texto 130

Comandos do sistema operacional 132

GUI – Interface gráfica com usuário (Java Swing) 133

Java JTextArea 164
Java JPasswordField 167
Java JCheckBox 169
Java JRadioButton 172
Java JComboBox 174
Java JTable 175
Java JList 177
Java JScrollBar 178
JMenu JMenuItem 180
Java JPopupMenu 183
Java JCheckBoxMenuItem 186
Java JSeparator 188
Java JProgressBar 191
Java JColorChooser 193
Java JTabbedPane 196

Java JSlider 199
Java JSpinner 200
Java JDialog 201
Java JFileChooser 203
Java JToggleButton 207
Java JToolBar 208
Java JViewport 210
Java JComponent 212
Java JLayeredPane 214
Java JDesktopPane 216
Java JEditorPane 218
Java JScrollPane 219
Java JSplitPane 221
Java JTextPane 222
Java JRootPane 223
Gerenciamento de Layouts 224
Border Layout 225
FlowLayout 226
CardLayout 239
GridBagLayout 240
SpringLayout 242
ScrollPaneLayout 243
GroupLayout 245
Usando ToolTip 245
Mudando o Ícone da barra de título 247
7 dicas para desenhar tela com Java Swing 248
Applet 248
Java JApplet 250

GUI – Interface gráfica com usuário (JavaFX) 254

JavaFX Text 254
JavaFX 2D Shapes 257
Line 257
Rectangle 259
Elipse 264
Arc 268

Circle 272
Polygons 274
Cubic Curve 276
Quad Curve 278
Color 280
Gradient Color 282
JavaFX Effects 286
ColorAdjust 286
ColorInput 287
ImageInput 289
Blend 290
Bloom 295
Glow 297
BoxBlur 299
GaussianBlur 301
MotionBlur 303
Reflection 304
SepiaTone 306
Shadow 308
DropShadow 310
JavaFX Transformation 311
Translation 312
Rotation 313
Scaling 315
Shearing 317
Multiple Transformations 320
JavaFX Animation 323
Rotate Transition 323
Scale Transition 325
Translate Transition 327
Fade Transition 328
Fill Transition 330
Stroke Transition 332
Sequential Transition 334
Parallel Transition 337
Pause Transition 340
Path Transition 343
JavaFX 3D Shapes 344
Box 345
Cylinder 347
Sphere 348

3D Object Properties 350
JavaFX UI Controls 356
Label 356
Button 357
RadioButton 360
CheckBox 361
TextField 363
PasswordField 365
FileChooser 367
Table 369
Menu 373
JavaFX Layouts 374
BorderPane 374
HBox 376
VBox 377
StackPane 378
GridPane 379
FlowPane 380
JavaFX Charts 381
Pie Chart 381
Line Chart 383
Area Chart 385
Bar Chart 388
JavaFX CSS 390

Banco de dados 394

Leitura de registros de dados 394
Gravação de registros de dados 395
Conexão com outros bancos de dados 396
Métodos da interface Connection 398
Métodos da classe Statement 398
Métodos da classe ResultSet 399
RowSet 399
Métodos da classe PreparedStatement 401
ResultSetMetaData 402
DataBaseMetaData 403
CallableStatement 404
Controle de transações no banco 405

Processamento batch 406
Conclusão 407

Minha gratidão a Deus por me permitir compartilhar o que tenho aprendido. Minha gratidão à minha esposa Beatriz e ao meu filho João por terem paciência. Minha gratidão aos professores e youtubers que de alguma forma procura passar o conhecimento pessoal às pessoas.

Rogério Laureano Martins

Linguagem Java

A linguagem Java é uma poderosa ferramenta de programação. Neste livro temos temos diversos conceitos desta linguagem.

Com o Java, o desenvolvimento de 4 tipos de programas são bastante populares.

Standalone → Programas populares para rodar em desktop. Semelhantes a Media Player, antivírus etc.

Web → Páginas HTML dinâmicas

Aplicativos corporativos → Aplicações para empresas como bancos, varejo, indústria etc.

Mobile → Aplicativos para equipamentos mobile como telefone celular, tablets etc.

Para tanto, forma criadas 4 plataformas de desenvolvimento para o java:

Java SE → Plataforma de desenvolvimento java que inclui todos os recursos de IO, Expressões regulares etc.

Java EE → Plataforma para desenvolvimentos corporativos e Web. É um pedaço importante do Java SE.

Java ME → Plataforma "micro" do Java. Utilizada principalmente para Mobile.

Java FX → Plataforma especial utilizada principalmente para aplicativos Web mais ricos.

