

METODOLOGIA

- › Interprete o documento calmamente e com atenção.
- › Acompanhe a execução do exercício no seu computador.
- › Não hesite em consultar o formador para o esclarecimento de qualquer questão.
- › Não prossiga para o ponto seguinte sem ter compreendido totalmente o ponto anterior.
- › Caso seja necessário, execute várias vezes o exercício até ter compreendido totalmente o processo.

CONTEÚDO PROGRAMÁTICO

1. [Conceitos básicos de programação](#)
2. [Compilador e programa executável](#)
3. [Apresentação da JVM](#)

1. Conceitos básicos de programação

Um computador, independentemente do fabricante, da dimensão ou do objetivo, é sempre composto por dois grandes componentes:

- **Hardware** – o conjunto de componentes eletrónicos, ou peças físicas, que constituem o computador (processador ou CPU, memória, disco, monitor, teclado, rato, placa gráfica, etc.).
- **Software** – o conjunto de instruções que indicam ao computador “o que fazer”.

Estes dois componentes não podem existir um sem o outro. O software não é mais que um conjunto de instruções que devem ser executadas em sequência pelo computador no sentido de atingir um determinado objetivo. A programação consiste em escrever essas instruções num programa que o computador irá executar. As instruções que são “entregues” ao computador para execução dependem do tipo de processador e fabricante.

Para perceber estes conceitos vejamos o exemplo de um automóvel. Um automóvel é composto pelo motor, a carroçaria, as rodas, etc. Ao conjunto das peças que compõem o automóvel chamemos-lhe “hardware”. No entanto, um automóvel não circula sozinho, necessita de um condutor. Ora, vamos chamar ao condutor o “software”. Cada condutor está habilitado e tem capacidades para conduzir um tipo de automóvel específico.

Em comparação, cada programa tem que ser especificamente preparado para cada computador onde deverá ser executado.

Existem diversas linguagens de programação disponíveis, no entanto, cada uma delas é mais direcionada para alguns objetivos específicos. Eis apenas alguns exemplos:

- **Basic:** uma das primeiras linguagens de programação, bastante limitada, que permitia fazer programas relativamente simples para computadores pessoais.
- **C:** uma das primeiras linguagens para grandes computadores centrais (mainframes) com um enorme potencial e número de funcionalidades. No entanto, é uma linguagem extremamente complexa que exige algum esforço em termos de programação, manutenção e aprendizagem.
- **C++:** linguagem de programação orientada por objetos (vamos aprofundar este ponto mais à frente) que teve origem na linguagem C. Tem a grande vantagem de utilizar todas as potencialidades do C, sendo mais simples de programar e manter.
- **Prolog e Lisp:** linguagens de programação muito utilizadas em sistemas de inteligência artificial.
- **Visual Basic:** linguagem de programação que permite desenvolver aplicações para computadores pessoais, com o tipo de interfaces gráficas que estamos habituados a utilizar. Bastante simples de utilizar permitindo o rápido desenvolvimento de programas.
- **Java:** uma das linguagens mais recentes e tecnologicamente mais avançadas que veio revolucionar os conceitos de programação. É uma linguagem orientada à programação por objetos, com uma sintaxe semelhante à do C, mas que disponibiliza imensas ferramentas e facilidades para desenvolvimento, que se traduz no aumento da produtividade.

Em conclusão, ao conjunto de instruções com um determinado objetivo chamamos **programa** ou **aplicação**.

Existem vários tipos de programas:

- **Programas ou aplicações de interface com o utilizador** – são as que melhor conhecemos e utilizamos no nosso dia-a-dia, instaladas no nosso computador.

