Qualidade e Testes de Software

Professor Gilmar Luiz de Borba

Prática:

Qualidade de código com métricas em software : usando a ferramenta Metrics do Eclipse.

Objetivo: analisar algumas métricas de código com vistas a qualidade do produto final.

As métricas trazem informações sobre a complexidade total de linhas de código, quantidade de campos estáticos, quantidade de classes, quantidade de interfaces, falta de coesão em métodos etc. essas informações facilitam a análise sobre a complexidade e possíveis fragilidades do código.

(1) Criar o projeto: QualiMetricas

Criar o package: a seu critério

Criar a classe: TestePilha

Copiar a classe: TestePilha ... **SALVE O PROJETO!**

Para verificar as métricas da aplicação, siga os passos:

(2) Instale o plugin metrics no Eclipse.

2.1 - Escolha a opção ***Help***, em seguida ***Install New Software***

2.2 - Acionar o botão ADD, inserir o link:

http://metrics2.sourceforge.net/update

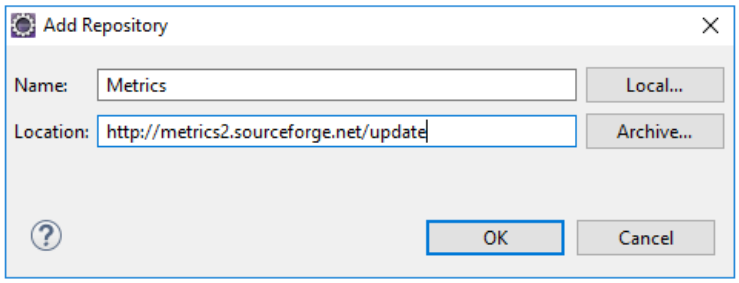
Use Metrics2 (link acima) para versões mais novas como "Photon"

Inserir o nome: Metrics

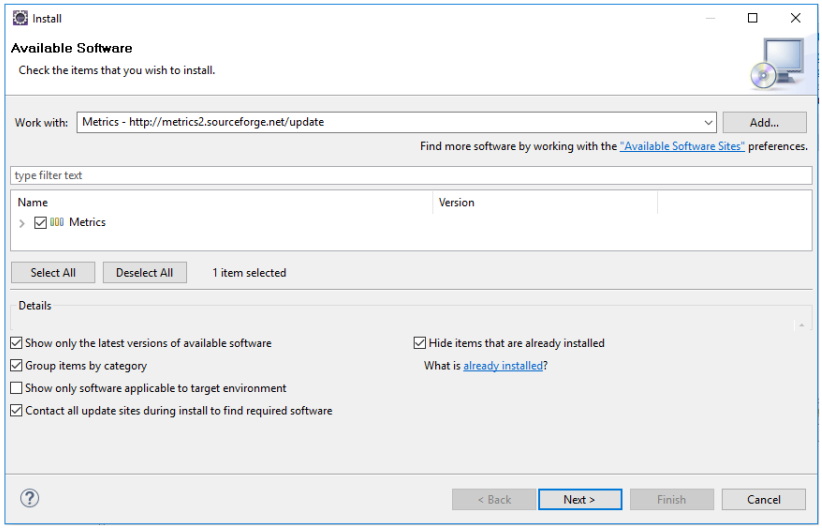
http://metrics.sourceforge.net/update

Use Metrics link (acima) para versões antigas do Eclipse (até 3.61)

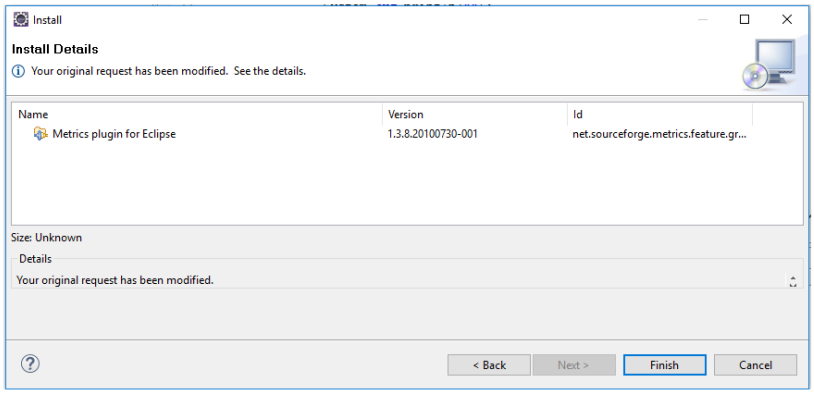
Inserir o nome: Metrics



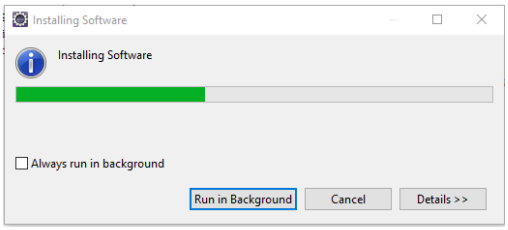
2.3 - Selecione/confirme o plugin: Metrics, em seguida acione o botão Next. Veja imagem.