História do Java

Em 1991 o projeto Java se iniciava com uma equipe formada por James Gosling, Mike Sheridan e Patrick Naughton. O time de desenvolvimento se denominou Green Team. Por isso o primeiro nome foi Greentalk. Posteriormente a linguagem mudou de nome para Oak, que era uma árvore típica dos Estados Unidos, França, Romênia etc. Como já existia uma empresa chamada Oak, mudou-se o nome para Java. Este nome foi dado em homenagem a ilha de Java, que tem uma história com a produção de café. Por isso também que o símbolo do java é uma chícara de café.

Features Java

O projeto Java, foi feito com o intuito de criar uma linguagem portável, segura e simples. Abaixo a lista dos principais features:

- Programa Simples → Os desenvolvedores pretenderam tirar a dificuldade da linguagem. É muito semelhante a linguagem C. Porém, sem as complicações de ponteiros. Este, por exemplo, podem ser usados. Porém, são internos no Java.

- Orientação por objeto → O Java é uma linguagem orientada por objetos. Possibilita a criação de Objetos, Classes, Heranças, Polimorfismo e todos os conceitos de orientação por objetos.

- Portável → O bytecode gerado é o mesmo em qualquer interpretador Java

- Multiplataforma → Por rodar em uma máquina virtual, "JVM" é portátil e roda em diversos sistemas operacionais.

- Segura → Por estar na JVM, tem mais uma camada de segurança independente do sistema operacional

- Robusta → Ferramente robusta no desenvolvimento

- Archetecture neutral → Por exemplo, os tamanhos dos tipos primitivos são reservados sempre o mesmo tamanho de bits.

- Linguagem interpretada → Ao rodar o aplicativo criado em Java, é necessário que se cria um bytecode. Ou seja, é feito uma "compilação" no Java para criar este bytecode. Após isso o Java interpreta o bytecode e executa o programa.

- Multithreaded → O Java permite programas com diversas threads em paralelo

- Distribuída → Facilmente distribuido

- Dinâmica → Linguagem bastante dinâmica e versátil

Preparação para programar em Java

Para executar qualquer programa java, você precisa

- Instale o JDK se você não o tiver instalado, faça o download do JDK e instale-o.
- Defina o caminho do diretório jdk
- Crie o programa java

- Compile e execute o programa java

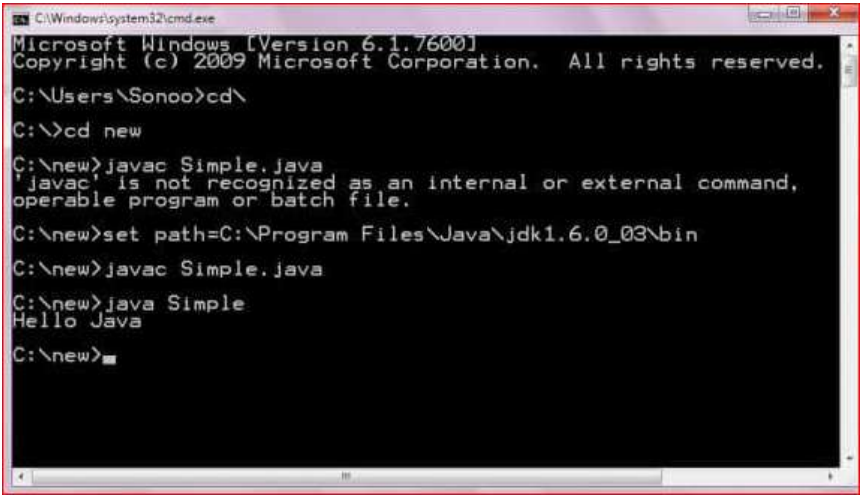
É necessário definir o caminho para o uso de ferramentas como javac, java, etc.

Se você estiver salvando o arquivo de origem Java dentro do diretório JDK / bin, não será necessário definir o caminho, pois todas as ferramentas estarão disponíveis no diretório atual.

No entanto, se você tiver seu arquivo Java fora da pasta JDK / bin, será necessário definir o caminho do JDK.

Existem duas maneiras de definir o caminho em Java:
1 – Temporário
2 – Permanente

1) Como definir o caminho temporário do JDK no Windows
Para definir o caminho temporário do JDK, você precisa seguir as seguintes etapas:
Abra o prompt de comando
Copie o caminho do diretório JDK / bin
Escreva no prompt de comando: defina path = copied_path
Por exemplo:
definir caminho = C: \ Arquivos de Programas \ Java \ jdk1.6.0_23 \ bin
Vamos ver na figura abaixo:



2) Como definir o caminho permanente do JDK no Windows

Para definir o caminho permanente do JDK, siga estas etapas:

Vá para Propriedades do MyComputer -> guia avançado -> variáveis de ambiente -> nova guia da variável do usuário -> caminho de gravação no nome da variável -> caminho de gravação da pasta bin no valor da variável -> ok -> ok -> ok

Por exemplo:

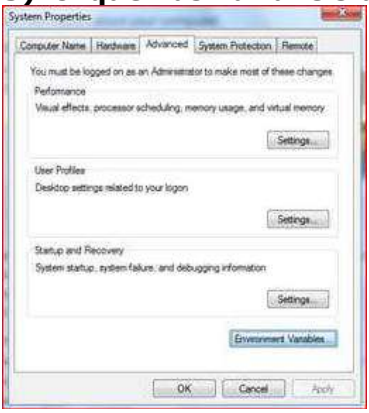
1) Vá para propriedades do MyComputer



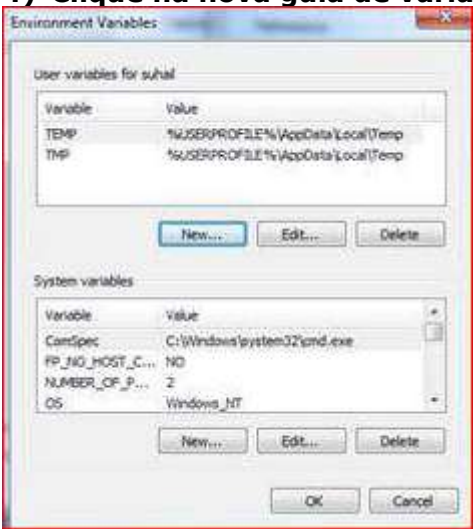
2) Clique na aba avançada



3) Clique nas variáveis de ambiente



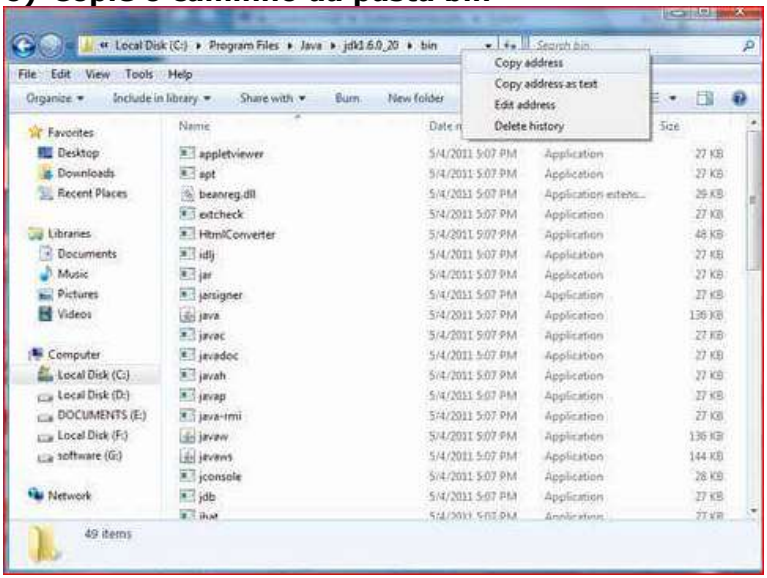
4) Clique na nova guia de variáveis do usuário



5) Escreva o caminho no nome da variável



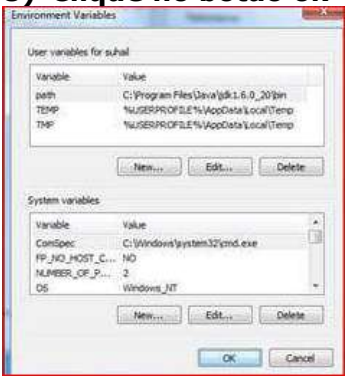
6) Copie o caminho da pasta bin



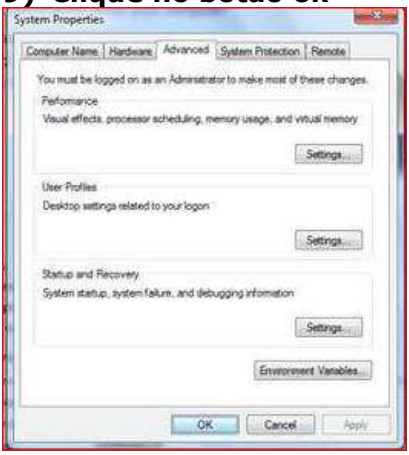
7) Cole o caminho da pasta bin no valor da variável



8) Clique no botão ok



9) Clique no botão ok



Agora seu caminho permanente está definido. Agora você pode executar qualquer programa de java a partir de qualquer unidade.

Configurando o caminho Java no Linux OS

Definir caminho no sistema operacional Linux é o mesmo que definir o caminho no sistema operacional Windows. Mas, aqui usamos a ferramenta de exportação em vez de definir. Vamos ver como definir o caminho no sistema operacional Linux:

```
exportar PATH = $ PATH: /home/jdk1.6.01/bin/
```

Aqui, instalamos o JDK no diretório inicial em Raiz (/ home).

