Gabriel Campos,Luigi Soares,Vinicius Carvalho,Matheus Possas

# Linguagem de Programação JAVA

Belo Horizonte - Brasil

2018

Gabriel Campos,Luigi Soares,Vinicius Carvalho,Matheus Possas

# Linguagem de Programação JAVA

Trabalho Teórico Prático apresentado na disciplina de Linguagens de Programação do curso de Ciência da Computação da Pontifícia Universidade Católica de Minas Gerais sobre a linguagem de programação JAVA.

Belo Horizonte - Brasil

2018

# Lista de ilustrações

# Sumário

# 1 Introdução

Java é uma linguagem de programação interpretada de alto nível orientada a objetos e plataforma de software. 9 milhões de desenvolvedores criaram aplicações Java em várias indústrias de ponta. Diferente das linguagens de programação convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um código binário que é interpretado por uma máquina virtual. Os dois principais componentes da plataforma Java são a Java Application Programming Interface (API), que é a biblioteca de linhas de comando do Java, e a Java Virtual Machine (JVM), que interpreta o código Java em linguagem de máquina.

# 2 Histórico

Em 1991, o Green Project foi iniciado, na Sun Microsystems, o início do Java, uma linguagem de programação orientada a objetos. Os líderes do projeto eram Patrick Naughton, Mike Sheridan, e James Gosling. O projeto tinha como objetivo antecipar e planejar o próximo passo do mundo digital, e não a criação de uma nova linguagem de programação.

Eles acreditavam que em algum tempo haveria uma maior integração dos computadores com os equipamentos e eletrodomésticos bastante utilizados pelas pessoas em casa e no trabalho durante o cotidiano. Para validar a viabilidade desta idéia, treze pessoas trabalharam arduamente durante 18 meses. No segundo semestre de 1992, a equipe fez emergir de um escritório de Sand Hill Road, no Menlo Park, uma demonstração funcional da idéia inicial.

O protótipo se chamava *7 (leia-se “StarSeven”, ou “Estrela Sete” em português), um controle remoto com uma interface gráfica utilizando a tecnologia de tela sensível ao toque. Para o *7 foi criado um mascote, hoje amplamente conhecido no mundo Java, o Duke.

Figura 1 – Mascote Java


Disponivel em: <http://sahet.net/src/software/Java-duke.jpg>; Acesso em abr. 2018.

Duke no *7 era ser um guia virtual, ajudando e ensinando o usuário a utilizar o equipamento. O *7 tinha a capacidade de controlar diversos dispositivos e aplicações. James Gosling especificou uma nova linguagem de programação para o *7. Gosling decidiu batizá-la de “Oak”, que quer dizer “carvalho”, uma árvore que ele podia observar quando olhava pela sua janela.

O próximo passo era encontrar um mercado para o *7. A equipe achava que uma boa idéia seria controlar televisores e vídeo por demanda com o equipamento. Eles construíram um demo chamado MovieWood, mas infelizmente era muito cedo para que o vídeo por demanda bem como as empresas de TV a cabo pudessem viabilizar o negócio, pois o custo era alto demais e a infraestrutura era escassa.

Hoje em dia, já é realidade em programas interativos e também na televisão digital, porém a ideia do *7 na época não emplacou. Permitir ao telespectador interagir com a emissora e com a programação em uma grande rede cabos, era algo muito visionário e estava muito longe do que as empresas de TV a cabo tinham capacidade de entender e comprar. A idéia certa, na época errada.

Com a alta da Internet, rapidamente uma grande rede interativa estava se estabelecendo; Era este tipo de rede interativa que a equipe do *7 estava tentando vender para as empresas de TV a cabo. E, em poucos meses, não era mais necessário construir a infra-estrutura para a rede, ela simplesmente estava lá. Gosling foi incumbido de adaptar o Oak para a Internet e, em janeiro de 1995, foi lançada uma nova versão do Oak rebatizada para Java.

