

VETORES DINÂMICOS COM CLASSES

O projeto restaurante será usado como exemplo de controle de um sistema de cadastro dinâmico de vetores, onde a matriz não tem um tamanho definido.

Crie o projeto restaurante conforme abaixo:

The screenshot shows the 'Nome e Localização' (Name and Location) dialog box in NetBeans. The 'Nome do Projeto' (Project Name) field is filled with 'prjRestaurante'. The 'Localização do Projeto' (Project Location) field is filled with 'C:\Users\aluno\Documents\NetBeansProjects', and there is a 'Procurar...' (Browse...) button next to it. The 'Pasta do Projeto' (Project Folder) field is filled with 'Jsers\aluno\Documents\NetBeansProjects\prjRestaurante'. Below these fields, there is a checkbox labeled 'Usar Pasta Dedicada para Armazenar Bibliotecas' (Use Dedicated Folder for Storing Libraries), which is currently unchecked. Next to it is a 'Pasta Bibliotecas' (Libraries Folder) field and another 'Procurar...' button. A note below this section states: 'Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).' (Different users and projects can share the same compilation libraries (consult the Help for details)). At the bottom, there is a checked checkbox labeled 'Criar Classe Principal' (Create Main Class), and next to it is a text field containing the name 'Principal'.

Acrescenta ao projeto a classe Item, que irá conter os produtos vendidos em nosso restaurante, desde pratos a bebidas servidas.

The screenshot shows the NetBeans IDE with two tabs open: 'Principal.java' and 'Item.java'. The 'Código-Fonte' (Source) tab is selected for 'Item.java'. The code in the editor is as follows:

```
1  /*
2   * To change this template, choose the tool option 'Create
3   * and open the template in the editor'.
4   */
5
6  /**
7   *
8   * @author aluno
9   */
10 public class Item {
11
12 }
```

Acrescente as propriedades:

```
public class Item {

    private int cod;
    private String descricao;
    private double preco;
    private double total;

}
```

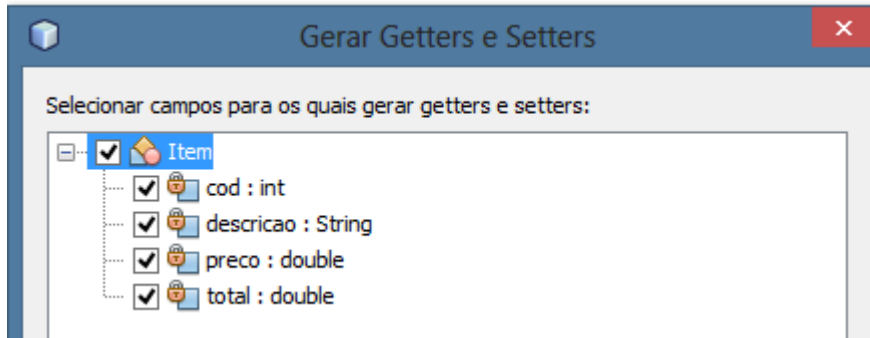
Acrescente o construtor:

```

public Item(int cod, String descricao, double preco) {
    this.cod = cod;
    this.descricao = descricao;
    this.preco = preco;
}

```

E acrescente os getters e Setters para cada propriedade:



Acrescente a classe o método que vai calcular o valor total do item na mesa, por exemplo, precisamos saber quantas cervejas foram consumidas, sabendo a quantidade pedida, devolvemos o valor total pelo preço unitário. Portanto, acrescente o seguinte método a classe Item:

```

public void consumo(int qtd) {
    // retorna a quantidade consumida pelo preço unitário
    this.setTotal(this.getPreco() * qtd);
}

```

Codificaremos agora a classe Comanda:

```

public class Comanda {
}

```

Acrescente as propriedades:

```

public class Comanda {

    private ArrayList<Item> pedidos;
    private double total;

}

```

Codifique o construtor:

```

~
public class Comanda {

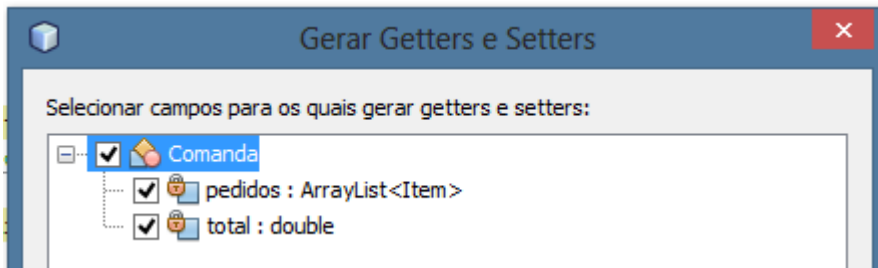
    private ArrayList<Item> pedidos;
    private double total;

    public Comanda(ArrayList<Item> pedidos, double total) {
        this.pedidos = pedidos;
        this.total = total;
    }
}