Exibição de mensagem e textos

O programa mais simples para qualquer linguagem é aquele que tem como objetivo retornar uma mensagem ao usuário.
Assim, para iniciarmos o estudo da linguagem Java, siga as instruções:

1 – Abra o programa Notepad
2 – Digite o seguinte fonte:

```
public class mensagem1
{
  public static void main(String[] args)
  {
    System.out.print("Olá Mundo");
  }
}
```

3 – Salve o arquivo como "C:\testeJava\mensagem1.java"
4 – Abra o programa cmd.exe

5 – Digite o comando “cd \ testeJava”
6 – Digite o comando “javac mensagem1.java”
7 – Digite o comando “java mensagem1”
8 – O sistema mostrará a seguinte tela:



Uma segunda maneira é utilizando a biblioteca iostream. Segue exemplo

```
public class mensagem2
{
  public static void main(String[] args)
  {
    System.out.println("Olá Mundo");
  }
}
```

Neste caso a diferença é que acrescentará uma linha no final do “Olá Mundo”

Outra maneira de ter este resultado, porém com mais recursos

1 – Abra o programa Notepad
2 – Digite o seguinte fonte:

```
public class mensagem3
{
  public static void main(String[] args)
  {
    System.out.printf("Olá Mundo");
  }
}
```

3 – Salve o arquivo como “C:\testeJava\mensagem1.java”

4 – Abra o programa cmd.exe
5 – Digite o comando “cd \ testeJava”
6 – Digite o comando “javac mensagem1.java”
7 – Digite o comando “java mensagem1”
8 – O sistema mostrará a seguinte tela:

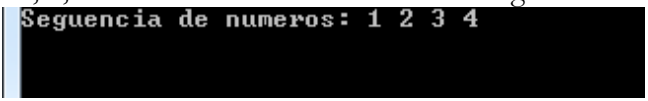


A função printf da linguagem java possibilita a exibição de mensagem. Tem a seguinte sintaxe:
printf(<caracteres> [, <valor 1>, <valor 2> ....])

O primeiro parâmetro é o texto a ser escrito. Os demais parâmetros são valores separados por “,” que permitem substituir as expressões de controle. Exemplo:

```
public class mensagem4
{
  public static void main(String[] args)
  {
 System.out.printf("Seguencia de numeros: %d %d %d %d",1,2,3,4);
  }
}
```

No fonte acima, o texto “Sequencia de números: ” tem 4 expressões de controle “%d”. Em seguida com os demais parâmetros que são “1, 2, 3 e 4”. Assim o sistema exibirá a seguinte mensagem:



Abaixo a lista dos tipos de expressões de controle.

Expressão de controle	Funcionalidade
%d	Permite que seja efetuada a escrita de números

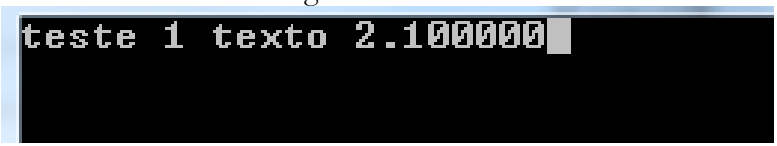
Expressão de controle	Funcionalidade
	inteiros
%e	Permite que seja efetuada a escrita de números em notação científica. Tem a opção de limitar a quantidade de dígitos após o ponto fluante. Se por exemplo for informado "%.2e" aparecerá a notação científica com dois dígitos após o ponto flutuante
%E	Igual ao %e. Porém com letras maiúsculas
%f	Permite que seja efetuada a escrita de números reais (números fracionados). Tem a opção de limitar a quantidade de dígitos após o ponto fluante. Se por exemplo for informado "%.2f" aparecerá a o número fracionado com dois dígitos após o ponto flutuante
%g	Permite que seja efetuada a escrita de números fracionados no formato mais curto. No caso da notação "%.2g" ou "%.3g" sempre exibirá o valor fracionado arredondado em 1 casa decimal a menos que a informada. Por exemplo: se o valor 2,1234 for colocado como "%.2g" retornará "2.1" se for colocado como "%.3g" retornará "2.12"
%G	Igual ao %g. Porém com letras maiúsculas
%o	Permite que seja efetuada a escrita de números octais
%s	Permite que seja efetuada a escrita de uma série de caracteres
%x	Permite que seja efetuada a escrita de um número hexadecimal
%X	Igual ao %x. Porém com letras maiúsculas
%p	Ponteiro em hexadecimal
%%	Imprime % no texto

Exemplo de utilização de um texto com várias expressões:

```
public class mensagem5
{
  public static void main(String[] args)
  {
    System.out.printf("teste %d %s %f", 1, "texto", 2.1);
```

```
   }
}
```

