Programação Orientada a Objetos

Gerência de Objetos por Outro Objeto

Programação de Computadores II

Professor: Edwar Saliba Júnior

Gerenciamento de Clientes

Neste exemplo, veremos de forma bem simples como gerenciar objetos através de outros objetos.

Contextualização

Num ambiente de trabalho qualquer, deseja-se informatizar o cadastro de clientes que até então é feito via fichas de papel. E estas por sua vez são organizadas em um dispositivo conhecido como fichário.

Para darmos prosseguimento ao nosso exemplo, levaremos em consideração que o computador utilizado para esta informatização nunca será desligado e também nunca terá nenhum problema de *hardware* ou de *software* (a máquina perfeita!), pois, não salvaremos os dados em memória secundária.

Para desenvolvermos este projeto devemos entender como se dá o funcionamento da empresa em relação ao seu cadastro de clientes, ou seja, como todo o processo é feito sem o computador.

Após compreendermos como se dá o processo de cadastro, alteração, exclusão e consulta de clientes, então poderemos propor algo que seja adequado as necessidades reais da empresa.

Em princípio, podemos pensar nos clientes como sendo, cada qual, uma ficha de papel cheia de campos preenchidos com os seus dados pessoais. Pensando desta maneira, podemos dizer que o gerenciamento de clientes se dá da seguinte forma:

- se quisermos cadastrar um novo cliente, então deveremos pegar uma destas fichas de papel em branco (sem estar preenchida) e anotar os dados do novo cliente nos diversos campos que compõem a ficha de cadastro (Exemplo: Nome, endereço, identidade, telefone, CPF e etc.);
- após preenchidos os dados do novo cliente em uma nova ficha, esta vai para o fichário, que é organizado segundo uma convenção qualquer, ou seja, ordem alfabética, ordem do número de matrícula dos clientes e etc.;
- se precisarmos alterar os dados de um cliente, então devemos buscar a ficha deste no fichário. Poderemos encontrá-la facilmente através de um dos mecanismos de organização deste fichário e alterar os campos necessários;
- por último, se precisarmos excluir um cliente por um motivo qualquer, devemos retirar do fichário a ficha deste cliente, rasgá-la e jogá-la no lixo.

Observações:

- esta primeira versão do software fará uso de vetor para o gerenciamento de clientes;
- construiremos uma classe para representar a ficha de cadastro (que chamaremos de

Cliente), outra para representar o dispositivo de armazenamento das fichas, o fichário (que chamaremos de GerenciaClientes) e outra que será nossa classe principal, onde as classes se tornarão objetos e estes serão manipulados de acordo com as necessidades do usuário do *software*.

Agora, chega de blá... blá... blá... e vamos ao código-fonte:

Classe Cliente

```java
package gerenciamentodeclientes;

public class Cliente {
    private int identificador;
    private String nome;
    private String endereco;
    private String identidade;
    private String cpf;

    public Cliente(){
        this.identificador = 0;
        this.nome = "";
        this.endereco = "";
        this.identidade = "";
        this.cpf = "";
    }

    public Cliente(int identificador, String nome,
            String endereco, String identidade, String cpf){
        this.identificador = identificador;
        this.nome = nome;
        this.endereco = endereco;
        this.identidade = identidade;
        this.cpf = cpf;
    }

    public void alterar(int identificador, String nome,
            String endereco, String identidade, String cpf){
        this.identificador = identificador;
        this.nome = nome;
        this.endereco = endereco;
        this.identidade = identidade;
        this.cpf = cpf;
    }

    public void consultar(){
        System.out.println("Identificador: " + identificador);
        System.out.println("Nome: " + nome);
        System.out.println("Endereço: " + endereco);
        System.out.println("Identidade: " + identidade);
        System.out.println("CPF: " + cpf);
    }
}
```

Figura 1: Classe Cliente

Classe GerenciaClientes

```java
package gerenciamentodeclientes;

import java.util.Scanner;

public class GerenciaClientes {
    private final int qtdadeClientes = 100;

    private Cliente fichario[]; // Vetor para armazenar Clientes.
    private int posicao;  // Contador para a última posição preenchida.
    private Scanner e;

    public GerenciaClientes(){
        fichario = new Cliente[qtdadeClientes];
        posicao = -1;
        e = new Scanner(System.in);
    }

    public void cadastrarCliente(){
        String nome, endereco, identidade, cpf;

        posicao++;

        System.out.println("Digite nome: ");
        nome = e.nextLine();
        System.out.println("Digite endereço: ");
        endereco = e.nextLine();
        System.out.println("Digite identidade: ");
        identidade = e.nextLine();
        System.out.println("Digite CPF: ");
        cpf = e.nextLine();

        fichario[posicao] = new Cliente(posicao,
                nome, endereco, identidade, cpf);
    }

    public void alterarCliente(){
        String nome, endereco, identidade, cpf;
        int cod = -1;