```



E em seguida os getters e Setters:



Em seguida nós teremos que codificar os pedidos na comanda. O atendente tem que anotar (Adicionar) ou remover o pedido da comanda:

```
public void adicionar ( Item i ){
    // adiciona um pedido se o mesmo não é nulo
    if(i != null){
        pedidos.add(i);
    }
}

public void remover ( Item i ){
    // remove um pedido se o mesmo não é nulo
    if(i != null){
        pedidos.remove(i);
    }
}
```

Codificar também o método que fecha a mesa e devolve o valor pago:

```
public void fechar(){
    // calcula o valor gasto pela mesa
    for(Item p: pedidos){
        this.total+=p.getTotal();
    }
}
```

Vamos testar na classe Principal a lógica das nossas classes:

```

public class Principal {

    // para teste apenas uma mesa

    static Comanda mesa;

    public static void main(String[] args) {

        // simulando o garçon
        // abre a mesa vazia, sem pedidos
        mesa = new Comanda(new ArrayList<Item>(), 0);
        //define os pedidos
        Item p1 = new Item(1, "CERVEJA ANTARTICA",5);
        p1.consumo(3); // 3 cervejas

        Item p2 = new Item(1, "PORCAO BATATA FRITAS",10);
        p2.consumo(2); // duas porções

        mesa.adicionar(p1); //adiciona pedido 1 a mesa
        mesa.adicionar(p2); //adiciona pedido 2 a mesa

        // fecha a mesa
        mesa.fechar();

        JOptionPane.showMessageDialog (null, "Mesa:\n"
            + "Total a Pagar: R$" +mesa.getTotal());

    }
}

```

Ao executar teremos o seguinte resultado:



A próxima etapa é ampliar esta lógica para controlar um sistema completo de mesas, onde cada mesa tem a sua própria comanda onde serão anotados os pedidos de cada mesa. Em nosso caso uma vez a mesa fechada, ela será removida da memória para dar lugar a outra comanda. Podemos trabalhar com um sistema de mesas fixa, por exemplo, um restaurante terá digamos 30 mesas, cada um com sua comanda, ou um sistema móvel onde a mesa é identificada na comanda, neste caso as mesas contém diversas comandas, em nosso exemplo iremos adotar um sistema com 15 mesas fixas para efeitos de simplicidade do código.

Comece modificando a seguinte linha do seu programa na classe Principal:

```

public class Principal {

    // agora serão várias mesas
    static Comanda[] mesas;

```

Apague o código do método main e troque para o código abaixo:

```
public static void main(String[] args) {

    int op;
    mesas = new Comanda[15];
    do {
        op = menu();
        switch (op) {
            case 1: break;
            case 2: break;
            case 3: break;
            case 4: break;
            case 5: break;
        }
    } while (op!=0);

}
```

Método menu:

```
private static int menu() {
    return Integer.parseInt(
        JOptionPane.showInputDialog("Restaurante Bom Garfo\n"
            + "1- Selecionar Mesa"
            + "\n2- Cadastrar Pedido na Mesa"
            + "\n3- Remover Pedido na Mesa"
            + "\n4- Fechar Mesa"
            + "\n0- Sair")
    );
}
```

Vamos manter um índice que usaremos para saber qual mesa estamos, por padrão a mesa padrão será zero.

```
public class Principal {

    // agora serão várias mesas

    static Comanda[] mesas;
    // qual a mesa que estamos servindo

    static int numMesa = 0;
```



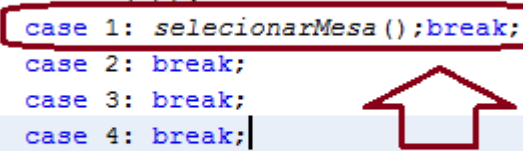
A primeira rotina a ser criada será a seleção da mesa, que usaremos cadastraremos quando um cliente for ocupa-la:

```

do {
    op = menu();
    switch(op){
        case 1: selecionarMesa();break;
        case 2: break;
        case 3: break;
        case 4: break;
        case 5: break;

    }
} while (op!=0);