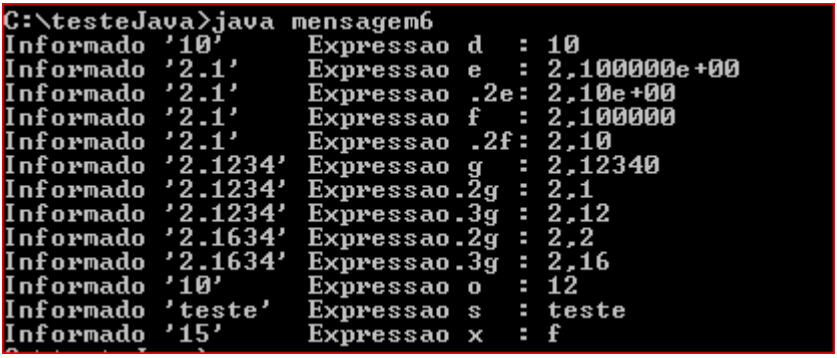
O sistema retornará o seguinte:



Exemplo de utilização das expressões de controle:

```
public class mensagem6
{
  public static void main(String[] args)
  {
  System.out.printf("Informado '10'     Expressao d  : %d \n",
                    10);
  System.out.printf("Informado '2.1'    Expressao e  : %e \n",
                    2.1);
  System.out.printf("Informado '2.1'    Expressao .2e: %.2e \n",
                    2.1);
  System.out.printf("Informado '2.1'    Expressao f  : %f \n",
                    2.1);
  System.out.printf("Informado '2.1'    Expressao .2f: %.2f \n",
                    2.1);
  System.out.printf("Informado '2.1234' Expressao g  : %g \n",
                    2.1234);
  System.out.printf("Informado '2.1234' Expressao.2g : %.2g \n",
                    2.1234);
  System.out.printf("Informado '2.1234' Expressao.3g : %.3g \n",
                    2.1234);
  System.out.printf("Informado '2.1634' Expressao.2g : %.2g \n",
                    2.1634);
  System.out.printf("Informado '2.1634' Expressao.3g : %.3g \n",
                    2.1634);
  System.out.printf("Informado '10'     Expressao o  : %o \n",
                    10);
  System.out.printf("Informado 'teste'  Expressao s  : %s \n",
                    "teste");
  System.out.printf("Informado '15'     Expressao x  : %x", 15);
  }
}
```

O sistema exibirá:



Note que neste último script, além de ter as expressões de controle, tem também os caracteres "\n" ao final do primeiro parâmetro. Este é o caracter especial que representa a quebra de linha.

Abaixo a listagem dos caracteres especiais da função printf.

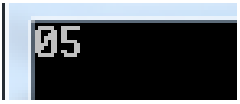
Caracter especial	**Funcionalidade**
\n	Gera uma nova linha
\t	Gera um espaço de tabulação.
\b	Executa um retrocesso de espaço
\"	Adiciona o símbolo aspa
\\	Adiciona o símbolo "\"
\f	No caso de impressora, gera uma quebra de página
\0	Adiciona nulo

Se notar, também tem as expressões semelhantes a "%.2f". Ou seja ainda temos outras possibilidades para exibição da mensagem. Na verdade a expressão de controle é composta da seguinte maneira:
%[opção][largura][.precisão][tamanho]tipo.
Os tipos já foram mencionados acima e são obrigatório. Exemplo: %s, %d, %f.

As opções são as seguintes:
0 (zero) – Permite, em combinação com a largura, definir zeros à esquerda de valores numéricos.

```
public class mensagem7
{
  public static void main(String[] args)
  {
  int teste = 5;
  System.out.printf("%02d",teste);
  }
}
```

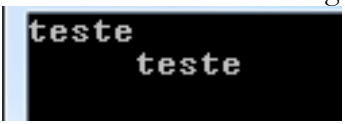
O sistema exibe a mensagem:



A opção "-" permite alterar o alinhamento conforme o tamanho do campo:

```
public class mensagem8
{
  public static void main(String[] args)
  {
  String teste = "teste";
  System.out.printf("%-10s\n",teste);
  System.out.printf("%10s",teste);
  }
}
```

O sistema exibirá a mensagem:



Ou seja, o primeiro alinhou pela esquerda (invertendo o alinhamento padrão) e o segundo pela direita (mantedo o alinhamento padrão)
A opção " " (espaço) reserva um caracter para sinal de negativo para números menores que zero.

```
public class mensagem9
{
  public static void main(String[] args)
  {
  double numero = 1.2345;
  printf("% f\n",numero);
  numero = -1.2345;
  printf("% f",numero);
  }
}
```

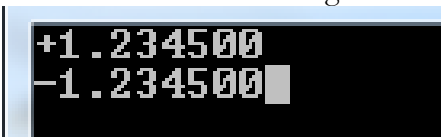
O sistema exibirá a mensagem:



A opção "+" coloca o sinal dos números positivos "+" e negativos "-"

```
public class mensagem10
{
  public static void main(String[] args)
  {
  double numero = 1.2345;
  printf("%+f\n",numero);
  numero = -1.2345;
  printf("%+f",numero);
  }
}
```

O sistema exibirá a mensagem:

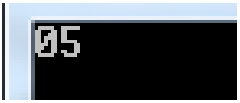


A opção (') permite formatar números com pontos de milhar, milhão, bilhão etc. Porém, funciona no UNIX.