```

Figura 2: Classe GerenciaClientes - Parte 1/2

```
40          System.out.println("Informe a posição do vetor a ser alterada: ");
41          cod = e.nextInt();
42          System.out.println("Digite nome: ");
43          nome = e.nextLine();
44          System.out.println("Digite endereço: ");
45          endereco = e.nextLine();
46          System.out.println("Digite identidade: ");
47          identidade = e.nextLine();
48          System.out.println("Digite CPF: ");
49          cpf = e.nextLine();
50
51          fichario[cod].alterar(cod, nome, endereco, identidade, cpf);
52      }
53
54      public void excluirCliente(){
55          int cod = -1;
56
57          System.out.println("Informe a posição do vetor a ser excluída: ");
58          cod = e.nextInt();
59
60          fichario[cod] = null;
61      }
62
63      public void consultarCliente(){
64          int cod = -1;
65
66          System.out.println("Informe a posição do vetor a ser consultada: ");
67          cod = e.nextInt();
68
69          if(fichario[cod] != null)
70              fichario[cod].consultar();
71          else
72              System.out.println("Posição vazia!");
73      }
74
75      public void imprimirClientes(){
76          for(int i = 0; i <= posicao; i++){
77              if(fichario[i] != null){
78                  fichario[i].consultar();
79                  System.out.println("\n");
80              }
81          }
82      }
83  }
```

Figura 3: Classe GerenciaClientes - Parte 2/2

Limitações do Código-fonte Apresentado

Na classe Cliente apresentada na Figura 1, podemos ver claramente que os métodos cadastrar e alterar desempenham funções idênticas, ou seja, caso o usuário do *software* queira alterar um único campo da ficha de cadastro, ele também terá que digitar todos os demais, como se estivesse preenchendo um novo cadastro. Isto não está bom e pode ser melhorado!

Já a classe GerenciaClientes apresentada nas Figuras 2 e 3, como já dito anteriormente, ela funciona como um fichário, ou seja, um lugar para se guardar de forma organizada as fichas dos clientes.

Programadores experientes sabem que usar um vetor, como foi feito na construção desta classe, não é a melhor maneira de se implementar tal estrutura. No entanto, para aqueles que estão começando, pode ser considerada uma ótima maneira de se fazer. Visto que a outra maneira ainda não lhes foi apresentada.

O grande mal de se usar um vetor para o armazenamento das fichas de clientes, doravante tratada por objeto cliente, é que um vetor tem uma quantidade pré-definida de posições de armazenamento, que no caso do código apresentado são apenas 100. Além disto, neste código há uma outra limitação. Se analisarmos minuciosamente o método cadastrarCliente descrito na linha 18 da classe GerenciaClientes, veremos que existe um cadastro sequencial. Contudo, se em algum momento nós excluirmos algum objeto cliente do vetor, a posição excluída ficará vazia porém sem a possibilidade de ser preenchida novamente.

Classe Main

Nossa classe principal, aquela que carrega consigo o método *main*, mostrada na Figura 4 (adiante), está bem simples e bem enxuta. Ou seja, pouquíssima codificação. O motivo para isto é que transferimos boa parte do código que poderia estar nela, para a classe GerenciaClientes. O que vem a ser uma boa prática de programação, visto que o código-fonte fica bem mais organizado, mais fácil de ser entendido, alterado e analisado.

Nesta classe há uma estrutura de *menu* com o intuito de proporcionar as operações que predefinimos no início deste tutorial. Ou seja: inclusão, exclusão, alteração e consulta de clientes.

Exercício

1. Reescreva o *software* para Gerenciamento de Clientes de modo que as limitações apresentadas neste tutorial sejam totalmente corrigidas.

2. Reescreva o *software* para Gerenciamento de Clientes, porém desta vez, eliminando a classe GerenciaClientes, ou seja, o código-fonte que está nesta classe deverá passar para o método *main* da classe que o abriga. Assim que você terminar compare a estrutura deste novo código-fonte com a do exercício anterior e exponha suas conclusões.

   Em ambos os exercícios, o seu código-fonte deverá continuar usando vetor.

```java
package gerenciamentodeclientes;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner e =  new Scanner(System.in);
        int opcao;
        GerenciaClientes clientes =  new GerenciaClientes();

        do{
            System.out.println("Menu de Clientes: " );
            System.out.println("1 - Cadastrar");
            System.out.println("2 - Alterar");
            System.out.println("3 - Excluir");
            System.out.println("4 - Conultar");
            System.out.println("5 - Relatório");
            System.out.println("6 - Sair");
            System.out.println("Opção: ");
            opcao = e.nextInt();

            switch(opcao){
                case 1:
                    clientes.cadastrarCliente();
                     break;
                case 2:
                    clientes.alterarCliente();
                     break;
                case 3:
                    clientes.excluirCliente();
                     break;
                case 4:
                    clientes.consultarCliente();
                     break;
                case 5:
                    clientes.imprimirClientes();
                     break;
                default:
                     if(opcao != 6)
                        System.out.println("Opção inválida! \n");
            }
        }while(opcao != 6);
    }
}
```

Figura 4: Classe Main