```



Codificar a seleção da mesa:

```

private static void selecionarMesa() {
    numMesa = Integer.parseInt(JOptionPane.showInputDialog("Número da Mesa:"));
    // a mesa esta vazia?
    if(mesas[numMesa] == null){
        // cadastra a mesa vazia sem pedidos
        mesas[numMesa] = new Comanda(new ArrayList<Item>(),0);
        JOptionPane.showMessageDialog(null,"Mesa aberta");
    }
    else {
        JOptionPane.showMessageDialog(null,"Mesa selecionada");
    }
}
}

```

Agora cadastrar o pedido na mesa selecionada. Antes é necessário criar o cardápio servido. Para isso crie o objeto a seguir:

```

public class Principal {

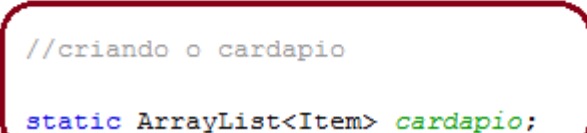
    // agora serão várias mesas

    static Comanda[] mesas;
    // qual a mesa que estamos servindo

    static int numMesa = 0;

    //criando o cardapio
    static ArrayList<Item> cardapio;
}

```



Preenchendo cardápio:

```

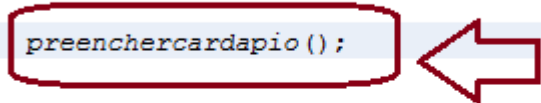
public static void main(String[] args) {

    int op;
    mesas = new Comanda[15];

    preencherCardapio();

    do {
        op = menu();
        switch(op) {
            case 1: selecionarMesa();break;
            case 2: break;
            case 3: break;
            case 4: break;
            case 5: break;
        }
    } while (op != 0);
}

```



Criando o cardápio (Adapte as suas necessidades):

```

private static void preencherCardapio() {
    cardapio = new ArrayList<>();
    cardapio.add(new Item(1, "CERVEJA", 5.40));
    cardapio.add(new Item(2, "REFRIGERANTE LATA", 2.30));
    cardapio.add(new Item(3, "SUCO NATURAL", 2.30));
    cardapio.add(new Item(4, "PORCAO BATATA", 10));
    cardapio.add(new Item(5, "PORCAO SALAME", 12));
    cardapio.add(new Item(6, "PIZZA MEDIA", 21));
    cardapio.add(new Item(7, "PRATO COMERCIAL", 9.50));
    cardapio.add(new Item(8, "MACARRONADA ITALIANA", 14));
    cardapio.add(new Item(9, "STROGONOFF CARNE", 17));
    cardapio.add(new Item(10, "PETIT GAUTEAU", 19));
    cardapio.add(new Item(11, "CAFE EXPRESSO GRANDE", 4.20));
}

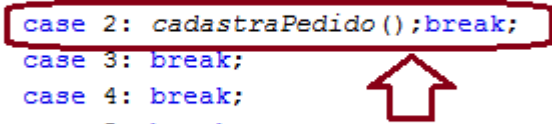
```

Rotina de cadastro de Pedido:

```

do {
    op = menu();
    switch(op) {
        case 1: selecionarMesa();break;
        case 2: cadastraPedido();break;
        case 3: break;
        case 4: break;
        case 5: break;
    }
} while (op != 0);

```



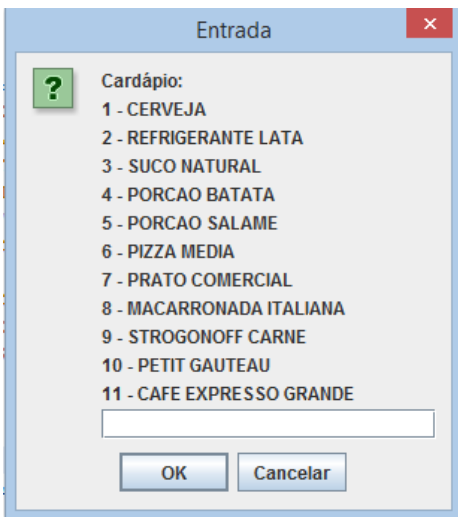
```
private static void cadastraPedido() {

    int op = Integer.parseInt(JOptionPane.showInputDialog("Cardápio:\n"
        + exibirCardapio()));
}
```

Note que a primeira coisa será exibir o cardápio para o garçon escolher o item do pedido:

```
private static String exibirCardapio() {
    String texto = "";
    for(Item p: cardapio){
        texto += (p.getCod() + " - " + p.getDescricao()+"\n");
    }
    return texto;
}
```

Que exibirá a seguinte tela:



Continuando a rotina de cadastro do Pedido, selecionando um item do cardápio e adicionando a lista de pedidos da mesa corrente:

```
private static void cadastraPedido() {

    int op = Integer.parseInt(JOptionPane.showInputDialog("Cardápio:\n"
        + exibirCardapio()));

    Item i = (Item) cardapio.get(op - 1);
    i.consumo(Integer.parseInt(JOptionPane.showInputDialog("Quantidade:")));

    mesas[numMesa].adicionar(i);
    JOptionPane.showMessageDialog(null, "Pedido cadastrado...");
}
```

Para testar vamos codificar o fechar a mesa, que calcula o valor gasto pela mesa e exibe os valores como um recibo:

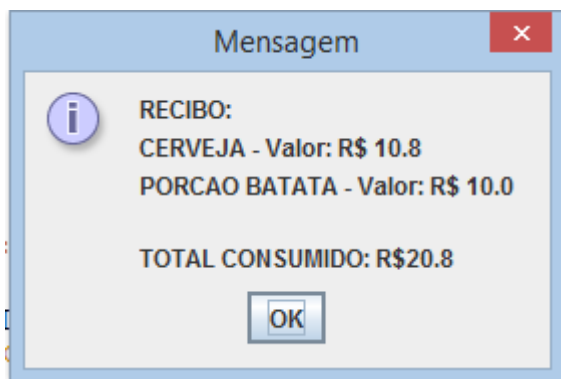

```

do {
    op = menu();
    switch(op) {
        case 1: selecionarMesa();break;
        case 2: cadastraPedido();break;
        case 3: break;
        case 4: fecharMesa();break;
        case 5: break;
    }
} while (op!=0);

private static void fecharMesa() {
    mesas[numMesa].fechar();
    String recibo="";
    for(Item p:mesas[numMesa].getPedidos() ){
        recibo+=p.getDescricao()+" - Valor: R$ " + p.getTotal() +"\n";
    }
    recibo+="\nTOTAL CONSUMIDO: R$"+ mesas[numMesa].getTotal();
    JOptionPane.showMessageDialog(null,"RECIBO:\n"
        + recibo);
    //remover a mesa
    mesas[numMesa] = null;
}

```

Testar o código, pedindo duas cervejas e uma porção de batata para a mesa 0:



Criando o método de exclusão de pedido:

```

do {
    op = menu();
    switch(op) {
        case 1: selecionarMesa();break;
        case 2: cadastraPedido();break;
        case 3: excluirPedido();break;
        case 4: fecharMesa();break;
        case 5: break;
    }
} while (op!=0);

