Prática de Programação J2ME (15)

Especialização em Desenvolvimento Web com Interfaces Ricas
www.especializacao.info

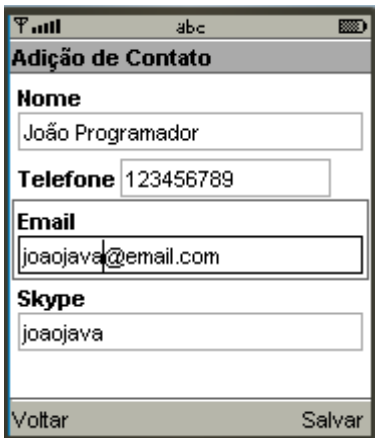
Crie a aplicação correspondente aos itens seguintes. Execute o programa e observe os resultados.

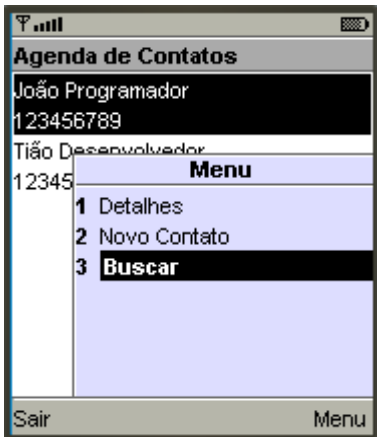
---

1. Crie um MIDlet simples, chamado `PeopleListMidlet.java`, como as práticas anteriores, que ao ser executado, mostra uma lista de contatos pessoais;
2. Os atributos de um contato devem seguir o padrão da Práticas de Programação 14;
3. A lista de contatos pessoais `PessoaList.java` a ser mostrada pelo MIDlet deve ser inicialmente vazia (sem dados),do tipo IMPLICIT, fonte média e plana, com título “Agenda de Contatos”, que permite um item se configurar em mais de uma linha e que possui quatro comandos: Sair, Detalhes, “Novo Contato” e Buscar;
4. A lista de pessoas cadastradas deve ser guardada em um vetor (`Vector – java.util`) em `PessoaList.java;`
5. O comando “Sair” deve estar na posição BACK e fechar a aplicação ao ser clicado;
6. O comando “Detalhes” deve estar na posição “OK”, prioridade 0 e, ao selecionado, mostra-se os detalhes do contato em um novo formulário (como na Prática de Programação 7);
7. O comando “Novo contato” deve estar na posição “OK”, prioridade 1 e, a selecionado, mostra um formulário para entrada de dados de uma nova pessoa a ser cadastrada;
8. Ao selecionar o comando “Buscar”, presente em `PessoaList.java`, *abre-se um novo formulário, chamado FiltraForm.java;*
9. O formulário `NovoContatoForm.java` deve conter quatro TextFields,sendo eles:
   - Nome – aceita até 40 caracteres quaisquer;
   - Telefone – aceita até 10 caracteres numéricos;
   - Email – aceita até 30 caracteres do tipo EMAILADDR;
   - Skype – aceita até 40 caracteres quaisquer.
10. Tal formulário de cadastro de pessoas deve conter dois comandos: Voltar e Salvar.
11. Ao selecionar o comando “Voltar”, a tela do aparelho deve retornar à lista de contatos cadastrados;
12. Ao selecionar “Salvar”, cria-se uma nova Pessoa, que é armazenada no vetor de contatos de `PessoaList.java`;
13. Após adição da pessoa no vetor de cadastrados, a tela da lista é atualizada e mostrada ao usuário. Novas pessoas podem ser cadastradas;
14. *O formulário `FiltraForm.java` contém, inicialmente, um `TextField` para busca de nomes e vários `StringItems`, cada um relativo a um contato existente.*
15. *Ao escrever qualquer caracter no TextField de busca, a lista de contatos filtrados deve ser rearranjada de forma a mostrar somente os nomes dos indivíduos que começam com a “subString” presente no buscador;*
16. *O usuário pode navegar nos nomes dos indivíduos filtrados, cada um contendo um comando “Detalhes”, que mostra os detalhes do contato selecionado.*

Resposta da Prática de Programação J2ME (15)

TELA(S)