- **Programas ou aplicações cliente-servidor** – servem para desempenhar funções ou cálculos complexos, normalmente distribuídos por uma rede de computadores. A este tipo de programas chamamos normalmente “um sistema”. Exemplos disso são sistemas de gestão bancária ou operadores de telecomunicações.
- **Programas ou aplicações servidor (batch)** – estão em constante execução e simplesmente processam informação que recebem a partir de um determinado canal. Por exemplo, um sistema de faturação de chamadas telefónicas num operador de telecomunicações: as chamadas estão todas numa base de dados e o programa percorre todas essas chamadas e atribui-lhes um custo a partir de um conjunto de critérios como a duração, a origem, o destino e plano tarifário.

📄 Solicite mais esclarecimentos ao formador se necessário

2. Compilador e programa executável

Os computadores, ou mais concretamente os processadores, conseguem apenas interpretar instruções em formatos de zeros e uns (0101101001001001 ...), ou seja, instruções em formato binário. As cadeias de zeros e uns que os CPU's entendem são, para nós, totalmente impercetíveis.

Seria uma tarefa quase impossível desenvolver programas utilizando apenas zeros e uns. Desta forma, as linguagens de programação são normalmente criadas com uma sintaxe humanamente interpretável, para facilitar a vida aos programadores.

Vejamos o seguinte exemplo de programa numa linguagem fictícia:

```
programa calcula definido como
```

```
    soma x y
      devolve x + y
    subtrai x y
      devolve x - y
    multiplica x y
      devolve x * y
    divide x y
      devolve x / y
```

fim

Este programa, digamos criado na linguagem X, implementa as funcionalidades de uma calculadora.

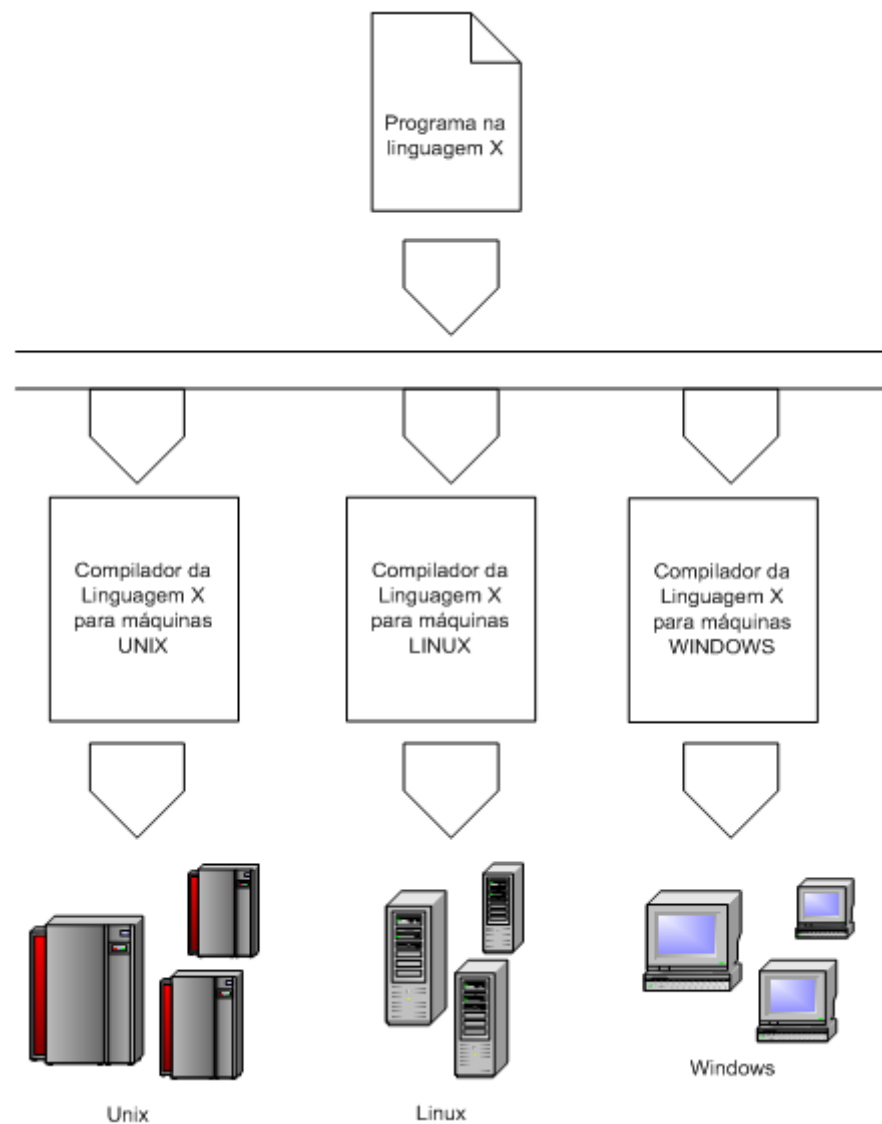
A linguagem X, que conseguimos ler facilmente, não é interpretável pelo CPU para que este consiga executar as instruções necessárias. Hipoteticamente, algo que fosse interpretável pelo CPU, seria algo do género:

```
0100100001011100110001011001001001011010010010010010
0100100000111111010010100100010001001001000010010001
0010010001001001000100010010010010100100100100100100
1000010010001001001001001001001010010010010010010
010010010010010010000100101001001001001001001001010
```