```

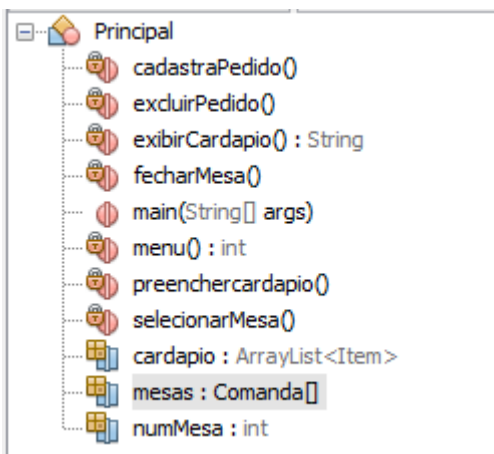
```

private static void excluirPedido() {
    String texto = "";
    int i = 0;
    for (Item p: mesas[numMesa].getPedidos()) {
        i++;
        texto += ("Item N°" + i + " - " + p.getDescricao()+"\n");
    }

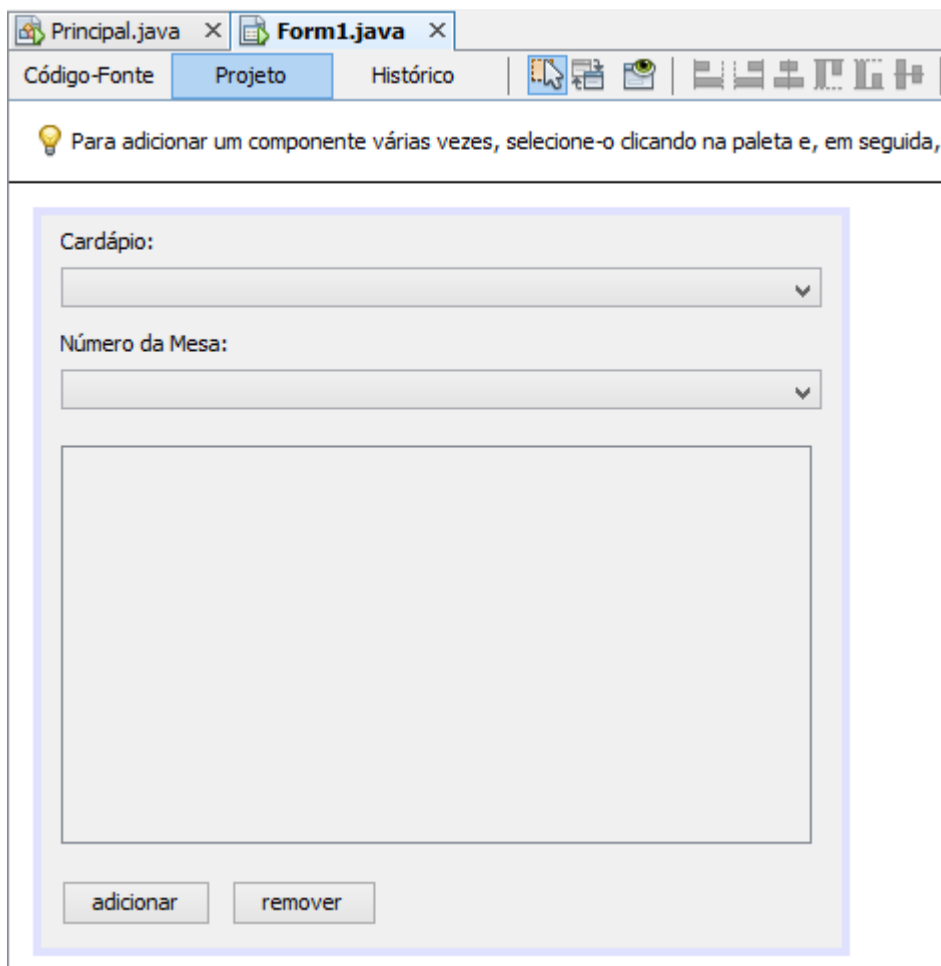
    JOptionPane.showMessageDialog(null, "MESA:" + numMesa
        + "\n"+texto);
    int op = Integer.parseInt(JOptionPane.showInputDialog("Item a Excluir:\n"
        ));
    // o numero do item é valido?
    if (op > 0 && op < mesas[numMesa].getPedidos().size()) {
        mesas[numMesa].getPedidos().remove(op-1);
        JOptionPane.showMessageDialog(null, "Item removido do pedido");
    }
}
}

```

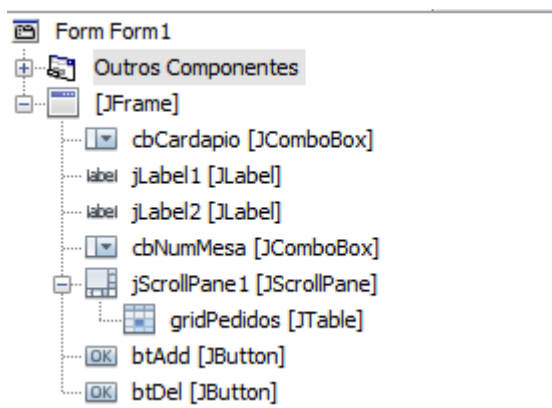
O método size devolve o tamanho do vetor, o método remove retira do vetor o item de pedido selecionado. A estrutura final do nosso projeto é:



A próxima etapa é criar uma interface mais elaborada para a nosso projeto. Remova a classe Principal de seu projeto e acrescente uma JFrame ao seu projeto chamado Form1, que será a janela padrão ao abrir o projeto. Nela iremos desenhar usando a paleta a seguinte aparência de formulário:



A definição dos nomes será:



Modifique o código fonte de seu projeto para trabalhar agora com a sua comanda:

```

7      * and open the template in the editor.
8      */
9
10     /**
11     *
12     * @author aluno
13     */
14     public class Form1 extends javax.swing.JFrame {
15
16         // agora serão várias mesas
17
18         static Comanda[] mesas;
19         // qual a mesa que estamos servindo
20
21         static int numMesa = 0;
22
23         //criando o cardapio
24
25         static ArrayList<Item> cardapio;
26     }
27     /**
28     * Creates new form Form1
29     */

