Trabalho – 03

Organização de Dados em Vetores usando Interfaces

Rômulo Ferreira Douro

Programação Orientada a Objetos I

Vitória – ES

SUMÁRIO

1 DESCRIÇÃO

O objetivo deste trabalho é desenvolver um sistema computacional para armazenamento otimizado de objetos usando array com a especificação dada através de uma interface em Java.

2 ENTREGA DO TRABALHO

O trabalho deve ser feito em grupos de, no máximo, 5 componentes.

2.1 ENTREGA

A pasta do projeto JAVA (feito no NetBeans) deve ser compactada em um arquivo (ZIP).

Esse arquivo deve ser nomeado seguindo o seguinte critério:

- POO1_171_TR03_nome01_nome02_nome03.zip
  - (para grupos de 3 alunos)
- POO1_171_TR03_nome01_nome02_nome03_nome04.zip
  - (para grupos de 4 alunos)

Onde nome01 (nome02, nome03, nome04) são os nomes dos integrantes do grupo. Por exemplo, para os alunos Paulo Roberto, José Alcantara e Marina Ferreira o nome do arquivo será:

POO1_171_TR03_PauloRoberto_JoseAlcantara_MarinaFerreira.zip

O arquivo deverá ser postado no moodle (http://moodlep.catolica-es.edu.br/moodle) da disciplina por um integrante do grupo (APENAS). Grupos com mais de um envio não serão avaliados!

Os trabalhos só serão avaliados com envios pelo moodle sendo que envios por e-mail não serão aceitos.

A data limite de envio é 01/06/2017 até às 23:55.

2.2 ATENÇÃO

- **Enviem o trabalho no prazo especificado e no formato especificado. Trabalhos recebidos fora do prazo ou em formato inadequado serão penalizados.**
- **Grupos com número diferente de pessoas NÃO serão avaliados.**

3 AVALIAÇÃO DO TRABALHO

O presente trabalho pretende avaliar a capacidade de implementação de algoritmos básicos que é imprescindível a qualquer aluno de programação.

Os códigos dos algoritmos devem ser escritos na linguagem JAVA podendo utilizar a IDE NetBeans em ambiente Windows ou Linux.

Serão quesitos de averiguação para a nota do trabalho:

- **Correta implementação usando JAVA**
- **Saída correta do algoritmo**
- **Pontualidade na entrega**
- **Conformidade com o solicitado no trabalho**

A distribuição de pontos será tal que:

- **Envio correto do trabalho segundo o tópico Entrega – 3 pontos.**
- **Solução corretamente implementada – 50 pontos.**
    - **Correta, sem bugs, implementada conforme especificada.**
- **Organização, endentação e comentários da solução – 7 pontos.**

Organização e endentação do código-fonte bem como comentários adequados.

Dúvidas a respeito do trabalho serão sanadas nas aulas ou via e-mail.

Este trabalho vale no máximo **60 pontos**.

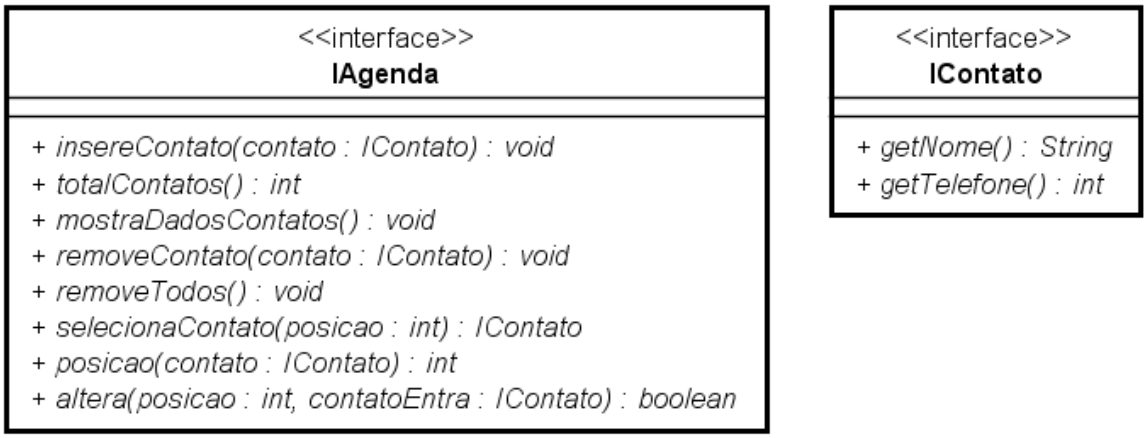
Dúvidas a respeito do trabalho serão sanadas nas aulas ou via e-mail.

4 IMPLEMENTAÇÃO

Leia atentamente o que se pede e gere um programa em JAVA.

4.1 O QUE DESENVOLVER

Cada grupo deverá desenvolver duas classes que implementarão as interfaces fornecidas a seguir.



As letras iniciais são “i” de interface.

