UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
Departamento de Informática e de Estatística

Exercícios – Lista 6 (Arrays unidimensionais – Outros Exercícios)

Disciplina: INE5603 (20072) - Programação Orientada a Objetos I **Turma**: 0138D
Professora: Carla Merkle Westphall

1. Considerando o código da `classe ColecaoInteiros`, disponível no moodle, escreva a implementação de um método para cada um dos casos a seguir:
    a. Considerando a ordem em que os valores estão armazenados no atributo `osValores`, fornecer a posição do primeiro valor negativo, caso exista;
    b. Fornecer a diferença entre o maior e o menor valor;
    c. Fornecer a quantidade de valores positivos que são pares;
    d. Fornecer uma coleção que contenha cópia dos valores maiores que determinado valor;
    e. Transformar em zero todos os valores negativos;
    f. Considerando a ordem de armazenamento dos valores, fazer com que o primeiro valor passe a ser o último, o segundo passe a ser o penúltimo e assim por diante;
    g. Dado um inteiro positivo `k`, verificar se a coleção representada pelo objeto executor contém ou não todos os valores inteiros positivos e menores ou iguais a `k`;
    h. Considerando apenas os valores positivos, fornecer a coleção formada pelos valores primos constantes na coleção representada pelo objeto executor. Esse método deve fazer com que os valores primos sejam retirados da coleção original e inseridos na coleção dos primos;
        - Informações sobre como encontrar os números primos (Crivo de Erastótenes): http://pt.wikipedia.org/wiki/Crivo_de_Erat%C3%B3stenes
    i. Fornecer a coleção formada por cópias dos valores integrantes da coleção representada pelo objeto executor que são múltiplos de `n`, onde `n` é um inteiro positivo;
    j. Eliminar da coleção todos os valores duplicados (repetidos).
2. Para a `classe ColecaoInteiros`, faça a implementação de um método que, ao ser executado por uma instância da classe, gere uma cópia desta, isto é, devolva como resposta um outra instância que seja um clone (cópia) da instância executora.
3. Para a `classe ColecaoInteiros`, faça a implementação de um método que forneça uma cópia da coleção representada pelo objeto executor, onde, considerando a ordem de armazenamento no array `osValores`, os valores devem ser copiados da coleção original e inseridos na nova coleção (cópia) de forma que essa nova coleção fique ordenada em ordem crescente.
4. Considere o `array x` de `n` elementos. Esses elementos podem ser ordenados em ordem crescente seguindo o seguinte algoritmo:
    a. Definir a posição da última troca como `n-1`;
    b. Fazendo `i` variar de 1 até a posição da última troca, comparar x_i com x_{i+1} e caso $x_i > x_{i+1}$ efetuar a troca de x_i com x_{i+1}, guardando a posição onde ocorreu a última troca;
    c. Repetir o passo b) até que não ocorra nenhuma troca.

    Para a `classe ColecaoInteiros`, usando esse algoritmo, escreva a implementação de um método que, considerando a ordem de armazenamento dos valores no array `osValores`, coloque em ordem crescente esse array.

5. Para a `classe ColecaoInteiros`, considerando a ordem de armazenamento no atributo `osValores`, faça a implementação de um método que, ao ser executado por uma instância da classe, insira um novo valor na coleção, caso os seus elementos estejam ordenados em ordem crescente. A inserção deve ser tal que a coleção continue ordenada sem a necessidade de efetuar uma nova ordenação. Caso a coleção não esteja ordenada, o novo valor não deve ser inserido.
6. Considerando o código da `classe Fila`, disponível no moodle, implemente os métodos que estão faltando na classe Fila.
7. Ainda considerando o código da `classe Fila`, escreva uma aplicação que controle o atendimento das pessoas que chegam à determinada agência bancária. Se necessário escreva novos métodos na `classe Fila` ou escreva novas classes. A aplicação deve disponibilizar ao usuário as seguintes opções:
    a. Receber pessoa: essa opção deve fazer com que uma nova pessoa que chegou na agência seja colocada na fila de atendimento;
    b. Atender pessoa: essa opção deve fazer com que uma pessoa seja retirada da fila para atendimento. A aplicação deve mostrar quem é essa pessoa e retira-la da fila;
    c. Listas pessoas da fila de atendimento: essa opção deve fazer com que a aplicação mostre os nomes de todas as pessoas que estão na fila. Os nomes devem ser mostrados por ordem de fila;
    d. Listar pessoas ordem alfabética: essa opção deve fazer com que a aplicação mostre os nomes de todas as pessoas da fila. Os nomes devem ser mostrados em ordem alfabética;
    e. Listar faixa etária: essa opção deve fazer com que a aplicação mostre os nomes das pessoas da fila de determinada faixa etária;
    f. Verificar pessoa: essa opção deve informar se determinada pessoa está ou não na fila de atendimento.
8. Teste o exemplo a seguir. Procure na documentação da API do Java (http://java.sun.com/javase/6/docs/api/java/util/Arrays.html) informações sobre: `classe Arrays` e métodos `fill`, `sort` e `equals` desta classe.

```
import java.util.*;

public class Principal {

    public static void printArray(int[] vet){
           System.out.println("Array: ");
           for (int i = 0; i < vet.length; i++)
           System.out.print(vet[i]+" ");
           System.out.println();
      } // fim printArray

    public static void main(String[] args) {
      int[] a1 = { 10, -1, 3 };
      int[] a2 = new int[3];

      Arrays.fill(a2,5);
      printArray(a2);

      Arrays.sort(a1);
      printArray(a1);

      boolean res = Arrays.equals(a1,a2);
      if ( res )
            System.out.println("Array a1 igual ao array a2");
      else
            System.out.println("Array a1 diferente do array a2");

    }
}
```