```

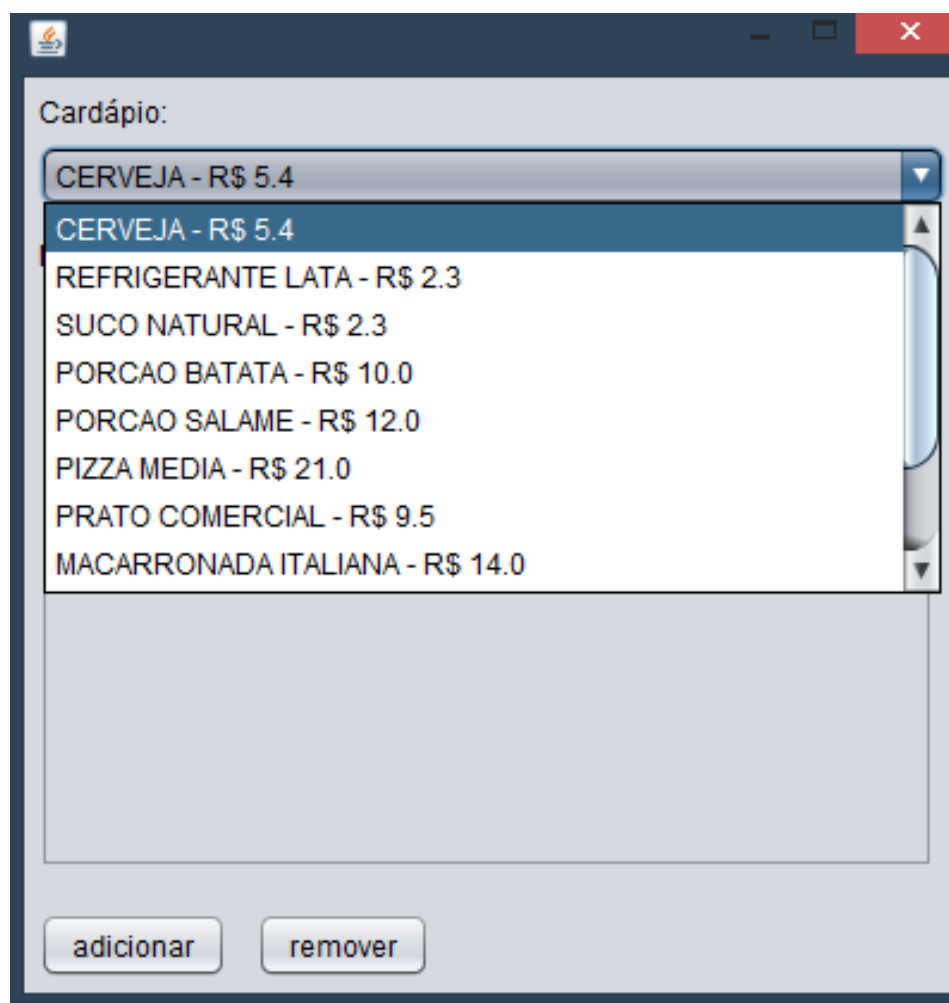
É necessário carregar o cardápio e os números das mesas:

[JFrame] - Propriedades X		
Propriedades	Vinculação	Código
componentResized	<nenhum>	
componentShown	<nenhum>	
focusGained	<nenhum>	
focusLost	<nenhum>	
hierarchyChanged	<nenhum>	
inputMethodTextChanged	<nenhum>	
keyPressed	<nenhum>	
keyReleased	<nenhum>	
keyTyped	<nenhum>	
mouseClicked	<nenhum>	
mouseDragged	<nenhum>	
mouseEntered	<nenhum>	
mouseExited	<nenhum>	
mouseMoved	<nenhum>	
mousePressed	<nenhum>	
mouseReleased	<nenhum>	
mouseWheelMoved	<nenhum>	
propertyChange	<nenhum>	
windowActivated	<nenhum>	
windowClosed	<nenhum>	
windowClosing	<nenhum>	
windowDeactivated	<nenhum>	
windowDeiconified	<nenhum>	
windowGainedFocus	<nenhum>	
windowIconified	<nenhum>	
windowLostFocus	<nenhum>	
windowOpened	formWindowOpened	

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    cardapio = new ArrayList<>();

