05

Compile e rode seu primeiro programa Java

Transcrição

Como falamos no início, em um primeiro contato, o código em Java pode ser complicado de ser escrito e compreendido. Às vezes precisamos escrever um pouco mais do que gostaríamos para fazer algo.

Antes de usarmos um IDE para lidarmos com o código, é legal que você o faça em um sistema bem simples, como o bloco de notas - outras opções são o TextPad, Atom, Visual Studio Code, Sublime, ou qualquer outro.

Nosso primeiro código Java será feito no editor de texto mais simples possível, em *plain text*. Faremos o "Olá mundo" para testarmos e vermos como funciona a compilação e execução de programas Java.

O Java veio da linguagem C na década de 90, então, não é tão simples quanto digitarmos `print("olá mundo")`. A linha que faz um print na tela, por exemplo, é

```
System.out.println("olá mundo");
```

Nesta linguagem, toda instrução que damos sem as chaves necessita do ponto e vírgula (`;`). Todo código Java também precisa estar dentro de uma classe, que pode ser uma interface, um `Enum`. Neste caso, ele se insere na classe `Programa`.

Uma instrução como esta, com `System.out.println()`, precisa estar dentro de um método chamado `main`, que ainda não vimos, acompanhado de outros termos que também aprenderemos depois.

É muito comum o uso de `public` antes de `class Programa`, e embora isto não seja estritamente necessário no nosso caso, vamos colocá-lo para quando formos ler códigos de outros programadores e IDEs.

```
public class Programa {

    public static void main(String[] args) {
        System.out.println("olá mundo");
    }
}
```

No momento, focaremos na linha `System.out.println("olá mundo");`, que poderá ser considerado um comando apesar de não ser um, e mostrará algo na saída padrão, no caso o prompt do MS-DOS.

O menor programa Java seria similar ao código acima. Vamos tentar ver como funciona sua compilação e execução? Antes disso, salvaremos o arquivo nomeando-o com "Programa.java", em uma nova pasta denominada "java-codigo".

O nome do arquivo é muito importante - entenderemos melhor o motivo mais adiante, mas ele precisa ser o mesmo da `class` inserida no código.

No Prompt de Comando, digitaremos `cd ..` duas vezes, seguidos de "Enter", e `dir`, para a listagem de todos os diretórios. Depois, usaremos `cd java-codigo` para acessar o diretório, e em seguida digitaremos `dir` novamente.

Dica: é possível usar a tecla TAB para autocompletar palavras!

Ali, é listado um arquivo "Programa.java"! No Windows, há um comando chamado `type` (equivalente ao `cat` do terminal do Linux), o qual permite a visualização do conteúdo do arquivo. Neste caso, usaríamos `type Programa.java` .

A extensão `.java` não é entendida pela *virtual machine*, que entende o formato "meio máquina" de *Virtual Machine Java*, o ***bytecode***, um arquivo com extensão `.class` .

A seguir, usaremos o comando `javac Programa.java` , e daremos um "Enter", com o qual serão mostradas as mensagens de erro de compilação, fundamentais para o aprendizado.

Apesar de não entendermos o que é `public class` ou `static void main` ainda, sabemos que `System.out.println()` seguido de aspas e o conteúdo, irá mostrar uma mensagem.

Por meio de `dir` no prompt, você verá que há dois arquivos: "Programa.java" e "Programa.class", este último no formato binário, em *bytecode*. E para chamarmos a *virtual machine*, usaremos o comando `java Programa` , e veremos a impressão de "olá mundo". Trata-se da primeira execução do nosso programa Java!

Agora, veremos os principais erros e características deste código. O primeiro surge ao digitarmos `java Programa.class` , o que traz a seguinte mensagem de erro na execução do programa:

```
Erro: Não foi possível localizar nem carregar a classe principal Programa.class
```

Isto acontece porque o programa não se chama "Programa.class", e sim simplesmente "Programa", apesar de estar contido no arquivo "Programa.class".

Outros erros mais comuns são os de compilação, como quando esquecemos de colocar o ponto e vírgula no fim da linha. Além disso, o Java possui palavras chave (*keywords*, ou palavras reservadas), dentre os quais utilizamos "public", "class", "static" e "void", que devem estar em letra minúscula, uma vez que o Java é ***case sensitive*** (reconhece o uso de letras maiúsculas ou minúsculas).

Em um ambiente mais complexo, veremos que isto ficará mais claro e fácil de ser trabalhado. É importante **praticar e não ter medo das mensagens de erro de compilação**.

As chaves abrem e fecham os blocos de códigos, indicando por exemplo que tudo aquilo que se encontra em `public static void main` pertence ao `public class Programa` , da mesma forma que `System.out.println()` pertence ao `public static void main` visível também por meio das indentações.

O Java possui outras particularidades, como o "Enter" e a barra de espaço serem opcionais; são convenções do código. Agora, o importante é escrever, entendendo o que está por trás do código, errar e fazer vários testes!