COMPLIANCE & QUALITY ASSURANCE

AVALIAÇÃO DE ACOMPANHAMENTO (CHECKPOINT) - JUNIT

**Especificação de requisito**

Crie um programa que faça a validação de um número de CPF informado.

Considere que o CPF é dividido em duas partes: os 1os nove dígitos da esquerda para a direita e os dois últimos (10º e 11º) são dígitos de controle. Exemplo nnnnnnnnn.cc

O algoritmo de validação do CPF calcula o primeiro dígito verificador a partir dos 9 primeiros dígitos do CPF, e em seguida, calcula o segundo dígito verificador a partir dos 9 (nove) primeiros dígitos do CPF, mais o primeiro dígito, obtido na primeira parte.

Tomes como exemplo o CPF fictício : **111.444.777**-05.

a - Cálculo do primeiro dígito

O primeiro passo é calcular o primeiro dígito verificador, e para isso, separamos os primeiros 9 dígitos do CPF (111.444.777) e multiplicamos cada um dos números, da direita para a esquerda por números crescentes a partir do número 2, como no exemplo abaixo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **4** | **4** | **4** | **7** | **7** | **7** |
| **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** |
| **10** | **9** | **8** | **28** | **24** | **20** | **28** | **21** | **14** |

Multiplicamos cada digito do CPF pelo respectivo número e somamos cada um dos resultados : 10+9+8+28+24+20+28+21+14 = 162

Pegamos o resultado obtido 162 e dividimos por 11.  Consideramos como quociente apenas o valor inteiro.

162 / 11  =    14  com resto 8

- Se o resto da divisão for menor que 2, então o dígito é igual a 0 (Zero).  
- Se o resto da divisão for maior ou igual a 2, então o dígito verificador é igual a 11 menos o resto da divisão (11 - resto).

No nosso exemplo temos que o resto é 8 então faremos 11-8 = 3

Logo o primeiro dígito verificador é 3. Então podemos escrever o CPF com os dois dígitos calculados :  111.444.777-3X

b - Cálculo do segundo dígito

Para  calcular o segundo dígito vamos usar o primeiro digito já calculado. Vamos montar a mesma tabela de multiplicação usada no cálculo do primeiro dígito. Só que desta vez usaremos na segunda linha os valores 11,10,9,8,7,6,5,4,3,2 já que estamos incluindo mais um digito no cálculo(o primeiro dígito calculado):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **4** | **4** | **4** | **7** | **7** | **7** | **3** |
| **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** |
| **11** | **10** | **9** | **32** | **28** | **24** | **35** | **28** | **21** | **6** |

Novamente, efetuamos somamos o resultado da multiplicação : 11 + 10 + 9 + 32 + 28 + 24 + 35 + 28 + 21 + 6 = 204

Dividimos o total do somatório por 11 e consideramos o resto da divisão.

204 / 11  =  18  e  resto 6

Após obter o resto da divisão, precisamos aplicar a mesma regra que utilizamos para obter o primeiro dígito:

- Se o resto da divisão for menor que 2, então o dígito é igual a 0 (Zero).  
- Se o resto da divisão for maior ou igual a 2, então o dígito é igual a 11 menos o resto da divisão (11 - resto).

11-6= 5   logo 5 é o nosso segundo dígito verificador.

Logo o nosso CPF fictício será igual a : 111.444.777-35.

**Desafio**

Crie um projeto JAVA com a classe CodigoPessoaFisica fornecida a seguir.

Essa Classe fornecida possui um método chamado validaCPF, que serve para avaliar se número de CPF informado está correto, retornando true se sim e false, se não for válido.

Crie uma Classe JUNIT de testes com os métodos para validar se o método validaCPF está funcionando em caso positivo (cpf informado é correto) e em caso negativo (cpf informado é incorreto).

Crie também um teste na mesma classe JUNIT, o qual aciona o método removeCaracteresEspeciais, o qual deve receber um CPF com “.” e “-“ e, retornar só os números (a implementação desse método está no fonte a seguir, junto com o método validaCPF).

Em um documento Word ou em editor equivalente, coloque os textos dos códigos fontes das JUNITs Test Cases e cole junto as imagens dos resultados da execução das JUNIT Cases.

Ao final, gere um PDF e entregue na área de Entrega de Trabalhos que foi aberta pelo professor com o nome Checkpoint-Junit.

OBS: Não altere a Classe de Objetos que foi fornecida (a ser testada). Seu foco é criar, executar os testes e observar se a Classe fornecida funciona ou não. Como o programa fonte não tem main(), não tente rodar a Classe fornecida diretamente – rode as JUNITs!

Segue a Classe a testar:

**import** java.util.InputMismatchException;

**public** **class** CodigoPessoaFisica {

**public** **boolean** validaCPF(String CPF) {

CPF = removeCaracteresEspeciais(CPF);

// considera erro CPF formado por uma sequência de nros 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 contra erros de conversao de tipo (int)

**try** {

// Cálculo do 1o. Digito Verificador

sm = 0;

peso = 10;

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

// converte o i-esimo caractere do CPF em número:

// por exemplo, transforma o caractere '1' no nro 1

// (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 numérico

// Cálculo 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 dígitos calculados conferem com os dígitos informados.

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

**return** (**true**);

**else**

**return** (**false**);

} **catch** (InputMismatchException erro) {

**return** (**false**);

}

}

**public** String removeCaracteresEspeciais(String doc) {

**if** (doc.contains(".")) {

doc = doc.replace(".", "");

}

**if** (doc.contains("-")) {

doc = doc.replace("-", "");

}

**if** (doc.contains("/")) {

doc = doc.replace("/", "");

}

**return** doc;

}

}

Bom trabalho!

O feedback da avaliação será publicado no mesmo local onde você entregar seu arquivo, no local apontado como comentários do professor.