i Cada um dos zeros anteriores representa um bit. Ao conjunto de 8 bits chamamos um byte. Assim sendo, o programa anterior, que é composto por 256 bits, dizemos que ocupa 32 bytes.

Chegámos facilmente à conclusão que é impossível para nós criar algo deste género. O que nos leva a um problema: como traduzir um programa numa linguagem perceptível para um humano, numa linguagem que represente um programa que um CPU possa executar?

A solução é o **compilador**. O compilador não é mais que um tradutor de linguagens, que recebe um programa na linguagem X e o converte para um conjunto de instruções binárias, a executar pelo CPU. Como dissemos anteriormente, cada máquina tem uma forma distinta de interpretar os “zeros e uns”, pelo que terá de existir um compilador para cada linguagem e para cada tipo de máquina onde o programa vai ser executado:



i Ao programa resultante da compilação chamamos programa executável.

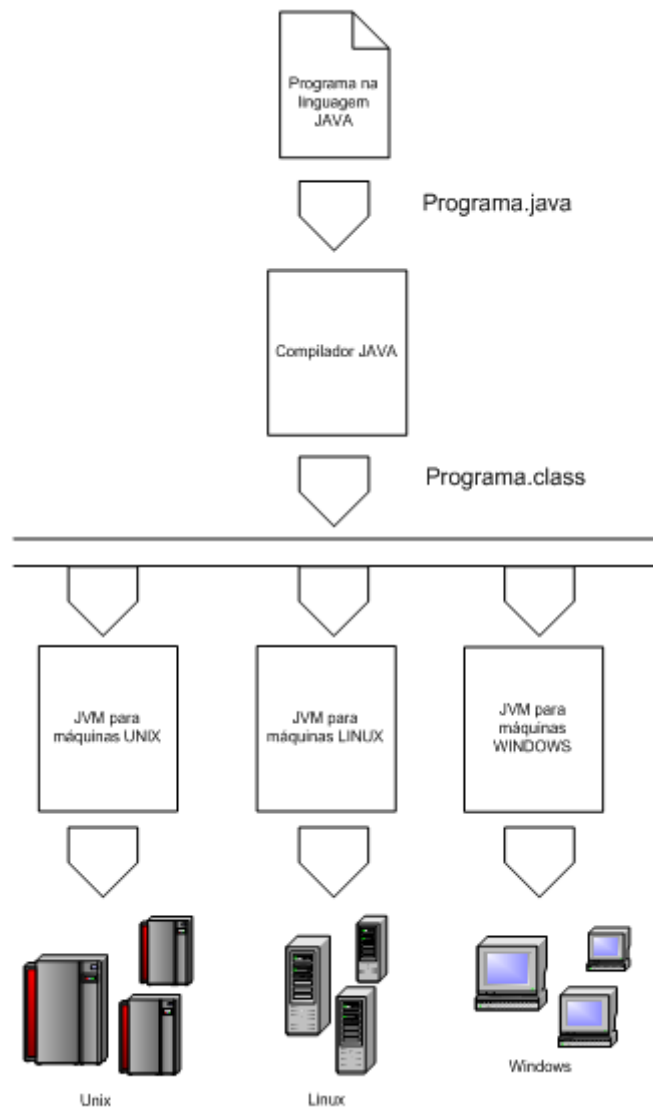
Resumindo, para cada linguagem necessitamos de um compilador específico que produza um programa executável.

