Programação de Soluções Computacionais – PSC

Exercícios de Laboratório – Prof. Calvetti

Aula: 09
Coleções, Interface List e ArrayLIst

Exemplo Resolvido: Copie e cole o código abaixo em sua I.D.E., analise o código fonte e execute o mesmo código em sua I.D.E. Avalie se o seu entendimento prévio do código fonte corresponde ao resultado obtido pela execução do programa.

a) Sentindo-se solitário em um sábado à noite, o aluno de computação pensa consigo mesmo "Que bom seria se houvesse um jeito de conversar com meus amigos sem precisar sair do computador!" Então ele resolve criar um software chamado Rede de Amigos, onde você possa adicionar seus amigos e conversar com eles. Para isso ele criou uma classe chamada Amigo, com os atributos nome (String), sexo(String), idade(int) e mensagem(String), que armazena a última mensagem enviada para o amigo. Claro que, sendo bom programador, respeitou o encapsulamento e criou os métodos de acesso e os modificadores.

b) Depois o aluno de computação criou a classe Rede, que contém um ArrayList de amigos. Ele fez um método para adicionar amigos e um para bloquear amigos (remover). Fez um método que encontra um amigo pelo nome, retornando sua posição no arraylist (ou -1 se não achar). E um método para enviar uma mensagem para o amigo, que encontra o amigo pelo nome e altera seu atributo mensagem (a mensagem pode ter no máximo 144 caracteres; se for maior o sistema trunca em avisar nada). Ele fez também um método que retorna um vetor com o(s) amigo(s) mais velho(s).

c) Para testar seu sistema o aluno de computação fez uma classe Teste com o método main que, usando o JOptionPane, possui um loop com as opções 1. add amigo, 2. block amigo, 3. procura amigo, 4. envia mensagens, 5. lista velhos e 6. sair.

```
public class Amigo{
   private String nome, sexo, mensagem;
   private int idade;

   public String getNome(){
      return nome;
   }

   public String getSexo(){
      return sexo;
   }

   public String getMensagem(){
      return mensagem;
   }
```

```
    public int getIdade(){
        return idade;
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public void setSexo(String sexo){
        this.sexo = sexo;
    }

    public void setMensagem(String mensagem){
        //tamanho da mensagem e no maximo 144
        if(mensagem.length() <= 144){
            this.mensagem = mensagem;
        } else {
            //trunca se for maior
            this.mensagem = mensagem.substring(0, 144);
        }
    }

    public void setIdade(int idade){
        this.idade = idade;
    }

    public String toString(){
        return "[Nome: "+nome+"] [Sexo: "+sexo+"] [Idade: "+idade+
            "]\n[Mensagem: "+mensagem+"]";
    }
}
```

```
import java.util.ArrayList;

public class Rede{
    private ArrayList<Amigo> amigos;

    public Rede(){
        amigos = new ArrayList<Amigo>();
    }

    public void addAmigo(Amigo amigo){
        amigos.add(amigo);
    }

    public boolean blockAmigo(String nome){
        int posicao = buscar(nome);

        if(posicao >= 0){
            amigos.remove(posicao);
            return true;
        } else {
            return false;
        }
    }
```

```
public int procurarAmigo(String nome){
   return buscar(nome);
}

public boolean enviarMensagem(String nome, String mensagem){
   int posicao = buscar(nome);

  if(posicao >= 0){
      Amigo amigo = amigos.get(posicao);
      amigo.setMensagem(mensagem);
      return true;
   } else {
      return false;
   }
}

public Amigo[] procurarVelhos(){
   if(amigos.size() == 0){
      return new Amigo[0];
   }
   int maior = amigos.get(0).getIdade();
   //encontrar a maior idade
   for(Amigo amigo:amigos){
      if(amigo.getIdade() > maior){
         maior = amigo.getIdade();
      }
   }
   //contar quantos tem a maior idade
   int qtde = 0;
   for(Amigo amigo:amigos){
      if(amigo.getIdade() == maior){
         qtde++;
      }
   }
   //criar vetor de amigos
   Amigo[] velhos = new Amigo[qtde];
   //popular o vetor de mais velhos
   int k = 0;
   for(int i = 0; i < amigos.size(); i++){
      Amigo amigo = amigos.get(i);
      if(amigo.getIdade() == maior){
         velhos[k++] = amigo;
      }
   }
   return velhos;
}

private int buscar(String nome){
   for(int i = 0; i < amigos.size(); i++){
      Amigo amigo = amigos.get(i);
      String nomeAmigo = amigo.getNome();
      if(nome.equals(nomeAmigo)){
         return i;
      }
   }
   return -1;//nao achou
}
```

```
public void addAmigo(String nome, String sexo, int idade){
      Amigo amigo = new Amigo();
      amigo.setNome(nome);
      amigo.setSexo(sexo);
      amigo.setIdade(idade);
      amigos.add(amigo);
   }

   public void listarAmigos(){
      for(Amigo amigo:amigos){
         System.out.println(amigo);
      }
   }
}
```

```
import javax.swing.JOptionPane;

public class Teste{
   public static void main(String[] args){
      Rede rede = new Rede();
      int menu;
      String nome = null;
      String sexo = null;

      do{
         menu = Integer.parseInt(JOptionPane.showInputDialog(
            "1 add amigo\n2 block amigo\n3 procura amigo"+
            "\n4 envia mensagem\n5 lista velhos\n6 sair"+
            "\n7 listar todos"));

         if(menu == 1){
            nome = JOptionPane.showInputDialog("Nome:");
            sexo = JOptionPane.showInputDialog("Sexo:");
            int idade = Integer.parseInt(
                  JOptionPane.showInputDialog("Idade:"));

            rede.addAmigo(nome, sexo, idade);
         }
         else if(menu == 2){
            nome = JOptionPane.showInputDialog("Nome para remover:");
            if(rede.blockAmigo(nome)){
               JOptionPane.showMessageDialog(null, "Removido");
            }
            else {
               JOptionPane.showMessageDialog(null, "Nao encontrado");
            }
         }
         else if(menu == 3){
            nome = JOptionPane.showInputDialog("Nome para"+
               " procurar:");
            int posicao = rede.procurarAmigo(nome);
            if(posicao >= 0){
               JOptionPane.showMessageDialog(null, "Encontrado em "
                  +posicao);
            }
            else {
               JOptionPane.showMessageDialog(null, "Nao encontrado");
            }
         }
```

```
          else if(menu == 4){
             String mensagem = JOptionPane.showInputDialog(
               "Mensagem:");
             nome = JOptionPane.showInputDialog("Nome para enviar:");
             if(rede.enviarMensagem(nome, mensagem)){
                JOptionPane.showMessageDialog(null,
                   "Mensagem enviada");
             }
             else {
                JOptionPane.showMessageDialog(null,
                   "Nao encontrado");
             }
          }
         else if(menu == 5){
             Amigo[] amigos = rede.procurarVelhos();
             for(int i = 0; i < amigos.length; i++){
                System.out.println(amigos[i]);
             }
          }
          else if(menu == 6){
         }
          else if(menu == 7){
             rede.listarAmigos();
          }
          else {
             JOptionPane.showMessageDialog(null,
                "Opcao invalida");
          }
       }while(menu != 6);

   }
}
```