O parâmetro de opções são estes citados acima. O segundo parâmetro possível é o de largura. Já vimos nos exemplos das opções "0" e "-" seu funcionamento. A largura permite limitar a exibição pela quantidade.

```
public class mensagem11
{
  public static void main(String[] args)
  {
  int teste = 5;
  printf("%02d",teste);
  }
}
```

O sistema exibe a mensagem:

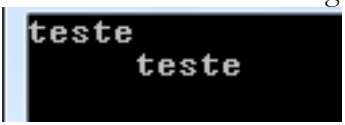


Neste exemplo foi definido uma largura de "2" completando com "0" à esquerda.

```
public class mensagem12
{
  public static void main(String[] args)
  {
```

```
  String teste = "teste";
  System.out.printf("%-10s\n",teste);
  System.out.printf("%10s",teste);
  }
}
```

O sistema exibirá a mensagem:



Neste exemplo foi definido uma largura de "10" com alinhamento invertido para primeira linha "teste ". Na segunda linha uma largura de 10 com alinhamento padrão " teste".

O terceiro parâmetro é a precisão. Para valores numéricos fracionados, é limitada a quantidade de dígitos após o ponto fracionado.

```
public class mensagem13
{
  public static void main(String[] args)
  {
  double numero = 1.2345;
  printf("%f\n",numero);
  printf("%.2f",numero);
  }
}
```

O sistema exibirá a mensagem

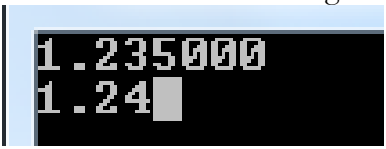


No exemplo acima, o sistema exibiu o valor arredondando para baixo. Ou seja, com valor (1,23).

Caso a fração que será omitida seja maior ou igual a "5" o sistema arredonda para cima.

```
public class mensagem14
{
  public static void main(String[] args)
  {
  double numero = 1.235;
  printf("%f\n",numero);
  printf("%.2f",numero);
  }
}
```

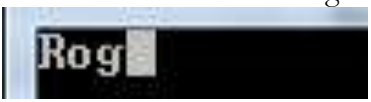
O sistema exibirá a mensagem



A precisão para valores caracters faz a extração dos primeiros caracteres

```
public class mensagem15
{
  public static void main(String[] args)
  {
  String texto = "Rogerio";
  System.out.printf("%.3s",texto);
  }
}
```

O sistema exibirá a mensagem



A seguir entrada de dados

1 – Crie um novo fonte em "Java" (utilizar os passos de 1 à 4 descritos no exercício acima)

2 – Digitar o seguinte fonte:

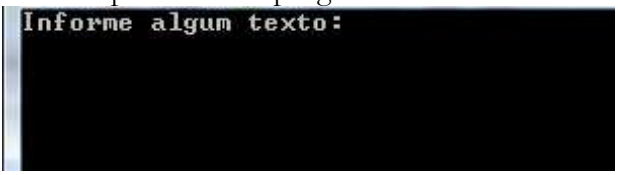
```
import java.util.Scanner;

public class testeEntrada {
  public static void main (String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Informe algum texto: ");

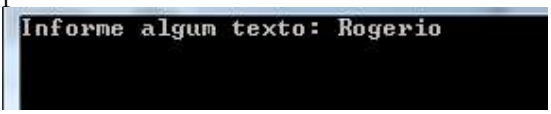
    String teste = entrada.nextLine();

    System.out.println("Foi informado (" + teste + ")");
  }
}
```

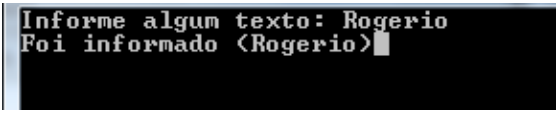
3 – Compile e rode o programa



4 – Informe um valor qualquer. No exemplo abaixo "Rogerio" e pressione ENTER



5 – O sistema exibirá a seguinte tela:



Variáveis

Como no exemplo acima da classe Scanner atribui as informações que o usuário inseriu em variáveis.