Figura 2 – Simbolo Java


Disponivel em:
<https://fernandofranzini.wordpress.com/2015/06/10/comemoracao-20-anos-de-java/>.
Acesso em abr. 2018.

A tecnologia Java tinha sido projetada para se expandir por meio das redes de dispositivos heterogêneos, redes como a Internet. Agora aplicações poderiam ser executadas dentro dos Browsers nos Applets Java e tudo seria disponibilizado pela Internet de forma instantânea. Foi o HTML estático dos Browsers que promoveu a rápida disseminação da tecnologia Java. A velocidade dos acontecimentos seguintes foi tremenda, o número de usuários cresceu rapidamente, grandes empresas do ramo, como a IBM anunciaram suporte para a tecnologia Java.

Desde seu lançamento, em maio de 1995, a plataforma Java foi adotada mais rapidamente do que qualquer outra linguagem de programação na história da computação. Em 2003 Java atingiu a marca de 4 milhões de desenvolvedores em todo mundo. Java continuou e continua crescendo e hoje é com certeza um padrão para o mercado oferecendo qualidade, performance e segurança ainda sem nenhum competidor a altura. Java tornou-se popular pelo seu uso na Internet e hoje possui seu ambiente de execução presente em web browsers, mainframes, SOs, celulares, palmtops e cartões inteligentes, entre outros.

Em 2008 a Oracle Corporation adquire a empresa responsável pela linguagem Java, a Sun Microsystems, por US$ 7,4 bilhões, com o objetivo de levar o Java e outros produtos da Sun ao dispor dos consumidores.

O Java foi influenciado pela linguagens C/C++ pela sua sintaxe. E influenciou o javascript, as duas linguagens possuem suporte à programação funcional.

## 2.1 Versões do Java

Desde o lançamento do Java, vêm surgindo várias versões e alterações nas nomenclaturas e conteúdos. Abaixo é apresentada uma linha do tempo referente às evoluções das versões que formaram a linguagem Java:

### 2.1.1 Java Development Kit (JDK ou Java 1.0)

Criada no ano de 1996. É a 1ª versão sendo hoje usada para compatibilidade de browsers mais antigos;

### 2.1.2 Java Development Kit (JDK ou Java 1.1)

Criada no ano de 1997. Obteve muitas bibliotecas adicionadas das quais se destacaram o Java RMI, JavaBeans, novo modelo de eventos, JDBC (driver para conexão com banco de dados).

### 2.1.3 Java Standard Edition (J2SE 1.2 ou Java 2)

Criada no ano de 1998. Grande aumento das classes na biblioteca Java (API), ficando considerada a versão da mudança do nome para as versões do produto (JDK) e

também sendo optada pela divisão de 3 tipos de plataformas. O principal motivo para essa ação foi que muitos desenvolvedores e usuários estavam confundindo a linguagem Java da linguagem Javascript, que são diferentes. A partir daqui todas as versões Java foram denominadas de Java 2 Standard Edition, que passaram a ter apelidos ou codinomes, esta versão ficou conhecida como Playground da qual foi adicionado o Framework Collections entre outros.

### 2.1.4 Java Standard Edition - J2SE 1.3

Criada no ano de 2000. Codinome Kestrel, inclusão das bibliotecas JNDI, JavaSound entre outros.

### 2.1.5 Java Standard Edition J2SE 1.4

Criada no ano de 2002. Codinome Merlin, criada a palavra reservada "assert", biblioteca NIO entre outros.

### 2.1.6 Java Standard Edition J2SE 5.0

Criada no ano de 2004. A versão mais usada, sendo conhecida com o codinome Tiger. Apesar da versão ser 1.5, agora é chamada apenas de 5. Adições importantes como: Enumeração, Autoboxing, Generics, for-each entre outros estão nela.

### 2.1.7 Java Standard Edition JSE 6

Criada no ano de 2006. Codinome Mustang, teve outras alterações que mudaram na nomenclatura (remoção do 2 - J2SE) e melhora significativa na performance e na estabilidade tendo o surgimento do JIT.