Problema Proposto: Exercício 1

a) Crie a classe BlocoDeNotas que possui como atributo um ArrayList<String> chamado notas. Crie métodos para inserir, remover e buscar notas. Crie um método que imprima todas as notas.

b) Crie a classe AppBloco, com um método main, e um menu que 1) Insira uma nota, 2) Remova uma nota, 3) Altere uma nota, 4) Listar todas as notas e 5) Saia do sistema.

Problemas Propostos: Exercício 2

a) Você vai gerenciar um depósito e resolveu criar um sistema para isso. Para isso criou uma classe chamada Caixa, com os atributos corredor (String), posicao(int), peso(double) e dono(String), que armazena o nome do dono da caixa. Respeitou o encapsulamento e criou os métodos de acesso e os modificadores.

b) Depois criou a classe Deposito, que contém um ArrayList de caixas. Fez um método para adicionar caixas e um para remover (pelo dono). Fez um método que encontra uma caixa pelo dono, retornando sua posição no arraylist (ou -1 se não achar). E um método para mudar o corredor e a posição de uma caixa, que encontra a caixa pelo dono e altera seu atributos. Ele fez também um método que retorna um vetor com a(s) caixa(s) que pesam mais do que um valor passado por parâmetro.

c) Para testar seu sistema fez uma classe Teste com o método main que, usando o JOptionPane, possui um loop com as opções 1. adiciona caixa, 2. remove caixa, 3. procura caixa, 4. muda caixa, 5. lista mais pesadas que 10.0 e 6. sair.

Problemas Propostos: Exercício 3

a) Crie a classe Cliente com os atributos privados do tipo String nome e fone e com o atributo inteiro id. Crie um construtor que receba valores para os atributos como parâmetros e os métodos de acesso e modificadores.

b) Crie a classe BancoDeClientes com um atributo privado do tipo ArrayList<Cliente> chamado clientes. Crie métodos para inserir um cliente, remover um cliente, alterar um cliente, listar os dados de um cliente e listar os dados de todos os clientes.

c) Crie a classe CadastroApp, com o método main, e que tenha um menu que insira um cliente, remova um cliente, altere um cliente, liste os dados de um cliente e liste os dados de todos os clientes.

Bibliografia

BARNES, David J.; KÖLLING, Michael. **Programação orientada a objetos com java:** uma introdução prática usando o BlueJ . 4. ed. São Paulo: Pearson Prentice Hall, 2009.

DEITEL, P. DEITEL, H. **Java: como programar**. 8 Ed. São Paulo: Prentice – Hall (Pearson), 2010.

LOPES, ANITA. GARCIA, GUTO. **Introdução à Programação: 500 algoritmos resolvidos**. Rio de Janeiro: Elsevier, 2002.