Quanto ao tipo de informação primitivos que as variáveis suportam, destaco:

byte → Número inteiro de -128 à 127
short → Número inteiro de -32.768 à 32.767
int → Número inteiro de -2.147.483.648 até 2.147.483.647
long → Número inteiro de -9.223.372.036.854.770.000 à -9.223.372.036.854.770.000
float → Número real (com valores fracionados) de $3,4^{-38}$ à $3,4^{38}$
double → Número real de $1,7^{-308}$ à $1,7^{308}$
char → Caracter que vai de 0 à 65535
boolean → Valor verdadeiro e falso.

O java é case sensitive. Ou seja, faz diferença entre letras maiúsculas e minúsculas no momento da codificação. Abaixo temos a lista de classes básicas do java e a referência com as variáveis primitivas:

Byte → Classe que contem regras, transformações dos valores primitivos do tipo byte.
Short → Classe que contem regras, transformações dos valores primitivos do tipo short.

Integer → Classe que contem regras, transformações dos valores primitivos do tipo int.
Long → Classe que contem regras, transformações dos valores primitivos do tipo long.
Float → Classe que contem regras, transformações dos valores primitivos do tipo float.
Double → Classe que contem regras, transformações dos valores primitivos do tipo double.
String → Classe que contem regras, transformações dos valores primitivos do tipo char.

Quanto ao tipo de declaração das variáveis, o Java pode ser de três tipos:

```
class A{
  int vIntancia = 50; //Variável de instância
  static int m = 100; //Variável estática
  void method(){
    int n=90; //Variável local
  }
}
```

Variável de instância é o tipo de variável que é declarada com não estática a nível da classe. Seu valor não é compartilhado para outras instâncias da mesma classe

Variável estática é o tipo de variável com a cláusula "static". Seu valor é compartilhado para outras instâncias da mesma classe. Não pode ser declarada como local.

Variável local é o tipo de variável declarada dentro de um método da classe.

Quanto aos acessos, temos 4 tipos:

Access modifiers	Dentro da classe	Dentro da package	Fora da package em uma subclasse	Fora da package
private	Permitido	Não perm	Não perm	Não perm
default	Permitido	Permitido	Não perm	Não perm
protected	Permitido	Permitido	Permitido	Não perm
public	Permitido	Permitido	Permitido	Permitido

private → Apenas a classe.
default → Na classe e na package em que esta classe pertence.
protected →Além de permitir na classe e na package também permite em uma sub-classe fora da package
public → Acesso irrestrito à variável ou método.

OBS: O conceito de package será melhor entendido mais à frente no livro.

Além do static, temos outras cláusulas que não são modificadores de acesso.

static → Valor que pode ser utilizado em outras instâncias
abstract → Declara variáveis abstratas
synchronized → Utilizado em threads
volatile → Utilizado em threads
final → Utilizado para finalizar implementações de classes, métodos e variáveis

No caso de inteiros, podemos fazer na entrada das variáveis a notação decimal, binária e hexadecimal. Na saída decimal, octal e hexadecimal. Esta funcionalidade é bastante útil na linguagem java principalmente para programação de microcontroladores. Segue os exemplos:

Exemplo com variáveis do tipo int com escrita em decimal saída em decimal

```
public class variaveis1 {

  public static void main(String[] arg) {
    int A = 15;
    System.out.printf("%d", A);
  }
}
```

Exemplo com variável do tipo int com escrita em binário saída em decimal

```
public class variaveis2 {

  public static void main(String[] arg) {
    int A = 0b1111;
    System.out.printf("%d", A);
  }
}
```

Exemplo com variável do tipo int com escrita em hexadecimal saída em decimal

```
public class variaveis3 {

  public static void main(String[] arg) {
    int A = 0xf;
    System.out.printf("%d", A);
  }
```

```
}
```

Exemplo com variáveis do tipo int com escrita em decimal saída em hexadecimal

```
public class variaveis4 {

  public static void main(String[] arg) {
    int A = 15;
    System.out.printf("%x", A);
  }
}
```

Exemplo com variável do tipo int com escrita em binário saída em hexadecimal

```
public class variaveis5 {

  public static void main(String[] arg) {
    int A = 0b1111;
    System.out.printf("%x", A);
  }
}
```

Exemplo com variável do tipo int com escrita em hexadecimal saída em hexadecimal

```
public class variaveis6 {

  public static void main(String[] arg) {
    int A = 0xf;
    System.out.printf("%x", A);
  }
}
```

Exemplo com variáveis do tipo int com escrita em decimal saída em octal

```
public class variaveis7 {

  public static void main(String[] arg) {
    int A = 15;
    System.out.printf("%o", A);
  }
}
```

Exemplo com variável do tipo int com escrita em binário saída em octal

```
public class variaveis8 {

  public static void main(String[] arg) {
    int A = 0b1111;
    System.out.printf("%o", A);
  }
}
```

Exemplo com variável do tipo int com escrita em hexadecimal saída em octal

```
public class variaveis9 {

  public static void main(String[] arg) {
    int A = 0xf;
    System.out.printf("%o", A);
  }
}
```