### 2.1.8 Java Standard Edition JSE 7

Criada no ano de 2011. Suporte ao uso de strings em condições do switch. Inferência na criação de objetos com tipos genéricos. Simplificação na invocação de métodos com parâmetros varargs e tipos genéricos. Gerenciamento automático de recursos, tais como conexões a bancos de dados, I/O. Possibilidade de tratar diversas exceções em um mesmo catch (Multicatch) entre outros.

### 2.1.9 Java Standard Edition JSE 8

Java 8 é a release mais recente do Java, porém em fase de testes, disponível apenas para desenvolvedores. Possui métodos de expressão Lambda e Extensão virtual além de API de Data e Hora.

Uma nova implementação leve de alto desempenho do motor JavaScript foi integrada ao JDK, a Nashhorn JavaScript Engine.

## 2.2 Linguagens Similares

### 2.2.1 Semelhança em Orientação à objetos

As linguagens a seguir tem semelhança com Java em orientação por objetos: VB.NET, Object Pascal, Objective-C, Python, SuperCollider, Ruby, Smalltalk, C#, C++.

### 2.2.2 Semelhança Imperativa

As linguagens a seguir tem semelhança com Java por causa de seu paradigma Imperativo: Ada, ALGOL, Basic, C, PHP, Cobol, Fortran, Pascal, Python, Lua, Mathematica.

### 2.2.3 Semelhança funcional

A partir da versão 8, o Java adiciona aspectos de linguagem funcional, permitindo utilizar técnicas funcionais, como mapeamento, redução, bem como tratar funções como variáveis, para isso são utilizadas interfaces para esse tipo de manipulação. As linguagens a seguir tem semelhança em Java por causa de seu paradigma funcional: APL, Lisp, ML, Haskell, OCaml, F#, Elixir.

# 3 Paradigma

Java é uma linguagem de programação multiparadigma, isto é, ela é baseada em mais de um paradigma ao mesmo tempo. Temos a orientação à objetos, programação imperativa e funcional como seus principais paradigmas.

Linguagens OO(orietada à objetos) descrevem as iterações entre os objetos. Um objeto é formado por estados e funcionamento. Os estados é representados por dados e o funcionamento por funções, que atuam no objeto em questão. Linguagens OO também manipulam os dados de entrada para gerar de dados de saída específicos, e cada objeto tem uma funcionalidade diferente.

No Java todos os objetos são auto-sufiencietes, isto é todos os seus modulos são inerentemente reutilizáveis. Podemos também adicionar novos procedimentos e subclasses a qualquer objeto.

Para o paradigma imperativo, podemos pensar que ele transforma a computação em ações, assemelhando-se com o comportamento das linguagens naturais que expressam ordens. Tal paradigma também pode ser chamado de paradigma procedural, e a forma que este paradigma funciona é baseada em comandos, porque o intuito era ser parecido com a linguagem de maquina.

Java também é funcional, o que pode ser um pouco estranho já que é imperativo, porém é uma linguagem que consegue diferenciar bem os paradigmas de sua computação. A ideia do paradigma funcional é de avaliar funções matematicas e o java serve muito bem para este proposito, como realizar calculos de volumes e areas de funções trigonometricas.

Os demais paradigmas são: estruturado, genérico, reflectivo e concorrente.

# 4 Características

Java é uma linguagem simples em sua especificação, tanto da linguagem em si como do ambiente de execução (JVM), é distribuída com um vasto conjunto de bibliotecas (ou APIs), suporta nativamente caracteres Unicode, possui facilidades para criação de programas distribuídos e multitarefa (múltiplas linhas de execução num mesmo programa) e desalocação de memória automática por um processo conhecido como coletor de lixo.

Confiabilidade: O processo de compilação elimina vários possíveis problemas e a checagem dinâmica (em tempo de execução) contorna várias situações que poderiam gerar erros. Essas situações, chamas exceções, podem ser devidamente tratadas para evitar que o programa aborte.

