**Validando o CPF**

**Para calcular o 1º dígito verificador**

* Cada um dos nove primeiros números do CPF é multiplicado por um peso que começa de 10 e que vai sendo diminuido de 1 a cada passo, somando-se as parcelas calculadas:

sm = (5\*10)+(4\*9)+(6\*8)+(4\*7)+(7\*6)+(1\*5)+(4\*4)+(2\*3)+(9\*2) = 249;

* Calcula-se o dígito através da seguinte expressão:

11 - (sm % 11) = 11 - (249 % 11) = 11 - 7 = 4

**obs.** Se o resto da divisão (operador %) calculado for 10 ou 11, o dígito verificador será 0; nos outros casos, o dígito verificador é o próprio resto.

**Para calcular o 2º dígito verificador**

* Cada um dos dez primeiros números do CPF, considerando-se aqui o primeiro DV, é multiplicado por um peso que começa de 11 e que vai sendo diminuido de 1 a cada passo, somando-se as parcelas calculadas:

sm = (5\*11)+(4\*10)+(6\*9)+(4\*8)+(7\*7)+(1\*6)+(4\*5)+(2\*4)+(9\*3)+(4\*2) = 299;

* Calcula-se o dígito através da seguinte expressão:

11 - (sm % 11) = 11 - (299 % 11) = 11 - 2 = 9

**obs.** se o resto da divisão (operador %) calculado for 10 ou 11, o dígito verificador será 0; nos outros casos, o dígito verificador é o próprio resto.

**Desenvolvendo a Aplicação Java**

Criar um novo projeto no ambiente de desenvolvimento NetBeans através da opção do menu "Arquivo|Novo Projeto..." e realizar os dois passos descritos a seguir:

**1º Passo**

Criar um pacote Java e identificá-lo com o nome "meuPacote". No pacote criado implementar a classe Java ValidaCPF, mostrada na **Listagem 1**, com os métodos:

**isCPF()** - retorna um valor booleano que indica se o número de CPF é válido ou inválido;

**imprimeCPF()** - retorna o CPF formatado na máscara: 999.999.999-99.

package meuPacote;

import java.util.InputMismatchException;

public class ValidaCPF {

public static boolean isCPF(String CPF) {

// considera-se erro CPF's formados por uma sequencia de numeros iguais

if (CPF.equals("00000000000") ||

CPF.equals("11111111111") ||

CPF.equals("22222222222") || CPF.equals("33333333333") ||

CPF.equals("44444444444") || CPF.equals("55555555555") ||

CPF.equals("66666666666") || CPF.equals("77777777777") ||

CPF.equals("88888888888") || CPF.equals("99999999999") ||

(CPF.length() != 11))

return(false);

char dig10, dig11;

int sm, i, r, num, peso;

// "try" - protege o codigo para eventuais erros de conversao de tipo (int)

try {

// Calculo do 1o. Digito Verificador

sm = 0;

peso = 10;

for (i=0; i<9; i++) {

// converte o i-esimo caractere do CPF em um numero:

// por exemplo, transforma o caractere '0' no inteiro 0

// (48 eh a posicao de '0' na tabela ASCII)

num = (int)(CPF.charAt(i) - 48);

sm = sm + (num \* peso);

peso = peso - 1;

}

r = 11 - (sm % 11);

if ((r == 10) || (r == 11))

dig10 = '0';

else dig10 = (char)(r + 48); // converte no respectivo caractere numerico

// Calculo do 2o. Digito Verificador

sm = 0;

peso = 11;

for(i=0; i<10; i++) {

num = (int)(CPF.charAt(i) - 48);

sm = sm + (num \* peso);

peso = peso - 1;

}

r = 11 - (sm % 11);

if ((r == 10) || (r == 11))

dig11 = '0';

else dig11 = (char)(r + 48);

// Verifica se os digitos calculados conferem com os digitos informados.

if ((dig10 == CPF.charAt(9)) && (dig11 == CPF.charAt(10)))

return(true);

else return(false);

} catch (InputMismatchException erro) {

return(false);

}

}

public static String imprimeCPF(String CPF) {

return(CPF.substring(0, 3) + "." + CPF.substring(3, 6) + "." +

CPF.substring(6, 9) + "-" + CPF.substring(9, 11));

}

}

Na implementação de uma classe ou de um **conjunto de classes em um pacote Java** o código fonte deve ser iniciado com a diretiva package, seguida do nome do pacote (meuPacote). Um pacote ou package em **Java** nada mais é do que um conjunto de classes (framework). Usualmente, são colocadas em um package classes relacionadas, construídas com um propósito comum; assim, sob certos aspectos, os packages reproduzem a ideia das bibliotecas de código (libraries e unit's) de outras linguagens de programação.

Outro aspecto destacável da programação foi a utilização da estrutura de controle de erros try – catch que irá proteger o código para eventuais erros de conversão de tipo feita pelo operador de coerção (int).

#### 2º Passo

Implementar a classe principal **Exemplo** mostrada na **Listagem 2**.

import java.util.Scanner;

// importanto a classe "ValidaCPF" do pacote "meuPacote"

import meuPacote.ValidaCPF;

public class Exemplo {

public static void main(String[] args) {

Scanner ler = new Scanner(System.in);

String CPF;

System.out.printf("Informe um CPF: ");

CPF = ler.next();

System.out.printf("\nResultado: ");

// usando os metodos isCPF() e imprimeCPF() da classe "ValidaCPF"

if (ValidaCPF.isCPF(CPF) == true)

System.out.printf("%s\n", ValidaCPF.imprimeCPF(CPF));

else System.out.printf("Erro, CPF invalido !!!\n");

}

}

}

Alguns aspectos da programação utilizada na classe ValidaCPF apresentada na **Listagem 1** foram abordados no post:

O que são String em Java: Métodos da Classe String: **[String em Java.](https://www.devmedia.com.br/trabalhando-com-string-string-em-java/21737" \o "Trabalhando com string" \t "_blank)**

Testando a execução da aplicação da **Listagem 2**:

|  |
| --- |
|  |
| 1. CPF 444.444.444-44 é inválido:CPF | 2. CPF é um número formado por 11 dígitos:CPF |
| 3. Existem caracteres alfabéticos no CPF:CPF | 4. CPF informado está correto (ou válido):CPF |

**Tabela 1**. Testando a execução da aplicação da **Listagem 2**

**Fonte: https://www.devmedia.com.br/validando-o-cpf-em-uma-aplicacao-java/22097**