02

Teste a igualdade entre Strings e outros objetos usando == e equals() - Parte 2

Quando concatenamos literais, a String resultante também será colocada no pool.

```
String ab = "a" + "b";
System.out.println("ab" == ab); // true
```

COPIAR CÓDIGO

Mas isso é verdade **apenas usando literais em ambos os lados da concatenação**. Se algum dos objetos não for um literal, o resultado será um novo objeto, que não estará no pool:

```
String a = "a";
String ab = a + "b"; //usando uma referência e um literal
System.out.println("ab" == ab); // false
```

COPIAR CÓDIGO

Sabemos que Strings são imutáveis, e que cada método chamado em uma String retorna uma nova String, sem alterar o conteúdo do objeto original. Esses objetos resultantes de retornos de métodos não são buscados no pool, são novos objetos:

```
String str = "um texto qualquer";
String txt1 = "texto";
String txt2 = str.substring(3, 8); //cria uma nova string
```

```
System.out.println(txt1 == txt2); // false
System.out.println(txt1.equals(str.substring(3, 8))); // true
```

COPIAR CÓDIGO

os métodos de String sempre criam novos objetos?

Nem sempre. Se o retorno do método for exatamente o conteúdo atual do objeto, nenhum objeto novo é criado:

```
String str = "HELLO WORLD";
String upper = str.toUpperCase();          // já está
maiúscula
String subs = str.substring(0,11);         // string completa
System.out.println(str == upper);          // true
System.out.println(str == subs);           // true
System.out.println(str == str.toString()); // true
```

COPIAR CÓDIGO

Contando Strings

Uma questão recorrente na prova é contar quantos objetos do tipo `String` são criados em um certo trecho de código. Veja o código a seguir e tente descobrir quantos objetos String são criados:

```
String h = new String ("hello ");
String h1 = "hello ";
String w = "world";
```

```
System.out.println("hello ");
System.out.println(h1 + "world");
System.out.println("Hello " == h1);
```

COPIAR CÓDIGO

E então? Vamos ver passo a passo:

```
//Cria 2 objetos, um literal (que vai para o pool) e o outro
//com o new
String h = new String ("hello ");

//nenhum objeto criado, usa o mesmo do pool
String h1 = "hello ";
//novo objeto criado e inserido no pool
String w = "world";

//nenhum  objeto criado, usa do pool
System.out.println("hello ");

//criado um novo objeto resultante da concatenação,
// mas este não vai para o pool
System.out.println(h1 + "world");

//Novo objeto criado e colocado no pool (Hello com H
maiúsculo).
System.out.println("Hello " == h1);     // 1
```

COPIAR CÓDIGO

Logo temos 5 Strings criadas.

Cuidado com String já colocadas no pool

Para descobrir se uma String foi criada e colocada no pool é necessário prestar muita atenção ao contexto do código e ao enunciado da questão. A String só é colocada no pool na primeira execução do trecho de código. Cuidado com questões que criam Strings dentro de métodos, ou que dizem em seu enunciado que o método já foi executado pelo menos uma vez:

```
public class Testes {
    public static void main(String[] args) {
        for(int i = 0; i< 10; i++)
            System.out.println(metodo());
    }

    private static String metodo() {
        String x = "x";
        return x.toString();
    }
}
```

COPIAR CÓDIGO

Ao executar essa classe, apenas **um** objeto String será criado. O único lugar onde a String é criada é na linha 8 do código.