Segurança: Um programa sempre é verificado antes de ser executado, e como a linguagem não permite acesso direto à memória, impede seu uso para o desenvolvimento de vírus.

Portabilidade: Um programa em Java não depende de uma plataforma específica, é possível executá-lo em qualquer lugar.

Recursos de Rede: Java possui uma extensa biblioteca de rotinas, que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP.

## 4.1 Método Main

O método main é onde o programa inicia. Ele pode estar presente em qualquer classe. Os parâmetros de linha de comando são enviados para o array de Strings chamado args.

```
1 public static void main(String[] args) {
2     // Corpo da main
3 }
```

## 4.2 Classes

Uma classe é um elemento do código Java que utilizamos para representar objetos do mundo real. Dentro dela é comum declararmos atributos e métodos, que representam, respectivamente, as características e comportamentos desse objeto.

```
1 class Carro{
2     // Corpo da classe
3 }
```

É possível especificar o modificador de acesso da classe de forma a determinar sua visibilidade. Pode ser public, private ou protected.

```
public Class Moto {
    // Corpo da classe
}
```

Programas em Java são formados por uma coleção de classes armazenadas independentemente e que podem ser carregadas no momento de utilização. Java implementa os conceitos de orientação à objetos como abstração, encapsulamento, polimorfismo e herança.

```
Class Automovel {
    public void acelerar() {
        // Corpo do metodo
    }
    public void abastecer() {
        // Corpo do metodo
    }
}
Class Carro extends Automovel {
    @Override
    public void acelerar() {
        motor();
    }
    public void abastecer() {
        // Corpo do metodo
    }
    private void motor() {
        // Corpo do metodo
    }
}
```

Java não suporta herança múltipla, devido à possibilidade de uma classe pai ter um método com o mesmo nome de outra classe pai, e gerar possíveis falhas ao chamar o método, e todas as classes em Java derivam da classe Object. A única possibilidade de se ver herança múltipla em Java é no uso de interfaces, pois uma classe pode implementar várias interfaces.

## 4.3 Estruturas

### 4.3.1 “Condicional”

Exemplo da estrutura "If/Else"em Java:

```
if(<condicao>){
    //Condicao verdadeira
}else{
    //Condicao falsa
}
```

Outra estrutura condicional bastante utilizada é o "Switch/case":

```
switch (numero inteiro % 2){
    case 0:
        System.out.println("Numero Par");
        break;
    case 1:
        System.out.println("Numero Impar");
        break;
    default:
        break;
}
```

### 4.3.2 "Repetição"

Existem 3 maneiras de utilizar uma estrutura de repetição em Java, "For","While", "Do while". O exemplo a seguir mostra a estrutura "For":

```
for(int i = 0;i < 10;i++){
    <linha de codigo a ser repetida 10 vezes>
}
```

### 4.3.3 "Tabela unidimensional"

Exemplo da criação de uma tabela unidimensional em Java:

```
public static void main(String[]args){
    int[] array = new int[<tamanho da tabela>];
}
```

### 4.3.4 "Chamada de função"

Exemplo de uma chamada de função em Java:

```
public static void main(String[]args){
    int a = 1;int b = 5;
    int c  = soma(a,b);
}
public int soma(int a,int b)\{
```

```
6       int x;
7       x = a+b;
8       return x;
9   }
```

### 4.3.5 "Tratamento Exceção"

Em caso de erro de compilação, erros devem ser tratados, uma boa maneira de tratá-los é usando Tratamento de Exceção. Segue um exemplo de Tratamento de Exceção:

```
1   try{
2       //Trecho de codigo ao qual deve ser tratado
3       ... (codigo)
4       throw new tipoExcecao(e);
5       ... (codigo)
6   }catch(Exception e){
7       //possiveis acoes e mensagens de erro
8   }
```

