Universidade Federal do Espírito Santo
Centro Tecnológico
Departamento de Informática

Programação Orientada a Objetos
(INF 15933)
Prof. Bruno (Adaptado do Prof. Vitor)

Exercício J2_03 de Programação Orientada a Objetos com Java

Sobre a sua solução:

Sua solução **deve conter** ao menos dois arquivos. Um deles implementará a classe Cliente (e portanto, deve ter o nome Cliente.java) com as especificações abaixo. Outro arquivo implementará a função main, com o nome que vocês desejarem.

O script de correção fará dois testes, onde cada um vale parte da "nota":

- Testará o seu código inteiro
- Testará se as suas classes foram implementadas do jeito solicitado (testa ela com uma main pré-definida)

Caso você implemente apenas uma classe Cliente, ou ainda implemente apenas uma main com saídas pré-definidas, você não receberá nota 10.

Crie uma classe `Cliente` com o seguinte contrato:

- Ao criar um novo cliente, é obrigatório informar o nome em seu construtor;
- O método `setCPF()` atribui um número de CPF ao cliente se o mesmo for válido;
- Os métodos `getNome()` e `getCPF()` retornam, respectivamente, o nome e o CPF do cliente.

Escreva um programa que receba, de acordo com as especificações abaixo, nomes e CPFs de clientes. Seu programa deve criar um objeto cliente quando ler seu nome e atribuir o CPF ao último objeto cliente criado quando ler um CPF. Após ler o último CPF, imprimir nome e CPF do cliente.

Para validar um CPF, você pode utilizar o código abaixo. Leia-o com atenção e veja se consegue compreender como um CPF é validado. Observe também que o método `charAt()` pode ser útil para verificar os símbolos "+" e "." na leitura (vide especificação).

```
private static boolean verificarCPF(String cpf) {
   // Nao verifica nulos e valores vazios.
   if (cpf == null || cpf.length() == 0) return false;

   // Considera apenas os digitos do CPF, colocando-os num string builder.
   StringBuilder builder = new StringBuilder();
   for (int i = 0; i < cpf.length(); i++) {
      char c = cpf.charAt(i);
      if (c >= '0' && c <= '9') builder.append(c);
   }

   // Um CPF valido tem 11 digitos.
   if (builder.length() != 11) return false;

   // Calcula o primeiro digito verificador e confere com o digito fornecido.
   int digito = calcularDigitoVerificador(builder, 9);
   if (digito != builder.charAt(9) - '0') return false;

   // Faz o mesmo com o segundo digito verificador.
   digito = calcularDigitoVerificador(builder, 10);
```

```
    if (digito != builder.charAt(10) - '0') return false;

    // Se tudo deu certo, retorna OK.
    return true;
}

private static int calcularDigitoVerificador(StringBuilder builder, int indice) {
    int soma = 0, peso = indice + 1, digito;
    for (int i = 0; i < indice; i++, peso--) {
        digito = builder.charAt(i) - '0';
        soma += digito * peso;
    }
    digito = 11 - (soma % 11);
    if (digito > 9) digito = 0;
    return digito;
}
```

Especificações	
Entrada:	Uma linha contendo uma string representando o nome do cliente, seguida de uma ou mais linhas contendo uma string representando um CPF a ser atribuído ao cliente, seguido de uma linha com sinal de "+" caso tenha outros clientes a ler ou uma linha com um sinal de "." caso não tenha mais dados a ler.
Saída:	Nome e CPF dos clientes, separados por espaço. No caso do cliente não possuir CPF, apenas um espaço deve ser impresso ao lado do seu nome.
Exemplo de entrada 1:	Fulano 123.456.789-09 123.456.789-10 + Beltrano 111.222.333-44 111.111.111-11 222.222.222-22 + Cicrano 141.592.653-59 141.592.653-08 .
Exemplo de saída 1:	Fulano 123.456.789-09 Beltrano 222.222.222-22 Cicrano 141.592.653-08
Exemplo de entrada 2:	Eleanor 010.919.201-09 + Chidi 201.619.092-21 201.622.093-72 +

	Tahani 106.101.316-20 106.101.316-21 106.101.316-22 106.101.316-23 106.101.316-24 106.101.316-25 106.101.316-26 106.101.316-27 106.101.316-28 106.101.316-29 + Janet + Jason 000.000.000-00 .
Exemplo de saída 2:	Eleanor 010.919.201-09 Chidi 201.619.092-21 Tahani 106.101.316-21 Janet Jason 000.000.000-00