Unicesumar

Programação I

Operações com String

Prof. Me. Fábio Perfetto

Lendo caracteres de uma String

Podemos fazer a leitura do conteúdo de uma String, caracter a caracter e atribuir ou alterar estes valores :

Método charAt

```
 Scanner ler = new Scanner(System.in);
char Acpf [] = new char [11];
 System.out.println("digite o CPF");
  String CPF = ler.nextLine();
  for(int i = 0; i < CPF.length(); i++) {
  Acpf[i]=CPF.charAt(i); }
```

Metodo Raplace

Precisamos sumir com os caracteres indesejados sem remover nenhum dígito ou mudar a ordem dos caracteres da nossa String. Para isso podemos usar o método replace() da classe String.

Este método nos permite substituir todas as ocorrências do caractere que passarmos por outro caractere.

Então primeiro vamos remover todos os ".". Dessa forma temos:

```
public static String formataDados(String dado){
  dado = dado.replace(".","");
  return dado;
}
```

Perceba que o primeiro argumento que passamos foi ".", pois ele é o caractere que queremos substituir. Logo em seguida passamos uma String vazia representada pelas aspas duplas juntas.

Com isso estamos dizendo que qualquer ponto que ele ache na nossa String será substituído por nada, logo, será removido.

Variações método Replace

- **replace :** vai substituir todas as ocorrências de uma substring que casar na string, mas sem processar expressões regulares.
- **replaceAll :** vai substituir todas as ocorrências de uma substring que casar na string, mas com suporte a expressões regulares.
- **replaceFirst :** vai substituir a primeira ocorrência de uma substring que casar na string, com suporte a expressões regulares.

Convertendo caracter para inteiro

Utilizando o método Character.getNumericValue é possível converter uma variável do tipo caracter para uma variável do tipo inteiro.

```
int icpf[] = new int [11];
   for (int i=0; i<11; i++){
       icpf[i]= Character.getNumericValue(Acpf[i]);
    }
```

Validação de CPF

Primeiramente multiplica-se os 9 primeiros dígitos pela sequência decrescente de números de 10 à 2 e soma os resultados. Assim:

5 * 10 + 2 * 9 + 9 * 8 + 9 * 7 + 8 * 6 + 2 * 5 + 2 * 4 + 4 * 3 + 7 * 2

O resultado do nosso exemplo é:

295

O próximo passo da verificação também é simples, basta multiplicarmos esse resultado por 10 e dividirmos por 11.

295 * 10 / 11

O resultado que nos interessa na verdade é o RESTO da divisão. Se ele for igual ao primeiro dígito verificador (primeiro dígito depois do '-'), a primeira parte da validação está correta.

Observação Importante: Se o resto da divisão for igual a 10, nós o consideramos como 0.

Vamos conferir o primeiro dígito verificador do nosso exemplo:

O resultado da divisão acima é '268' e o RESTO é 2

Isso significa que o nosso CPF exemplo passou na validação do primeiro dígito.

Validação – Parte 2

A validação do segundo dígito é semelhante à primeira, porém vamos considerar os 9 primeiros dígitos, mais o primeiro dígito verificador, e vamos multiplicar esses 10 números pela sequencia decrescente de 11 a 2. Vejamos:

5 * 11 + 2 * 10 + 9 * 9 + 9 * 8 + 8 * 7 + 2 * 6 + 2 * 5 + 4 * 4 + 7 * 3 + 2 * 2

O resultado é:

347

Seguindo o mesmo processo da primeira verificação, multiplicamos por 10 e dividimos por 11.

347 * 10 / 11

Verificando o RESTO, como fizemos anteriormente, temos:

O resultado da divisão é '315' e o RESTO é 5

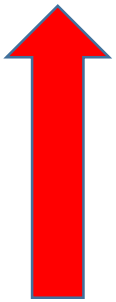
Verificamos, se o resto corresponde ao segundo dígito verificador.

Com essa verificação, constatamos que o CPF 529.982.247-25 é válido.

Resto

O operador que atribui o resto é %.

```
digi1 = (295*10)%11;
```




Unicesumar