### 4.3.6 "Arquivo"

Exemplo de Arquivos em Java:

```
1   File arquivo = new File("texto.txt");
2   if(!arquivo.exists()){
3       System.out.println("O arquivo nao existe");
4   }
```

## 4.4 *Interface*

Uma interface modela um comportamento esperado. Pode-se entendê-la como uma classe que contenha apenas métodos abstratos. Embora uma classe não possa conter mais de uma superclasse, a classe pode implementar mais de uma interface.

```
1   public interface Combustao {
2       void queimarGasolina();
3       void queimarAlcool();
4   }
5   Class Automovel {
6       public void acelerar() {
7           // Corpo do metodo
8       }
9       public void abastecer() {
10          // Corpo do metodo
```

```
11      }
12  }
13  Class Carro extends Automovel implements Combustao {
14      @Override
15      public void acelerar() {
16          motor();
17      }
18      public void abastecer() {
19          // Corpo do metodo
20      }
21      private void motor() {
22          queimarGasolina();
23      }
24      private void queimarGasolina() {
25          // Corpo do metodo
26      }
27      private void queimarAlcool() {
28          // Corpo do metodo
29      }
30  }
```

## 4.5 Objetos Anônimos

```
1  public Class Poluicao {
2      public static void main(String[] args) {
3          // Nao e necessario instanciar o objeto em uma variavel
               para utiliza-lo:
4          new Carro().acelerar();
5      }
6  }
```

## 4.6 Coletor de lixo

O coletor de lixo (garbage collector) é um processo usado para a automação do gerenciamento de memória. Com ele é possível recuperar uma área de memória inutilizada, como uma instância de um objeto que não será mais utilizada no programa, desalocando então a área de memória reservada para aquela instância, o que pode evitar problemas de vazamento de memória, resultando no esgotamento da memória livre para alocação.

## 4.7 Arcabouços

Um arcabouço, também conhecido como framework, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

São utilizados para facilitar o desenvolvimento de aplicações. Os mais utilizados são: Hibernate, Junit, Log4g, Spring e Struts.

## 4.8 Ambientes de desenvolvimento

IDE, do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa que reúne ferramentas necessárias para o desenvolvimento de aplicações de software, como um editor, compilador, depurador e outros. Algumas IDEs mais conhecidas e utilizadas para o desenvolvimento de aplicações em Java são: Eclipse, Netbeans, JGRASP e IntelliJ IDEA.

# 5 Exemplos

## 5.1 Hello world!

Exemplo mais basico das linguagens de programação,o "print"de um Hello world na tela.

```
1 public static void main(String[] args) {
2     System.out.println("Hello World!");
3 }
```

## 5.2 Calculo Lambda

Exemplo de Calculo Lambda em Java.

```
1  import java.util.Arrays;
2  import java.util.List;
3  import java.util.function.IntFunction;
4  import java.util.function.Predicate;
5  public class MyClass {   //exemplo retirado de: https://www.devmedia.com.br/
6      public static void main(String args[]) {
7          System.out.println("Cria a lista com os elementos que serao realizadas operaçoes");
8          List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12);
9
10         System.out.println("Imprime todos os numeros:");
11         avaliaExpressao(list, (n)->true);
12
13         System.out.println("Nao imprime nenhum numero:");
14         avaliaExpressao(list, (n)->false);
15
16         System.out.println("Imprime apenas numero pares:");
17         avaliaExpressao(list, (n)-> n%2 == 0 );
18
19     }
20
21     public static void avaliaExpressao(List<Integer> list, Predicate<Integer> predicate) {
22         list.forEach(n -> {
23           if(predicate.test(n)) {
24               System.out.println(n + " ");
25           }
26         });
27     }
28
29 }
```