A classe que implementar a interface IAgenda servirá como armazenamento de dados para uma agenda de contatos e a classe que implementar a interface IContato será a representação de um contato dessa agenda.

A descrição dos métodos bem como as restrições impostas serão detalhadas a seguir.

4.1.1 Descrição dos métodos da interface

Interface **IAgenda**:

```
public interface IAgenda {
    public void insereContato(IContato contato);
    public int totalContatos();
    public void mostraDadosContatos();
    public void removeContato(IContato contato);
    public void removeTodos();
    public IContato selecionaContato(int posicao);
    public int posicao(IContato contato);
    public boolean altera(int posicao, IContato contatoEntra);
}
```

Interface **IContato**

```
public interface IContato {
    public String getNome();
    public int getTelefone();
}
```

- Métodos a serem desenvolvidas
    - **Método insereContato(IContato contato);**
        - **Recebe um IContato e insere na primeira posição livre**
    - **Método totalContatos();**
        - **Retorna a quantidade total de contatos inseridos na agenda**
    - **Método mostraDadosContatos();**
        - **Exibe os dados dos contatos (nome e telefone)**
    - **Método removeContato(IContato contato);**
        - **Verifica se um objeto passado como parâmetro existe na agenda e em caso positivo remove o objeto. Em caso de haverem objetos repetidos (com mesmo nome e mesmo telefone) deve-se remover a primeira ocorrência do mesmo**
    - **Método removeTodos();**
        - **Deve apagar todos os contatos da agenda**
    - **Método selecionaContato(int posicao);**
        - **Verifica se uma dada posição é válida e em caso positivo retorna o contato que existe naquela posição**
    - **Método posicao(IContato contato);**

- **Verifica se um contato passado como parâmetro existe na agenda e em caso positivo retorna a posição desse contato**

- **Método altera(int posicao, IContato contatoEntra);**
  - **Verifica se uma posição passada como parâmetro é válida e em caso positivo substitui o contato naquela posição pelo objeto contatoEntra passado como parâmetro.**
    - **Em caso positivo retorna true. Em caso de uma posição inválida retorna false.**

Para a interface IContato a sua implementação deverá contar com:

- getNome → para retornar o nome do contato
- getTelefone → para retornar o telefone do contato

Os métodos deverão ser desenvolvidos nas classes **AgendaImplementada (que implementa IAgenda)** e **ContatoImplementado (que implementa IContato)**

Atenção: na classe AgendaImplementada deverá existir um vetor de IContato para armazenar os objetos do tipo ContatoImplementado.

4.2 RESTRIÇÕES PARA IMPLEMENTAÇÃO

Atenção: os detalhes aqui descritos são regras de negócio e precisam ser corretamente implementados. A não implementação ou implementação incorreta acarreta perda de pontos!

Nenhum grupo, em hipótese alguma poderá utilizar estruturas disponíveis pelo Java tais como List e suas descendentes.

Para armazenamento de informações apenas são permitidos o uso de arrays.

4.2.1 Restrições para inserção

Não haverão gets e sets na sua implementação da AGENDA!

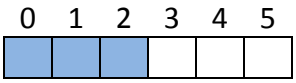
Na classe que você irá implementar a sua agenda apenas serão usados os métodos apresentados na interface IAgenda.

Dica: Você pode usar um array de objetos para armazenar os contatos e um contator do tipo int para averiguar a quantidade de objetos inseridos.

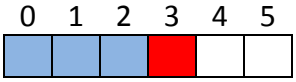
4.2.2 Restrições para inserção

Todos os elementos estarão compactados à esquerda do vetor!

Como os dados estão compactados à esquerda, caso tenhamos uma situação de um vetor parcialmente preenchido, como no exemplo abaixo:



Um dado só pode ser inserido na primeira posição vazia à direita, ou seja, na posição 3.

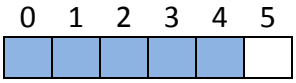


A célula em vermelho indica um elemento recém inserido

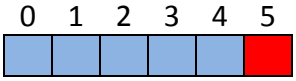
É esperado que a **quantidade de contatos possa crescer indefinidamente** portanto o grupo deve efetuar uma implementação para que, sempre que o vetor de contatos chegue ao seu limite, ou seja, sempre que o total de contatos cadastrados seja igual ao tamanho do vetor o mesmo seja **realocado um novo espaço com o dobro do tamanho atual do vetor** e os dados contidos sejam repassados para o novo vetor.

Por exemplo:

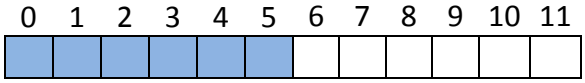
- Situação do vetor



- Situação após a inserção de um novo contato



- Verifica que está cheio e efetua uma nova alocação transportando os elementos atuais para o novo vetor



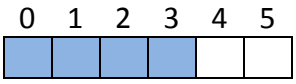
- Dessa forma, sempre que for inserido um novo elemento ao vetor de contatos o mesmo poderá “crescer” para abrigar mais e mais elementos
    - No início o vetor tem capacidade para 1 elemento
    - Após inserir um contato ele terá capacidade para 2
    - Após inserir o segundo contato terá capacidade para 4
    - Após inserir o quarto contato terá então capacidade para 8
    - Após inserir o 8º terá capacidade para 16 e assim sucessivamente

Essa rotina de ampliação do vetor deve ser efetuada ao final da inclusão de um novo contato sempre checando a quantidade total inserida de contatos com a quantidade máxima do vetor (length).