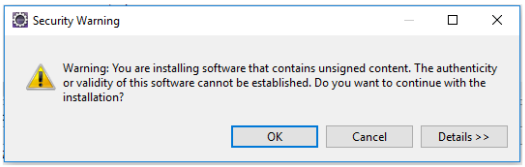


2.4 – Aguardar a instalação ...



2.5 – ... em seguida acionar FINISH.





c:\prof_gilmar_borba\qsoft\imagens\

Caso apareça mensagem de warning, sobre conteúdo sem assinatura, confirme. Reinicie o eclipse para que as alterações sejam confirmadas.

Qualidade e Testes de Software

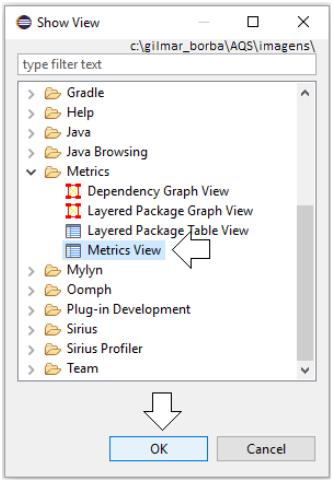
Professor Gilmar Luiz de Borba

(3) Verifique o item Preferences no menu Windows. Verifique a opção Metrics Preferences. Apenas para entender os acrônimos das métricas.

Observe as métricas abordadas, como: PAR (número de parâmetros), NOC (número de classes), NSM (número de métodos estáticos), NOP (número de packages) , VG (Complexidade ciclomática), LCOM (falta de coesão), NORM (número de métodos sobrepostos) , etc.

(4) Verifique o plugin Metrics instalado, para isso use o seguinte caminho de menu: Window, Show View, Other.

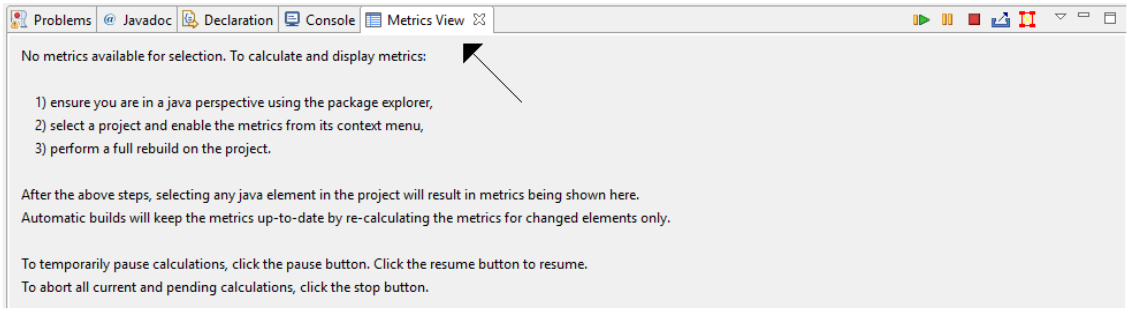
Janela metrics View



ATENÇÃO:

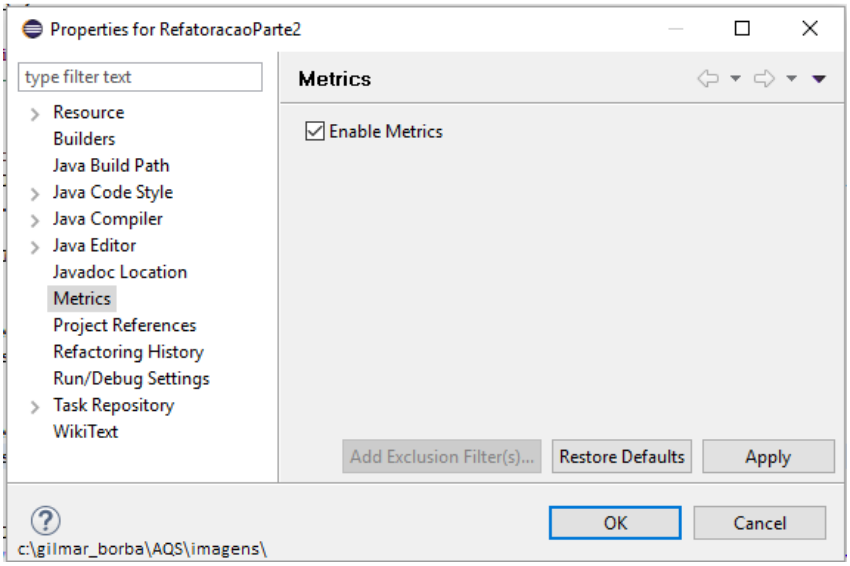
Expanda o item do plugin Metrics
Selecione o item Metrics View
Acione o botão [OK]