Figura 3 – Calculo Lambda

## 5.3 Forma Escalonada Reduzida

Exemplo de uma matriz sendo modificada para sua Forma Escalonada Reduzida, com a finalidade de achar a matriz identidade.

```
1   public double[][] soFer(double[][] Ml, int[] numlinNumcol) throws Exception {
2       double[][] fer = null;
3       fer = new double[numlinNumcol[0]][numlinNumcol[1] + 1];
4
5       // preenchendo a fer com a Matriz AX = B ~~~~ A | B
6       for (int i = 0; i < numlinNumcol[0]; i++) {
7           for (int j = 0; j < numlinNumcol[1]; j++) {
8               fer[i][j] = Ml[i][j];
9           }
10      }
11
12      //começando a escalonar - Regra 1 - jogar linha nula pro final
13      // Jogando as linhas 0 para o final varias vezes
14      int cont = 0;
15      for (int xvezes = 0; xvezes < numlinNumcol[0]; xvezes++) {
16          for (int i = 0; i < numlinNumcol[0]; i++) {
17              for (int j = 0; j < numlinNumcol[1] + 1; j++) {
18                  if (fer[i][j] == 0) {
19                      cont++;
20                  }
21              }
22
23              if (cont == numlinNumcol[1] + 1) {
24                  double[] linhaSwap = new double[numlinNumcol[0] + 1];
25                  linhaSwap = fer[i];
26                  for (int k = i; k < numlinNumcol[1] - 1; k++) {
27                      fer[k] = fer[k + 1];
28                  }
29                  fer[numlinNumcol[1] - 1] = linhaSwap;
30              }
31              cont = 0;
32          }
33      }
34
```

```
34
35      //array de linhas zeradas
36      int[] linhasZeradas = linhasZeradas(fer);
37      cont = 0;
38      //setando linhas para -1
39      for (int i = 0; i < numlinNumcol[0]; i++) {
40          linhasZeradas[i] = -1;
41      }
42      //array de linha zeradas com valor != -1 significa que é uma linha zerada
43      for (int i = 0; i < numlinNumcol[0]; i++) {
44          for (int j = 0; j < numlinNumcol[1] + 1; j++) {
45              if (fer[i][j] == 0) {
46                  cont++;
47              }
48          }
49
50          if (cont == numlinNumcol[1] + 1) {
51              linhasZeradas[i] = i;
52          }
53          cont = 0;
54      }
55
```

```
        int countSwapErro = 1;
        for (int i = 0; i < numlinNumcol[0]; i++) {
            for (int j = i; j < numlinNumcol[1] + 1; j++) {
                int[] posicaoPivo = new int[2];
                posicaoPivo[0] = i;
                posicaoPivo[1] = j;
                if ((posicaoPivo[0] > fer[0].length - 1) && (fer[i][j] == 0) && j == numlinNumcol[1] + 1) {
                    return fer;
                } else if ((fer[i][j] == 0) && (i < fer.length - 1)) {
                    boolean swaptrue = true;
                    boolean resp = false;
                    int k = 0;
                    for (k = i + 1; swaptrue && k < fer.length; k++) {
                        if (fer[k][j] != 0) {
                            swaptrue = false;
                            resp = true;
                        }
                    }
                    k--;
                    if (resp) {
                        swap(fer, linhasZeradas, i, k);
                        j--;
                    }
                } else if (fer[i][j] == 1) {
                    for (int k = 0; k < numlinNumcol[0]; k++) {
                        while (fer[k][j] != 0.0 /*&& k != j*/ && (k != posicaoPivo[0])) {
                            System.out.println("zerando coluna: " + (j + 1) + " na linha " + (k + 1));
                            double inv = -fer[k][j];
                            for (int l = j; l < numlinNumcol[1]; l++) {
                                //System.out.println("linha = " + (-fer[k][j]) + " * " + fer[i][l] + " + " + fer[k][l]);
                                fer[k][l + 1] = (inv * fer[i][l + 1]) + fer[k][l + 1];
                            }
                            fer[k][j] = 0.0;
                        }
                    }
                    j = numlinNumcol[1] + 1;
                } else if ((fer[i][j] != 0) && (fer[i][j] != 1) && (j < numlinNumcol[1])) {
                    //dividir(pra pivo) pra depois zerar
                    System.out.println("divindo linha: " + (i + 1));
                    int k = i;
                    double div = fer[i][j];
                    for (int l = j; l <= numlinNumcol[1]; l++) {
                        System.out.println("linha = " + fer[k][l] + " / " + div);
                        fer[k][l] = fer[k][l] / div;
                    }
                    //fer[k][j] = fer[k][j] / div;

                    for (k = 0; k < numlinNumcol[0]; k++) {
                        while ((fer[k][j] != 0.0)/* && (k != j) */ && (k != posicaoPivo[0])) {
                            System.out.println("zerando coluna: " + (j + 1) + " na linha " + (k + 1));
                            for (int l = j; l < numlinNumcol[1]; l++) {
                                //System.out.println("linha = " + (-fer[k][j]) + " * " + fer[i][l+1] + " + " + fer[k][l+1]);
                                fer[k][l + 1] = (-fer[k][j] * fer[i][l + 1]) + fer[k][l + 1];
                            }
                            fer[k][j] = 0.0;
                        }
                    }
                    j = numlinNumcol[1] + 1;
                }
            }
        }

        double[][] fer2 = new double[numlinNumcol[0]][numlinNumcol[1]];

        for (int i = 0; i < numlinNumcol[0]; i++) {
            for (int j = 0; j < numlinNumcol[1]; j++) {
                fer2[i][j] = fer[i][j];
            }
        }

        fer = null;
        fer = fer2;

        return fer;
    }
```

