01

Char e String

Transcrição

A seguir, trabalharemos com caracteres e palavras! Criaremos uma nova classe mais uma vez, a "TestaCaracteres". Existe uma variável primitiva básica do Java que trabalha com *chars*, isto é, caracteres, cuja peculiaridade é guardar um único caractere de `16bits`.

Usaremos as aspas simples para guardar a letra `a`, por exemplo:

```
public class TestaCaracteres {

    public static void main(String[] args) {
        char letra = 'a';
        System.out.println(letra);
    }
}
```

Ao salvarmos e rodarmos este código, lê-se `a` no Console, nada muito especial.

Quando trabalhamos com `char`s, estamos realmente "presos" a um único caractere. Se substituirmos `a` do código acima por `ab`, o código não compilará, e o mesmo ocorrerá se utilizarmos aspas duplas em vez das simples. O `char` guarda em si um único código, um número da tabela de Unicode, como a ASCII, porém muito maior e sem limite definido.

`letra`, portanto, é um número e, se observarmos bem, o `char` guarda em seu `valor` um número, mas é uma variável do tipo numérico equivalente àquele `short`, mas ele contém apenas valores positivos, possuindo mais detalhes. No momento, é interessante sabermos que ele é um número que é convertido em uma letra, como no trecho a seguir:

```
char valor = 66;
System.out.println(valor);
```

A partir do qual obteremos:

```
B
```

Isto ocorre pois na tabela Unicode o `65` corresponde à letra `a`, portanto `66` refere-se a `b`. Testando-se o código abaixo,

```
valor = valor + 1;
System.out.println(valor);
```

há um erro de compilação em `valor + 1`, por conta da regra do Java quando se trabalha com dois tipos distintos em uma mesma operação, de dar o resultado no maior deles. Neste caso, o `valor` é do tipo `char`, e `1` é um `int`, que é maior. O resultado desta operação, portanto, será dado em `int`. No entanto, um inteiro cabe em um `char`? Não! Porém, novamente, o inverso é possível.

Se queremos que isto seja válido, devemos informar que a resposta disso passará pelo *casting*, moldando-se para o `char`:

```
valor = (char) (valor + 1);
System.out.println(valor);
```

Salvando e rodando o código, receberemos a letra `C`. O `char` é interessante, mas não é tão usado no dia a dia, como no caso de `String`, com `S` em maiúsculo. Ela não é palavra chave do Java, não guarda valor, é um tipo referência. As diferenças ficarão mais claras quando formos entender melhor sobre orientação a objetos.

Atenção: o funcionamento básico de uma `String` exige aspas duplas, e não simples, as quais podem inclusive ficar vazias (`""`). Em `char`, por outro lado, não é possível deixar as aspas simples sem nada dentro (`''`) - um espaço seria algo, e compilaria. Um `char` vazio, não.

```
String palavra = "alura cursos online de tecnologia";
System.out.println(palavra);
```

Salvando e rodando o código, teremos a impressão `alura cursos online de tecnologia`, como esperado. E é possível utilizarmos o operador de soma (`+`) para concatenar `String`s, criando uma nova, como no exemplo abaixo:

```
palavra = palavra + 2020;
System.out.println(palavra);
```

Isto nos retornará `alura cursos online de tecnologia2020`. A `String`, então, não se comporta como um `int` ou um `char`, mas aparecerá recorrentemente. Em breve veremos que ela faz referência a um objeto e possui vários métodos. Ainda precisaremos aprender o básico e aprofundarmos nossos conhecimentos com calma!