Observe que apareceu a janela "***Metrics View***" no Rodapé da IDE do Eclipse.



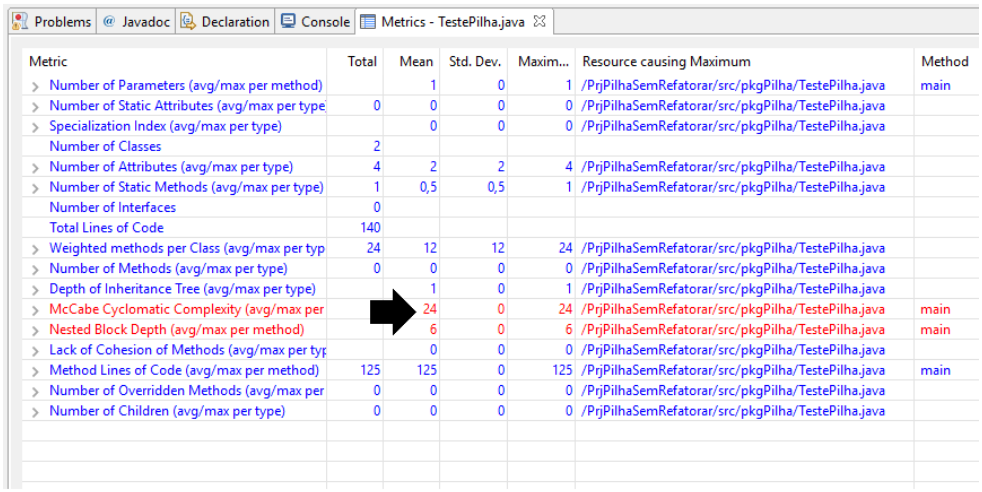
Para habilitar o ***Plugin Metrics*** nesse novo projeto, siga os passos:

1. Acione o botão direito do mouse sobre o projeto.
2. Escolha a opção (item de menu): **Properties**.
3. Selecione o item **Metrics** na Tree View.
4. Selecione o checkbox "**Enable Metrics**" (Veja a imagem).



(5) Ao reconstruir o projeto (Build ou Build All), ou executá-lo com Automatic Build, serão apresentadas as métricas na parte inferior da IDE "***Metrics View***".

Complexidade Ciclomática antes da refatoração:



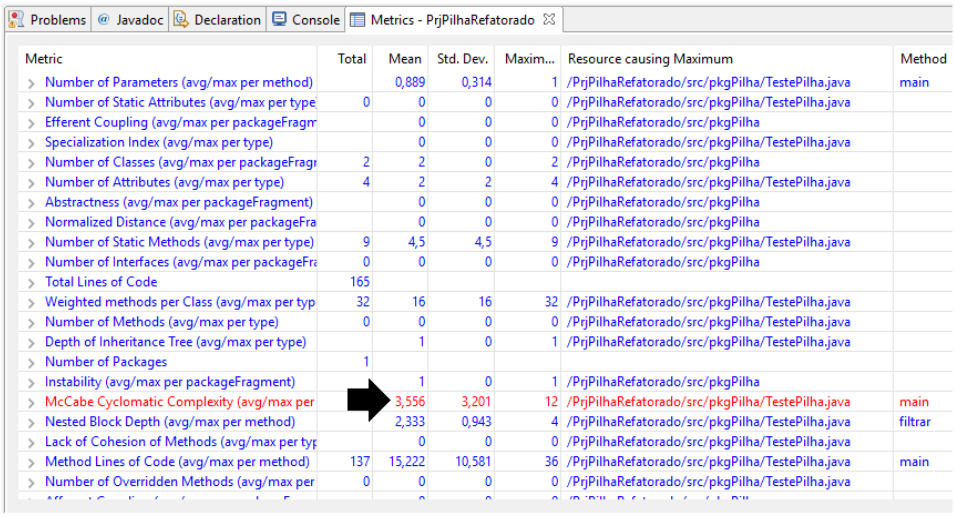
Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		1	0	1	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	main
Number of Static Attributes (avg/max per type	0	0	0	0	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Specialization Index (avg/max per type)		0	0	0	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Number of Classes	2					
Number of Attributes (avg/max per type)	4	2	2	4	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Number of Static Methods (avg/max per type)	1	0,5	0,5	1	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Number of Interfaces	0					
Total Lines of Code	140					
Weighted methods per Class (avg/max per typ	24	12	12	24	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Number of Methods (avg/max per type)	0	0	0	0	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Depth of Inheritance Tree (avg/max per type)		1	0	1	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
McCabe Cyclomatic Complexity (avg/max per		24	0	24	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	main
Nested Block Depth (avg/max per method)		6	0	6	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	main
Lack of Cohesion of Methods (avg/max per typ		0	0	0	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Method Lines of Code (avg/max per method)	125	125	0	125	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	main
Number of Overridden Methods (avg/max per	0	0	0	0	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	
Number of Children (avg/max per type)	0	0	0	0	/PrjPilhaSemRefatorar/src/pkgPilha/TestePilha.java	