Figura 4 – Forma Escalonada Reduzida

# 6 Considerações Finais

Java é uma linguagem de programação atual com paradigma orientado à objetos, imperativa e funcional, sendo ela uma linguagem forte e estática, ela se destacou no mercado pela sua ampla plataforma, pois a mesma consegue programar não só para desktop, mas também para aplicativos celulares, cartões de banco, programação web e até televisão digital. Java também consegue programar não só em windows, mas também em Mac e GNU/Linux.

# Referências

Ed Tittle. *60 minutos para aprender JAVA*. Berkeley Brasil, São Paulo, 1996. ISBN: 85-7251-392-2.

Alexander Newman. *Usando Java*. Campus, Rio de Janeiro, 1997. ISBN: 85-352-0098-3.

The java programming language and the java platform. oracle. <http://www.oracle.com/technetwork/topics/newtojava/downloads/index.html>. Accessed: 2018-05-12.

Java: Declaração e utilização de classes. devmedia. <https://www.devmedia.com.br/java-declaracao-e-utilizacao-de-classes/38374>. Accessed: 2018-05-12.

Java – principais características. tableless. <https://tableless.com.br/java-principais-caracteristicas/>. Accessed: 2018-05-12.

Coletor de lixo (informática). wikipedia. <https://pt.wikipedia.org/wiki/Coletor_de_lixo_(informática)>. Accessed: 2018-05-12.

Framework. wikipedia. <https://pt.wikipedia.org/wiki/Framework>. Accessed: 2018-05-13.

Ambiente de desenvolvimento integrado. wikipedia. <https://pt.wikipedia.org/wiki/Ambiente_de_desenvolvimento_integrado>. Accessed: 2018-05-13.

Do c/c++ para o java: Conheça as diferenças e principais características. devmedia. <https://www.devmedia.com.br/do-c-c-para-o-java-conheca-as-diferencas-e-principais-caracteristicas/26773>. Accessed: 2018-05-10.

Java (linguagem de programação). wikipedia. <https://pt.wikipedia.org/wiki/Java_(linguagem_de_programaç~ao)>. Accessed: 2018-05-13.

Entendendo e conhecendo as versões do java. devmedia. <https://www.devmedia.com.br/entendendo-e-conhecendo-as-versoes-do-java/25210>. Accessed: 2018-05-07.

Duke the java mascot. oracle. <https://www.oracle.com/java/duke.html>. Accessed: 2018-05-07.