

Exercícios de Programação Orientada a Objetos com Java

(Baseado no material do Prof. Vítor Souza)

Exercício :: J2_01

Implemente uma classe chamada `Aleatorio` que representa um número aleatório (como aquela utilizada nos slides do curso). Esta classe deve possuir:

- Um objeto da classe `java.util.Random` compartilhado por todas as instâncias da classe. Ao criar este objeto, passe um valor inteiro lido do teclado como parâmetro ao seu construtor;
- Uma constante `VALOR_MAXIMO_DEFAULT` (a ser utilizada somente pela própria classe) que define o valor máximo padrão = 100, caso um não seja especificado no construtor de `Aleatorio`;
- Dois construtores: um sem parâmetros e outro que recebe um valor inteiro como parâmetro, que deve ser utilizado como valor máximo para o número aleatório gerado;
- Um atributo que armazene o número aleatório gerado durante a construção do objeto e um método que retorne este valor.

Crie uma outra classe com um método `main()` que gere e imprima 10 números aleatórios utilizando a classe `Aleatorio`.

Especificações

<i>Entrada:</i>	Um número inteiro (long).
<i>Saída:</i>	Dez números inteiros, na mesma linha (quebra de linha ao final), separados por espaço, gerados por um objeto <code>Random</code> criado (semente) a partir do número inteiro dado como entrada.
<i>Exemplos de entrada:</i>	123 2019 9223372036854775807
<i>Exemplos de saída:</i>	82 50 76 89 95 57 34 37 85 53 20 30 15 21 31 84 77 12 12 26 13 25 79 39 4 38 77 78 65 31

Exercício :: J2_02

Altere o exercício anterior (J2_01), adicionando um método `renovar()` à classe `Aleatorio`, que gera um novo número a partir do objeto `Random` compartilhado e substitui o número gerado pelo construtor. Modifique o método `main()` para criar 5 números aleatórios e imprimi-los 2 vezes: uma logo após sua construção e outra após chamar o método `renovar()`. Como se pode esperar, os resultados devem ser os mesmos do exercício anterior.

Especificações

<i>Entrada:</i>	Um número inteiro (long).
<i>Saída:</i>	Dez números inteiros, na mesma linha (quebra de linha ao final), separados por espaço, gerados por um objeto <code>Random</code> criado (semente) a partir do número inteiro dado como entrada.
<i>Exemplos de entrada:</i>	123 2019 9223372036854775807
<i>Exemplos de saída:</i>	82 50 76 89 95 57 34 37 85 53 20 30 15 21 31 84 77 12 12 26 13 25 79 39 4 38 77 78 65 31

Exercício :: J2_03

Crie uma classe `Cliente` com o seguinte contrato:

- Ao criar um novo cliente, é obrigatório informar o nome;
- 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

<i>Entrada:</i>	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.
<i>Saída:</i>	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.
<i>Exemplo de entrada 1:</i>	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 .
<i>Exemplo de saída 1:</i>	Fulano 123.456.789-09 Beltrano 222.222.222-22 Cicrano 141.592.653-08
<i>Exemplo de entrada 2:</i>	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

	.
<i>Exemplo de saída 2:</i>	Eleanor 010.919.201-09 Chidi 201.619.092-21 Tahani 106.101.316-21 Janet Jason 000.000.000-00