    cardapio.add(new Item(1, "CERVEJA", 5.40));
    cardapio.add(new Item(2, "REFRIGERANTE LATA", 2.30));
    cardapio.add(new Item(3, "SUCO NATURAL", 2.30));
    cardapio.add(new Item(4, "PORCAO BATATA", 10));
    cardapio.add(new Item(5, "PORCAO SALAME", 12));
    cardapio.add(new Item(6, "PIZZA MEDIA", 21));
    cardapio.add(new Item(7, "PRATO COMERCIAL", 9.50));
    cardapio.add(new Item(8, "MACARRONADA ITALIANA", 14));
    cardapio.add(new Item(9, "STROGONOFF CARNE", 17));
    cardapio.add(new Item(10, "PETIT GAUTAU", 19));
    cardapio.add(new Item(11, "CAFE EXPRESSO GRANDE", 4.20));
    for(Item p: cardapio){
        cbCardapio.addItem(p.getDescricao()+" - R$ " +p.getPreco());
    }
}
```

Se executarmos teremos a seguinte aparência:



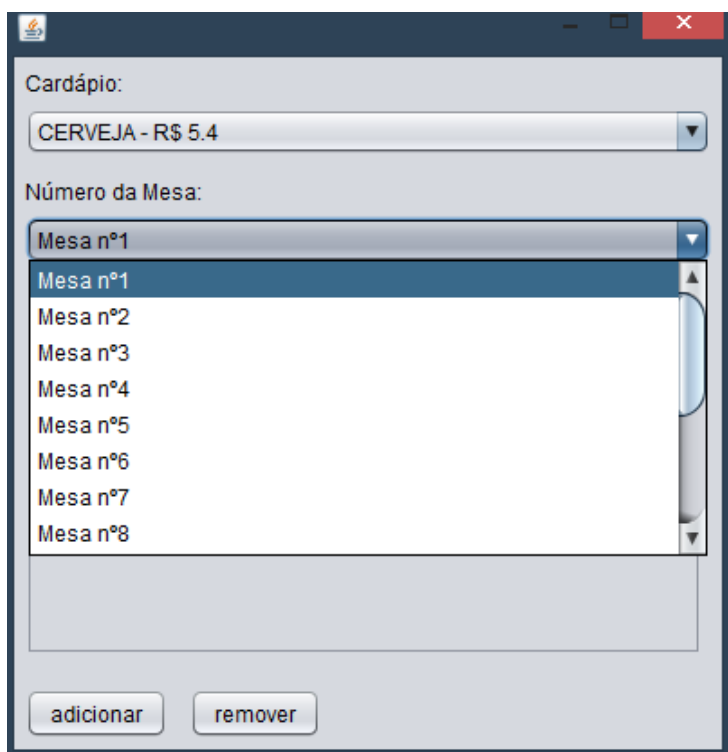
É necessário preencher também o combobox com o número das mesas:

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    cardapio = new ArrayList<>();

    cardapio.add(new Item(1, "CERVEJA", 5.40));
    cardapio.add(new Item(2, "REFRIGERANTE LATA", 2.30));
    cardapio.add(new Item(3, "SUCO NATURAL", 2.30));
    cardapio.add(new Item(4, "PORCAO BATATA", 10));
    cardapio.add(new Item(5, "PORCAO SALAME", 12));
    cardapio.add(new Item(6, "PIZZA MEDIA", 21));
    cardapio.add(new Item(7, "PRATO COMERCIAL", 9.50));
    cardapio.add(new Item(8, "MACARRONADA ITALIANA", 14));
    cardapio.add(new Item(9, "STROGONOFF CARNE", 17));
    cardapio.add(new Item(10, "PETIT GAUTAU", 19));
    cardapio.add(new Item(11, "CAFE EXPRESSO GRANDE", 4.20));

    mesas = new Comanda[15];
    for(Item p: cardapio){
        cbCardapio.addItem(p.getDescricao()+" - R$ " +p.getPreco());
    }
    for (int i = 0; i < 15; i++) {
        cbNumMesa.addItem("Mesa nº" + (i+1));
    }
}
```

Visualmente teremos:



Agora é preciso projetar que ao selecionar uma mesa eu possa adicionar itens no cardápio. Vamos criar as colunas do grid:

```

public class Form1 extends javax.swing.JFrame {

    // agora serão várias mesas

    static Comanda[] mesas;
    // qual a mesa que estamos servindo

    static int numMesa = 0;

    //criando o cardapio

    static ArrayList<Item> cardapio;

    DefaultTableModel modelo = new DefaultTableModel();
    /**
     * Creates new form Form1
     */
    public Form1() {
        initComponents();
    }
}