Exemplo com variável

```
public class variaveis10 {

  public static void main(String[] arg) {
    byte A = 15;
    System.out.printf("%d", A);
  }
```

```
}
```

Exemplo com variável short

```
public class variaveis11 {

  public static void main(String[] arg) {
    short A = 15;
    System.out.printf("%o", A);
  }
}
```

Exemplo com variável long

```
public class variaveis12 {

  public static void main(String[] arg) {
    long A = 15;
    System.out.printf("%x", A);
  }
}
```

Classe Scanner e System.in

No exemplo acima "testeEntrada.java", foi apresentado uma utilização da classe Scanner e do System.ini. As entradas de dados utilizadas desta forma, permite a entrada de informações conforme os tipos de variáveis.

Para o tipo String segue o exemplo:

```
import java.util.Scanner;
public class entrada1
{
  public static void main(String[] args)
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    String teste = sc1.nextLine();
```

```
    System.out.println(teste);
  }
}
```

Para o tipo int segue o exemplo:

```
import java.util.Scanner;
public class entrada2
{
  public static void main(String[] args)
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    int teste = sc1.nextInt();
    System.out.println(teste);
  }
}
```

Para o tipo byte segue exemplo:

```
import java.util.Scanner;
public class entrada3
{
  public static void main(String[] args)
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    byte teste = sc1.nextByte();
    System.out.println(teste);
  }
}
```

Para o tipo long segue exemplo:

```
import java.util.Scanner;
public class entrada4
{
  public static void main(String[] args)
```

```
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    long teste = sc1.nextLong();
    System.out.println(teste);
  }
}
```

Para o tipo float segue exemplo:

```
import java.util.Scanner;
public class entrada5
{
  public static void main(String[] args)
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    float teste = sc1.nextFloat();
    System.out.println(teste);
  }
}
```

Para o tipo double segue exemplo:

```
import java.util.Scanner;
public class entrada6
{
  public static void main(String[] args)
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    double teste = sc1.nextDouble();
    System.out.println(teste);
  }
}
```

Para o tipo boolean segue exemplo:

```
import java.util.Scanner;
```

```
public class entrada7
{
  public static void main(String[] args)
  {
    Scanner sc1 = new Scanner(System.in);
    System.out.println("Informe algo");
    boolean teste = sc1.nextBoolean();
    System.out.println(teste);
  }
}
```

Operações Numéricas

Segue tabela com os sinais e funções das expressões numéricas.

Sinal	Funcionalidade
+	Possibilita a soma entre dois números
-	Possibilita a subtração entre dois números
*	Possibilita a multiplicação entre dois números
/	Possibilita a divisão entre dois números
%	Possibilita a recuperar o resto de uma divisão.
++[var]	Pré-soma. Onde [var] é a variável
[var]++	Pós-soma. Onde [var] é a variável
--[var]	Pré-subtração. Onde [var] é a variável
[var]--	Pós-subtração. Onde [var] é a variável
[var]+=[val]	Soma implícita. Onde [var] é a variável e [val] o valor
[var]-=[val]	Subtração implícita. Onde [var] é a variável e [val] o valor
[var]*=[val]	Multiplicação implícita. Onde [var] é a variável e [val] o valor
[var]/=[val]	Divisão implícita. Onde [var] é a variável e [val] o valor
~	Subtrai 1 e inverte sinal. Exemplos: (10 vira -11) e (-10 vira 9)
>>	Regressão geométrica
<<	Progressão geométrica
pow	Possibilita a potenciação entre dois números

Sinal	Funcionalidade
sqrt	Possibilita a raíz entre dois números
cbrt	Possibilita a raíz cúbica
()	Agrupador de expressão numérica
abs	Permite arredondar em valor absoluto
round	Permite arredondar para o valor fracionado mais próximo
ceil	Permite arredondar para cima os números fracionados
floor	Permite arredondar para baixo os números fracionados
abs	Retorna o valor absoluto positivo de um número
acos	Função retorna o arco cosseno do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
asin	Função retorna o arco seno do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
atan	Função retorna o arco tangente do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
atan2	Função que retorna tangente entre dois números
cos	Função retorna o cosseno de f em radianos. O valor de retorno é de -1 a 1.
cosh	Função retorna o cosseno hiperbólico de x, definido matematicamente como (ex + e-x) / 2. Se o valor de x for muito grande (se ocorrer um estouro), a função falhará.
exp	A função retorna o valor de e - a base dos logaritmos naturais - aumentada para a potência x (ou seja, ex).
sin	Função retorna o seno de f em radianos. O valor de retorno é de -1 a 1.
sinh	Função retorna o cosseno hiperbólico de x, definido matematicamente como (ex + e-x) / 2. Se o valor de x for muito grande (se ocorrer um estouro), a função falhará.