Matéria: Programação

Assunto: Programação Java - Manipulação de Strings

Resumo Teórico do Assunto

Para dominar a manipulação de Strings em Java, especialmente para concursos, é fundamental compreender como elas funcionam e quais ferramentas a linguagem oferece. As questões apresentadas focam em iteração, acesso a caracteres e conversão de caixa, utilizando a classe `Character`.

Programação Java: Manipulação de Strings

Em Java, **Strings** são sequências de caracteres. Uma característica fundamental e crucial para entender seu comportamento é que Strings são **imutáveis**. Isso significa que, uma vez criada, uma String não pode ser alterada. Qualquer operação que pareça modificar uma String (como concatenação ou conversão de caixa) na verdade resulta na criação de uma **nova String**.

 # 1. Declaração e Inicialização de Strings

Uma String pode ser declarada e inicializada de diversas formas:

• **Literal de String:** A forma mais comum e simples.
```java
String nome = "João";
String mensagemVazia = ""; // String vazia
String espaco = " "; // String com um espaço
```
• **Usando o construtor `new String()`:** Menos comum para literais, mas útil em alguns cenários.
```java
String outraNome = new String("Maria");
```

 # 2. Acessando Conteúdo de Strings

Para trabalhar com os caracteres dentro de uma String, você precisa saber seu comprimento e como acessar caracteres individuais.

• **Comprimento da String (`length()`):**

O método `length()` retorna o número de caracteres na String como um `int`.
```java
String texto = "Java";
int tamanho = texto.length(); // tamanho será 4
```
• Caracteres Individuais (`charAt(int index)`):
O método `charAt(int index)` retorna o caractere (`char`) no índice especificado. Lembre-se que a indexação em Java (e na maioria das linguagens) é **baseada em zero**, ou seja, o primeiro caractere está no índice 0, o segundo no índice 1, e assim por diante, até `length() - 1`.
```java
String palavra = "Hello";
char primeiroChar = palavra.charAt(0); // 'H'
char ultimoChar = palavra.charAt(palavra.length() - 1); // 'o'
```

 # 3. Iterando sobre Strings

Existem duas formas comuns de percorrer os caracteres de uma String, como visto nas questões:

• Loop Tradicional (por índice):
Permite controle preciso sobre a posição de cada caractere.
```java
String exemplo = "Teste";
for (int i = 0; i < exemplo.length(); i++) {
char c = exemplo.charAt(i);
// Faça algo com 'c'
}
```
• Enhanced For Loop (For-Each Loop - por caractere):
Mais conciso e legível quando você só precisa acessar os caracteres, sem se preocupar com o índice. **Importante:** Para usar o for-each diretamente em uma String, ela é implicitamente tratada como uma sequência de `char`s.
```java
String exemplo = "Teste";
for (char c : exemplo.toCharArray()) { // Ou, como nas questões, o compilador pode inferir para String
// Faça algo com 'c'
}
// Nota: Embora o código da questão (D) use `for(char x : str)`,
// o método `toCharArray()` é o que explicitamente converte a String em um array de chars
// para que o for-each possa iterar. Em algumas versões do Java ou contextos,
// o compilador pode otimizar ou permitir a sintaxe direta para Strings.
```

// O importante é entender que ele itera sobre caracteres.
```

## 4. Manipulação e Construção de Strings (Concatenção)

Devido à imutabilidade das Strings, qualquer "modificação" na verdade cria uma nova String. As principais formas de concatenar (unir) Strings são:

• **Operador `+`:**
É o método mais comum e legível para concatenar Strings. Quando você usa `+` com Strings, o Java cria uma nova String que é o resultado da união. Para múltiplas concatenações em um loop, o compilador Java geralmente otimiza o uso de `StringBuilder` internamente para melhor performance.
```java
String parte1 = "Olá";
String parte2 = "Mundo";
String saudacao = parte1 + " " + parte2; // "Olá Mundo"
```
• **Método `concat()`:**
O método `concat()` é usado para anexar uma String ao final de outra. Ele também retorna uma **nova String**.
```java
String base = "Início";
String resultado = base.concat("Fim"); // "InícioFim"
```
Diferença prática: O operador `+` é mais flexível, podendo concatenar Strings com outros tipos de dados (que são automaticamente convertidos para String). O `concat()` só aceita argumentos do tipo `String`.

5. A Classe `Character`

A classe `java.lang.Character` fornece métodos utilitários para trabalhar com valores primitivos `char`. Ela é essencial para operações como verificação de tipo de caractere e conversão de caixa.

• **`Character.isLowerCase(char c)`:**
Retorna `true` se o caractere `c` for uma letra minúscula, `false` caso contrário.
```java
boolean ehMinuscula = Character.isLowerCase('a'); // true
boolean ehMaiuscula = Character.isLowerCase('A'); // false
```
• **`Character.toUpperCase(char c)`:**
Converte o caractere `c` para sua versão maiúscula. Se o caractere não tiver uma versão maiúscula (ex: um número, um símbolo), ele retorna o próprio caractere.
```

```java
char maiuscula = Character.toUpperCase('b'); // 'B'
char numero = Character.toUpperCase('5'); // '5'
```

• **`Character.toString(char c)`:**
Converte um valor primitivo `char` em um objeto `String` contendo apenas aquele caractere. Isso é útil quando você precisa concatenar um `char` usando o método `concat()`, que espera um argumento `String`.
```java
char letra = 'X';
String strLetra = Character.toString(letter); // "X"
```

 # Conhecimento Essencial para Resolver as Questões:

1. **Imutabilidade de Strings:** Entender que operações como `r = r + ...` ou `r = r.concat(...)` sempre criam uma **nova String** e atribuem essa nova String à variável `r`. A String original não é modificada.
2. **Acesso a Caracteres:** Saber usar `length()` para o tamanho e `charAt(index)` para acessar caracteres individuais.
3. **Iteração:** Estar confortável com ambas as formas de loop (`for` tradicional com índice e `for-each` para caracteres).
4. **Classe `Character`:** Dominar os métodos `isLowerCase()`, `toUpperCase()`, e `toString()` para manipulação de caracteres.
5. **Concatenção:** Conhecer o operador `+` e o método `concat()` e quando usar cada um.

Ao dominar esses conceitos, você estará bem preparado para analisar e entender o comportamento de códigos que manipulam Strings em Java, como os apresentados nas questões.

Questões de Provas Anteriores

Fonte: escriturario_agente_de_tecnologia.pdf, Página: 27

pcimarkpci MjgwNDowMTRkOjE0YTU6OTI1ODozOGQ2OjNhMGM6NTM0MzplZmI1:U3VuLCAyNyBKdWwgMjAyNSAyMzo0NzozMSAtMDMwMA==
www.pciconcursos.com.br
27
BANCO DO BRASIL
AGENTE DE TECNOLOGIA - Microrregião 158 -TIGABARITO 1

(D) public static String converte(String str) {
String r= " ";

for(char x : str)
if(Character.isLowerCase(x))
r=r.concat(Character.toString(Character.toUpperCase(x)));
else
r=r.concat(Character.toString(x));

return r;
}
(E) public static String converte(String str) {
String r= " ";

for(int i=0; i < str.length(); i++)
if(Character.isLowerCase(str.charAt(i)))
r=r+Character.toUpperCase(str.charAt(i));
else
r=r+str.charAt(i);

return r;
}
RASCUNHO