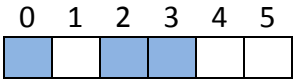
4.2.3 Detalhes para exclusão

No caso da exclusão de um elemento, caso desejássemos remover um elemento da posição 1 do vetor, por exemplo, os elementos à direita (caso existam), devem ser deslocados para a esquerda, dessa forma:

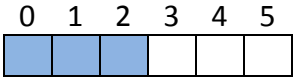
- Situação antes:



- Situação no momento da exclusão:



- Situação após a exclusão:



Posição inválida nos métodos de remoção, seleção ou alteração implica indicar uma posição < 0 ou >= totalDeContatos já que o vetor é indexado dentro desse espaço numérico

4.3 TESTES

São disponibilizadas duas classes de testes. As classes **AgendaImplementada (que implementa IAgenda)** e **ContatoImplementado (que implementa IContato)** se referem à implementação feita pelo professor e são as classes que os alunos deverão implementar (nelas deverão ser construídos os métodos). A seguir são mostrados os códigos para os grupos efetuarem seus testes.

4.3.1 Teste 1

```
public class Teste1 {

    public static void main(String[] args) {
        IAgenda A = new AgendaImplementada();
        A.insereContato(new ContatoImplementado("pai", 123));
        A.insereContato(new ContatoImplementado("mae", 456));
        A.insereContato(new ContatoImplementado("irma", 789));
        System.out.println("Existem " + A.totalContatos() + " contatos inseridos");
        A.mostraDadosContatos();
        System.out.println("-----------------");
        A.removeContato(new ContatoImplementado("pai", 123));
        System.out.println("Existem " + A.totalContatos() + " contatos inseridos");
        A.mostraDadosContatos();
        System.out.println("-----------------");
        A.altera(0, new ContatoImplementado("pai", 123));
        System.out.println("Existem " + A.totalContatos() + " contatos inseridos");
        A.mostraDadosContatos();
        System.out.println("-----------------");
        A.altera(1, new ContatoImplementado("mae", 456));
        System.out.println("Existem " + A.totalContatos() + " contatos inseridos");
        A.mostraDadosContatos();
        System.out.println("-----------------");
        A.removeContato(new ContatoImplementado("pai", 123));
        System.out.println("Existem " + A.totalContatos() + " contatos inseridos");
        A.mostraDadosContatos();
        System.out.println("-----------------");
        System.out.println(A.posicao(new ContatoImplementado("mae", 456)));
```

```
        System.out.println("-----------------");
        System.out.println(A.posicao(new ContatoImplementado("pai", 123)));
        System.out.println("-----------------");
        System.out.println(A.altera(0, new ContatoImplementado("pai", 123)));
        System.out.println("-----------------");
        System.out.println(A.selecionaContato(0));
        System.out.println("-----------------");
        A.removeTodos();
        System.out.println("Existem " + A.totalContatos() + " contatos inseridos");
    }
}
```

Saída

```
run:

Existem 3 contatos inseridos

Posição 0 = pai tel: 123

Posição 1 = mae tel: 456

Posição 2 = irma tel: 789

-----------------

Existem 2 contatos inseridos

Posição 0 = mae tel: 456

Posição 1 = irma tel: 789

-----------------

Existem 2 contatos inseridos

Posição 0 = pai tel: 123

Posição 1 = irma tel: 789

-----------------

Existem 2 contatos inseridos

Posição 0 = pai tel: 123

Posição 1 = mae tel: 456
```

```
----------

Existem 1 contatos inseridos

Posição 0 = mae tel: 456

----------

0

----------

-1

----------

true

----------

pai tel: 123

----------

Existem 0 contatos inseridos

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

4.3.2 Teste 2

```
public class Teste2 {

    public static void main(String[] args) {
        IAgenda A = new AgendaImplementada();
        IContato c;
        long tempo = System.currentTimeMillis();
        for (int i = 0; i < 1000; i++) {
            c = new ContatoImplementado("nome " + i, i);
            A.insereContato(c);
        }
        tempo = System.currentTimeMillis() - tempo;
        System.out.println("Total de Registros = " + A.totalContatos());
        A.mostraDadosContatos();
        System.out.println("Inseridos em " + tempo + " milisegundo(s)");

    }
}
```

Saída

```
run:

Total de Registros = 1000

Posição 0 = nome 0 tel: 0

Posição 1 = nome 1 tel: 1

Posição 2 = nome 2 tel: 2

...

Posição 998 = nome 998 tel: 998

Posição 999 = nome 999 tel: 999

Inseridos em 2 milisegundo(s)

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Bom trabalho!