```



Modifique também o evento de abertura do formulário:

```

private void formWindowOpened(java.awt.event.WindowEvent evt) {
    cardapio = new ArrayList<>();
    cardapio.add(new Item(1, "CERVEJA", 5.40));
    cardapio.add(new Item(2, "REFRIGERANTE LATA", 2.30));
    cardapio.add(new Item(3, "SUCO NATURAL", 2.30));
    cardapio.add(new Item(4, "PORCAO BATATA", 10));
    cardapio.add(new Item(5, "PORCAO SALAME", 12));
    cardapio.add(new Item(6, "PIZZA MEDIA", 21));
    cardapio.add(new Item(7, "PRATO COMERCIAL", 9.50));
    cardapio.add(new Item(8, "MACARRONADA ITALIANA", 14));
    cardapio.add(new Item(9, "STROGONOFF CARNE", 17));
    cardapio.add(new Item(10, "PETIT GAUTEAU", 19));
    cardapio.add(new Item(11, "CAFE EXPRESSO GRANDE", 4.20));

    mesas = new Comanda[15];

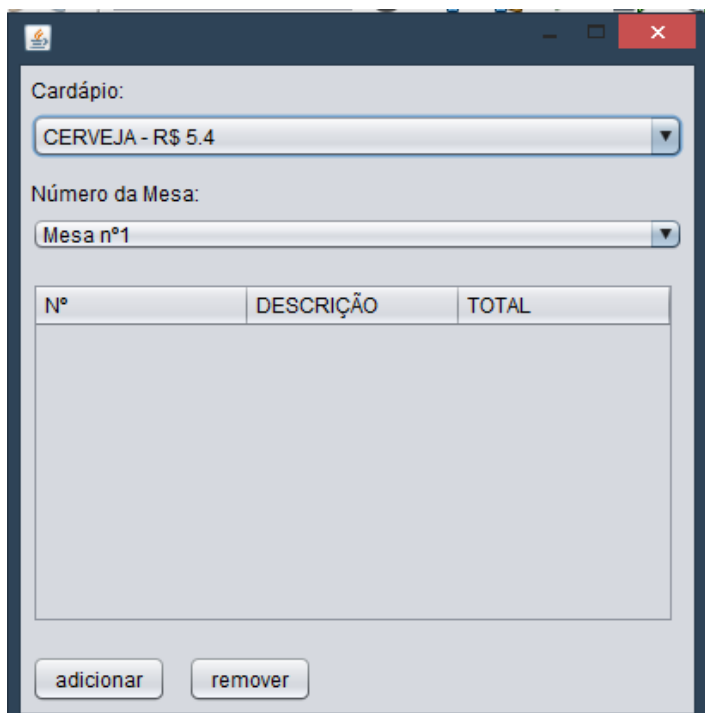
    modelo.addColumn("N°");
    modelo.addColumn("DESCRIÇÃO");
    modelo.addColumn("TOTAL");
    gridPedidos.setModel(modelo);

    for(Item p: cardapio){
        cbCardapio.addItem(p.getDescricao()+" - R$ " +p.getPreco());
    }
    for (int i = 0; i < 15; i++) {
        cbNumMesa.addItem("Mesa n°" + (i+1));
    }
}
}

```



Resultado visual:



Codificar o evento actionPerformed do combobox número da mesa:

```
private void cbNumMesaActionPerformed(java.awt.event.ActionEvent evt) {
    numMesa = cbNumMesa.getSelectedIndex();
    // a mesa esta vazia?
    if(mesas[numMesa] == null){
        // cadastra a mesa vazia sem pedidos
        mesas[numMesa] = new Comanda(new ArrayList<Item>(),0);
        JOptionPane.showMessageDialog(null,"Mesa aberta");
    }
    else {
        JOptionPane.showMessageDialog(null,"Mesa selecionada");
        if(mesas[numMesa].getPedidos() !=null){
            int i =1;
            for(Item p:mesas[numMesa].getPedidos() ){
                modelo.addRow(new Object[]{i,p.getDescricao(),p.getTotal()});
            }
        }
    }
}
```

Codificar o evento actionPerformed do botão adicionar:

```
private void btAddActionPerformed(java.awt.event.ActionEvent evt) {
    int op = cbCardapio.getSelectedIndex();
    Item i = (Item) cardapio.get(op);
    i.consumo(Integer.parseInt(JOptionPane.showInputDialog("Quantidade:")));
    mesas[numMesa].adicionar(i);
    JOptionPane.showMessageDialog(null,"Pedido cadastrado...");
    int pos = mesas[numMesa].getPedidos().size();
    modelo.addRow(new Object[]{
        pos,i.getDescricao(),
        i.getTotal()
    });
}
```

O resultado visual será:

Cardápio:

PORCAO SALAME - R\$ 12.0

Número da Mesa:

Mesa nº1

Nº	DESCRIÇÃO	TOTAL
1	CERVEJA	27.0
2	PORCAO BATATA	20.0
3	PORCAO SALAME	12.0

adicionar remover

Exercício

1. Codificar o botão excluir
2. Acrescentar e codificar o botão fechar mesa.