3. Apresentação da JVM

Partindo do capítulo anterior, e considerando todas as linguagens de programação e todos os tipos de máquinas, teremos que ter um número inabarcável de compiladores, cada um específico para uma linguagem e uma máquina. Isto traduz-se em enormes problemas de criação e manutenção de programas.

Isto torna-se ainda mais complexo com a expansão da internet, onde o número de utilizadores e máquinas distintas é enorme e onde se pretende que um programa seja “executável” no maior número de máquinas possível para facilitar a sua utilização.

O Java veio revolucionar todos estes conceitos e resolver o problema da proliferação de diferentes dispositivos (computadores pessoais, servidores, telemóveis, smartphones, etc.). Isto porque o Java funciona de forma distinta das outras linguagens de programação. O compilador de Java não gera um programa executável, mas sim um conjunto de instruções **interpretáveis** (ficheiro com extensão **.class**) por um programa específico: A **Java Virtual Machine** (JVM). Assim sendo, um programa Java pode ser utilizado em qualquer máquina que tenha uma JVM instalada.



A Sun Microsystems, empresa que criou o Java, disponibiliza gratuitamente no seu site, JVM's para os mais variados tipos de máquinas, no seguinte endereço: <http://java.sun.com/javase/downloads/>

Existem 2 tipos de "packs" que podemos ir buscar ao site da SUN:

- **JRE** (Java Runtime Environment) – JVM propriamente dita, que permite correr, ou executar, programas java
- **JDK** (Java Developer Kit) – pacote de instalação que inclui o JRE e o “compilador” de java (**javac**) que produz os ficheiros **.class** a serem executados pelas JVM’s

Os packs JRE ou JDK devem ser instalados no computador para ser possível correr ou criar programas Java.

Para criar programas Java, além do compilador, é necessário um editor de texto onde possamos criar (escrever) os programas. Podemos utilizar editores simples, como por exemplo o bloco de notas (Notepad), o Notepad++ ou, em alternativa, utilizar editores **IDE** (Integrated Development Environment) com funcionalidades muito avançadas, como por exemplo o **Eclipse**.

Ao longo deste curso começaremos por utilizar o **Notepad++**, para mostrar que é possível criar programas com um editor tão simples mas, posteriormente, iremos passar a usar o **Eclipse**.

O Notepad++ é um programa gratuito e semelhante ao bloco de notas. A diferença é que possui algumas funcionalidades extra, como a distinção de palavras específicas da linguagem com diferentes cores e ajudas automáticas para estruturar o código.

A tecnologia Java assenta sobre um pilar muito importante: freeware. Para criar programas Java, não necessitamos de comprar qualquer licença.

O JDK deve já estar instalado no computador onde está a trabalhar. Para validar a sua instalação execute os seguintes passos:

› Abra a linha de comandos a partir do menu do **Windows**

📄 Em alternativa pode executar o comando cmd na caixa executar do menu do Windows.


› Execute o comando `java`, seguido de um `ENTER`

📄 O comando `java` permite executar programas java. Deverá obter algo do género:


```
Usage: java [-options] class [args...]  
        (to execute a class)  
or  java [-options] -jar jarfile [args...]  
        (to execute a jar file)
```

Isto indica que deveremos complementar o comando com o nome do ficheiro `.class` a executar.

› **Execute** o comando `java -version` seguido de um `ENTER`

 Este comando permite obter a versão do Java instalado.