Escola de Artes, Ciências e Humanidades – EACH-USP

ACH2001 – Introdução à Ciência da Computação
EACH – Primeiro Semestre de 2008

Terceiro Exercício-Programa — Data de entrega: até 23 de junho de 2008.

Professor: Delano Medeiros Beder

CADASTRO DE CLIENTES

1 Introdução

Neste exercício, vocês criarão um pequeno sistema de cadastro de clientes para uma loja de São Paulo. Será um sistema simples, contendo apenas nome e telefone dos clientes, mas que ajudará bastante o gerenciamento dos clientes da loja e que poderá ser incrementado futuramente.

1.1 Cadastro de Clientes

Um sistema de cadastro deve permitir que sejam adicionados novos clientes e que as fichas de clientes já existentes sejam encontradas a partir do nome do cliente. A ficha do cliente conterá apenas dois campos, e será representado pela classe `Cliente`.

```
class Cliente {
    String nome;
    String telefone;
}
```

Já o sistema de cadastro deve ser implementado por uma classe de nome `CadastroDeClientes`. Esta classe deve possuir um atributo `Cliente[] listaDeClientes`, que mantém a lista dos clientes cadastrados. Ela deve ser inicializada com tamanho 10, para permitir a inserção de até 10 clientes.
A classe `CadastroDeClientes` deve possuir **obrigatoriamente**, os seguintes métodos:

- `boolean adicionaCliente(String nome, String telefone)`: Adiciona o cadastro de um novo Cliente no final da lista de clientes. Se o novo cliente for adicionado com sucesso, o método devolve `true`. Caso contrário, por exemplo se a lista de clientes já estiver cheia, devolve `false`.
- `String obtemTelefone(String nome)`: Busca o cadastro de um cliente utilizando o nome do cliente e devolve o telefone do cliente. Como esta busca é realizada com freqüência, ela deve ser eficiente. Para tal, você deve **obrigatoriamente** realizar uma *busca binária* para encontrar o cadastro do cliente. Se o cliente não for encontrado, o método deverá devolver uma `String` vazia.
- `void ordenaLista()`: Ordena a lista de clientes colocando seus nomes em ordem alfabética. Você pode utilizar os algoritmos de inserção direta, seleção direta ou o método da bolha. Além disso, sempre que este método for chamado, ele deverá imprimir na tela "Ordenando cadastros.".
- `void juntaCadastros(CadastroDeClientes cadastros)`: funcionamento explicado a seguir.

Lembre-se que para realizar uma busca binária, é preciso que a lista de clientes esteja ordenada. Portanto, antes de realizar uma busca, é preciso verificar se a lista de clientes está ordenada. Se não estiver, o método de busca binária deve-se ordená-la primeiro.

Observação: Para determinar se a lista está ordenada, utilize uma variável booleana `listaOrdenada`. Deste modo, o método `ordenaLista` pode determinar rapidamente se a lista está ordenada ou não. Além disso, só chame o método `ordenaLista` se for realmente necessário, pois a ordenação é bastante demorada!

1.2 Suporte a um número arbitrário de clientes

Um problema importante com a classe classe `CadastroDeClientes` acima é que esta permite que sejam inseridos apenas 10 clientes. Isto porque o tamanho do *array* é 10.

Uma solução possível é criar um novo objeto `CadastroDeClientes` e adicionar os novos clientes a este objeto. Em seguida, podemos juntar os cadastros dos dois objetos utilizando o método de fusão de seqüências. Para tal, implemente o seguinte método:

- `void juntaCadastros(CadastroDeClientes cadastros)`: recebe um objeto do tipo `CadastroDeClientes` e realiza a fusão da lista de cadastros do objeto `cadastros` recebido com a lista local de cadastros. Nesta fusão, o seu programa deverá criar um novo array que comporte todos os cadastros.

Lembre-se que antes de realizar a fusão das seqüências, é preciso verificar se estas estão ordenadas e ordená-las se necessário. Para tal, utilize as variáveis booleanas `listaOrdenada` de cada objeto.

1.3 Classe de Testes

Para ajudar na determinação do funcionamento de seu programa, será disponibilizada uma classe de testes `TestaCadastro` com o método `realizaTestes`. Este método realiza diversas operações com os métodos de `CadastroDeClientes` para verificar se estes estão funcionando corretamente.

Atenção: O fato de sua classe funcionar para estes testes não significa que ela está totalmente correta, apenas que para os testes realizados ela funciona. Os monitores realizarão testes diferentes daqueles da classe de teste fornecida.

2 Observações importantes

2.1 Sobre a elaboração:

- É **obrigatório** que a classe de testes funcione corretamente com o seu EP!
- Você deve utilizar **apenas** os conceitos apresentados em aula.
- Este exercício-programa deve ser elaborado individualmente.
- Guarde uma cópia do seu programa entregue.
- Coloque todas as classes em um arquivo `Cadastro<nusp>.java`

  Exemplo: `Cadastro1234567.java`

2.2 Sobre a avaliação:

- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO. O exercício do aluno alvo da cópia também receberá nota ZERO.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO.
- É muito importante que seu programa tenha comentários e esteja bem identado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A qualidade do seu trabalho sob esse ponto de vista influenciará sua nota!
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente apresentado estiver o seu programa, melhor será a disposição dele para dar-lhe uma nota generosa.

2.3 Sobre a entrega:

- O prazo de entrega é o dia 23 de junho de 2008.
- No início do arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*************************************************************************/
/**   ACH 2001 - Introdução à Ciência da Computação I                   **/
/**   EACH-USP - Primeiro Semestre de 2008                              **/
/**   <turma> - <nome do professor>                                     **/
/**                                                                     **/
/**   Terceiro Exercício-Programa                                       **/
/**   Arquivo: <nome do arquivo>                                        **/
/**                                                                     **/
/**   <nome do(a) aluno(a)>                <número USP>                 **/
/**                                                                     **/
/**   <data de entrega>                                                 **/
/*************************************************************************/
```

  Não é obrigatório que o cabeçalho seja idêntico a esse, apenas que contenha pelo menos as mesmas informações.

- A entrega será feita unicamente pelo CoL. Não serão aceitos trabalhos enviados por email. Fiquem atentos, pois o CoL agora possui horário limite para entrega de trabalhos.
- Coloque todas as classes em um arquivo `Cadastro<nusp>.java` e entregue somente este arquivo pelo CoL. (Cuidado para não enviar o arquivo errado!)