CÓDIGO EM JAVA

Pessoa.java

```
package PP15;

public class Pessoa {

    String nome, telefone, email, nomeSkype;

    public Pessoa(String nome, String telefone, String email, String
                      nomeSkype) {
        setNome(nome);
        setTelefone(telefone);
        setEmail(email);
        setNomeSkype(nomeSkype);
    }


    public String getNome() {
        return nome;
    }

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

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

     public String getNomeSkype() {
        return nomeSkype;
    }

    public void setNomeSkype(String nomeSkype) {
        this.nomeSkype = nomeSkype;
    }
}
```

PessoaList.java

```
package PP15;

import java.util.Vector;
import javax.microedition.lcdui.*;

public class PessoaList extends List implements CommandListener, Runnable
{

    private PeopleListMidlet controlador;
    private Vector pessoas;
    private Command info,  exit,  add, buscar;

    public PessoaList(PeopleListMidlet controlador) {
        super("Agenda de Contatos", List.IMPLICIT);
        this.controlador = controlador;
        this.pessoas = new Vector();
        setFitPolicy(List.TEXT_WRAP_ON);
        carregaLista();
        Font fonte = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN,
                Font.SIZE_MEDIUM);
        for (int i = 0; i < size(); i++) {
            setFont(i, fonte);
        }
        exit = new Command("Sair", Command.EXIT, 0);
        info = new Command("Detalhes", Command.OK, 0);
        add = new Command("Novo Contato", Command.OK, 1);
        buscar = new Command("Buscar", Command.OK, 2);
        addCommand(exit);
        addCommand(buscar);
        addCommand(add);
        addCommand(info);
        setCommandListener(this);
    }

    public void commandAction(Command arg0, Displayable arg1) {
        if (arg0 == exit) {
            controlador.destroyApp(true);
        } else {
            if (arg0 == info) {
                if (size() > 0) {
                    new Thread(this).start();
                }
            } else {
                if (arg0 == add) {
                    NovoContatoForm novo = new NovoContatoForm(this);
                    controlador.mudarTelaPara(novo);
                }else{
                    if(arg0 == buscar){
                        controlador.mudarTelaPara(
                           new FiltraForm(pessoas,this)
```

```
                        );
                    }
                }
            }
        }
    }

    public void run() {
        Pessoa selecionado = (Pessoa)
        pessoas.elementAt(getSelectedIndex());
        Form f = new ExibeContatoForm(selecionado, this);
        controlador.mudarTelaPara(f);
    }

    public PeopleListMidlet getControlador() {
        return controlador;
    }

    public void carregaLista() {
        deleteAll();
        for (int i = 0; i < pessoas.size(); i++) {
            Pessoa pessoa = (Pessoa) pessoas.elementAt(i);
            append(pessoa.getNome() + "\n" + pessoa.getTelefone(), null);
        }
    }

    public Vector getPessoas() {
        return pessoas;
    }
}
```

ExibeContatoForm.java

```
package PP15;

import javax.microedition.lcdui.*;

public class ExibeContatoForm extends Form implements CommandListener{

    private Command sair, voltar;
    private PeopleListMidlet controlador;
    private PessoaList telaAnterior;
    private StringItem nome, tel, email, skype;

    public ExibeContatoForm(Pessoa selecionado, PessoaList uiLista){
        super("Informação do Contato");
        this.telaAnterior = uiLista;
        this.controlador = uiLista.getControlador();
        sair = new Command("Sair", Command.EXIT, 0);
        voltar = new Command("Voltar", Command.BACK, 1);
        nome = new StringItem("Nome", selecionado.getNome());
        tel = new StringItem("Telefone", selecionado.getTelefone());
        email = new StringItem("Email", selecionado.getEmail());
```

```
        skype = new StringItem("Skype", selecionado.getNomeSkype());
        append(nome);
        append(tel);
        append(email);
        append(skype);
        addCommand(sair);
        addCommand(voltar);
        setCommandListener(this);
    }
    public void commandAction(Command arg0, Displayable arg1) {
        if(arg0 == sair)
            controlador.destroyApp(true);
        else
            if(arg0 == voltar)
                controlador.mudarTelaPara(telaAnterior);
    }
}
```