Qualidade e Testes de Software

Professor Gilmar Luiz de Borba

(6) Refatore o projeto de modo a reduzir a Complexidade Ciclomática

Complexidade Ciclomática, sugerida após a refatoração:



Problems | Javadoc | Declaration | Console | Metrics - PrjPilhaRefatorado

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		0,889	0,314	1	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	main
Number of Static Attributes (avg/max per type	0	0	0	0	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Efferent Coupling (avg/max per packageFragm		0	0	0	/PrjPilhaRefatorado/src/pkgPilha	
Specialization Index (avg/max per type)		0	0	0	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Number of Classes (avg/max per packageFragı	2	2	0	2	/PrjPilhaRefatorado/src/pkgPilha	
Number of Attributes (avg/max per type)	4	2	2	4	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Abstractness (avg/max per packageFragment)		0	0	0	/PrjPilhaRefatorado/src/pkgPilha	
Normalized Distance (avg/max per packageFra		0	0	0	/PrjPilhaRefatorado/src/pkgPilha	
Number of Static Methods (avg/max per type)	9	4,5	4,5	9	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/PrjPilhaRefatorado/src/pkgPilha	
Total Lines of Code	165					
Weighted methods per Class (avg/max per typ	32	16	16	32	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Number of Methods (avg/max per type)	0	0	0	0	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Depth of Inheritance Tree (avg/max per type)		1	0	1	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Number of Packages	1					
Instability (avg/max per packageFragment)		1	0	1	/PrjPilhaRefatorado/src/pkgPilha	
McCabe Cyclomatic Complexity (avg/max per		3,556	3,201	12	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	main
Nested Block Depth (avg/max per method)		2,333	0,943	4	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	filtrar
Lack of Cohesion of Methods (avg/max per typ		0	0	0	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	
Method Lines of Code (avg/max per method)	137	15,222	10,581	36	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	main
Number of Overridden Methods (avg/max per	0	0	0	0	/PrjPilhaRefatorado/src/pkgPilha/TestePilha.java	

(6) Dicas para refatoração da classe TestePilha.

6.0 - Refatoração de formatação: endentação, espaçamento vertical/horizontal.

6.1 - Refatoração de extração de método: Texto do menu.

6.2 - Refatoração de mudança de nome, vários casos.

6.3 - Refatoração de extração de comentário.

6.4 - Refatoração de extração de "*imports*" que não estão sendo usados.

6.5 - Refatoração de criação de um método para mostrar mensagens repetitivas como "A pilha está vazia" e outras.

6.6 - Refatoração de extração de método: criar o método ***empilhar***. A partir da entrada de dados, só passará como parâmetro um tipo Pilha.

6.7 - Refatoração de extração de método: criar o método ***desempilhar*** do início do código ao fim, só passará como parâmetro um tipo Pilha.

6.8 - Refatoração de extração de método: criar o método ***consultar*** do início do código ao fim, só passará como parâmetro um tipo Pilha.

Qualidade e Testes de Software

Professor Gilmar Luiz de Borba

6.9 - Refatoração de extração de método: criar o método ***esvaziar*** do início do código ao fim, só passará como parâmetro um tipo Pilha.

6.10 - Refatoração de extração de método: criar o método ***quantidadeChapas***, do início do código ao fim, só passará como parâmetro um tipo Pilha.