NovoContatoForm.java

```
package PP15;

import java.util.Vector;
import javax.microedition.lcdui.*;

public class NovoContatoForm extends Form
     implements CommandListener, Runnable{

    TextField nome, email, telefone, skype;
    Command salvar, voltar;
    PeopleListMidlet controlador;
    PessoaList telaAnterior;

    public NovoContatoForm(PessoaList anterior) {
        super("Adição de Contato");
        this.telaAnterior = anterior;
        controlador = anterior.getControlador();
        nome = new TextField("Nome", "", 40, TextField.ANY);
        email = new TextField("Email", "", 30, TextField.EMAILADDR);
        telefone = new TextField("Telefone", "", 10, TextField.DECIMAL);
        skype = new TextField("Skype", "", 40, TextField.ANY);
        salvar = new Command("Salvar", Command.OK, 0);
        voltar = new Command("Voltar", Command.BACK, 0);
        addCommand(salvar);
        addCommand(voltar);
        append(nome);
        append(telefone);
        append(email);
        append(skype);
        this.setCommandListener(this);
    }
```

```
    public void commandAction(Command arg0, Displayable arg1) {
        if(arg0 == voltar){
            controlador.mudarTelaPara(telaAnterior);
        }else{
            new Thread(this).start();
        }
    }


    public void run() {
        String n = nome.getString();
        String t = telefone.getString();
        String e = email.getString();
        String s = skype.getString();
        Pessoa pessoa = new Pessoa(n, t, e, s);
        Vector pessoas = telaAnterior.getPessoas();
        pessoas.addElement(pessoa);
        telaAnterior.carregaLista();
        controlador.mudarTelaPara(telaAnterior);
    }
}
```

FiltraForm.java

```
package PP15;

import java.util.Vector;
import javax.microedition.lcdui.*;;

public class FiltraForm extends Form implements CommandListener,
        ItemCommandListener, ItemStateListener, Runnable{

    Command voltar, detalhes;
    PessoaList telaAnterior;
    PeopleListMidlet controlador;
    TextField filtrador;
    Vector contatos, contatosFiltrados;

    public FiltraForm(Vector contatos, PessoaList telaAnterior){
        super("Filtrador de Contatos");
        this.telaAnterior = telaAnterior;
        this.contatos = contatos;
        this.controlador = telaAnterior.getControlador();
        contatosFiltrados = contatos;
        filtrador = new TextField("Nome", "", 40, TextField.ANY);
        voltar = new Command("Voltar", Command.BACK, 0);
        detalhes = new Command("Detalhes", Command.BACK, 0);
        addCommand(voltar);
        append(filtrador);
        adicionaFiltrados();
        setItemStateListener(this);
        setCommandListener(this);
    }
```

```
    public void commandAction(Command arg0, Displayable arg1) {
        if(arg0 == voltar)
            controlador.mudarTelaPara(telaAnterior);
    }

    public void commandAction(Command arg0, Item arg1) {
        if(arg0 == detalhes){
            Pessoa p  = getPessoa(arg1.getLabel());
            ExibeContatoForm e = new ExibeContatoForm(p, telaAnterior);
            controlador.mudarTelaPara(e);
        }
    }

    public void itemStateChanged(Item arg0) {
        if(arg0 == filtrador)
            new Thread(this).start();
    }
    public void apagaFiltrados(){
        contatosFiltrados = new Vector();
        deleteAll();
        append(filtrador);
    }

    public void run() {
        apagaFiltrados();
        for(int i=0; i < contatos.size(); i ++){
            Pessoa pessoa = (Pessoa) contatos.elementAt(i);
            if(pessoa.getNome().startsWith(filtrador.getString())){
                contatosFiltrados.addElement(pessoa);
            }
        }
        adicionaFiltrados();
    }

    public Pessoa getPessoa(String nome){
        for(int i= 0; i < contatosFiltrados.size(); i++){
            Pessoa pessoa = (Pessoa) contatosFiltrados.elementAt(i);
            if(pessoa.getNome().equals(nome))
                return pessoa;
        }
        return null;
    }

    private void adicionaFiltrados() {
        for(int i=0; i < contatosFiltrados.size(); i++){
            Pessoa p = (Pessoa) contatosFiltrados.elementAt(i);
            StringItem resumo = new StringItem(p.getNome(), "");
            append(resumo);
            resumo.addCommand(detalhes);
            resumo.setItemCommandListener(this);
        }
    }
}
```

PeopleListMidlet.java

```
package PP15;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class PeopleListMidlet extends MIDlet {

    Display display;

    public void startApp() {
        display = Display.getDisplay(this);
        PessoaList lista = new PessoaList(this);
        display.setCurrent(lista);
    }
    public void pauseApp() {}

    public void destroyApp(boolean unconditional) {
        notifyDestroyed();
    }

    public void mudarTelaPara(Displayable d){
        display.setCurrent(d);
    }
}
```