6.11 - Refatoração de extração de método: criar o método ***filtrar***, do início do código ao fim, só passará como parâmetro um tipo Pilha.

Qualidade e Testes de Software

Professor Gilmar Luiz de Borba

CÓDIGO FONTE DO PROGRAMA

```
package pkgPilha;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.*;
public class TestePilha {
    // Definindo a classe que representará cada elemento da Pilha
    private static class Pilha {
        public int numero;
        public double valor;
        public int pedido;
        public Pilha prox;
    }
    public static void main(String[] args) {
        Pilha topo = null;
        Pilha aux;
        int op = 0;
        do {
            try {
                op = Integer.parseInt(JOptionPane.showInputDialog(opcoesMenu(), "1"));
            } catch (Exception e) {
                // Silent exception
                System.out.println("");
            }

            if (op < 1 || op > 8) {
                mensagem("Opção Inválida!");
            }
            // INSERIR/EMPILHAR CHAPAS - DESAFIO 1
            if (op == 1) {
              int numero = Integer.parseInt(JOptionPane.showInputDialog("NÚMERO DA CHAPA", "0"));
                aux = topo;
                boolean achou = false;
                while (aux != null) {
                    if (aux.numero == numero) {
                        achou = true;
                        mensagem("Esse número de chapa já foi empilhado.\nFavor verificar!");
                        break;
                    }
                    aux = aux.prox;
                }
                if (achou == false) {
                    Pilha novo = new Pilha();
                    novo.numero = numero;
                    novo.valor = Double.parseDouble(JOptionPane.showInputDialog("VALOR DA CHAPA", "0"));
                     novo.pedido = Integer.parseInt(JOptionPane.showInputDialog("NÚMERO DO PEDIDO", "0"));
                    novo.prox = topo;
                    topo = novo;
                }
            }
            // CONSULTAR CHAPAS CADASTRADAS
            if (op == 2) {
                if (topo == null) {
                    mensagem("A PILHA está vazia!");
                } else {
                    JTextArea saida = new JTextArea(6, 35); // HEIGHT X WIDTH
                    JScrollPane scroll = new JScrollPane(saida);
                    saida.append("NÚMERO\t" + "VALOR\t" + "PEDIDO\n");
```

```
        saida.append("================================\n");
        aux = topo;
        while (aux != null) {
            // VERIFICAR ENDER E PROX (DESAFIO 2)
            if (aux.prox != null) {
                System.out.println("Num: " + aux.numero + ", endereço: "
                        + aux.hashCode() + " => " + " Prox => "
                        + aux.prox.hashCode() + "\n");
            }

             saida.append(aux.numero + "\t" + aux.valor + "\t" + aux.pedido + "\n");
            aux = aux.prox;
        }
        JOptionPane.showMessageDialog(null, scroll,
                "CONSULTAR CHAPAS CADASTRADAS",
                JOptionPane.INFORMATION_MESSAGE);
    }
}
// DESEMPILHAR CHAPA
if (op == 3) {
    if (topo == null) {
        mensagem("A PILHA está vazia!");
    } else {
        mensagem("Número: " + topo.numero + ", foi removido.");
        topo = topo.prox;
    }
}
// ESVAZIAR PÁTIO
if (op == 4) {
    if (topo == null) {
        mensagem("A Pilha está vazia!");
    } else {
        topo = null;
        mensagem("A Pilha foi esvaziada!");
    }
}
// QUANTIDADE E CHAPAS
if (op == 5) {
    aux = topo;
    int n = 0;
    while (aux != null) {
        aux = aux.prox;
        n++;
    }
    mensagem("A Pilha contém: " + n + " elementos.");
}
// FILTRAR CHAPAS CADASTRADAS - DESAFIO 3
if (op == 6) {
    if (topo == null) {
        mensagem("A PILHA está vazia!");
    } else {
        int npedido = Integer.parseInt(JOptionPane.showInputDialog(
                "NÚMERO DO PEDIDO", "0"));
        JTextArea saida = new JTextArea(6, 35); // HEIGHT X WIDTH
        JScrollPane scroll = new JScrollPane(saida);
        saida.append("FILTRO, CHAPAS PARA O PEDIDO NRO: " + npedido + "\n");
        saida.append("NÚMERO\t" + "VALOR\t" + "PEDIDO\n");
        saida.append("================================\n");
        aux = topo;
        while (aux != null) {
```

```
                        if (aux.pedido == npedido) {
                            saida.append(aux.numero + "\t" + aux.valor + "\t" + aux.pedido + "\n");
                        }
                        aux = aux.prox;
                    }
                    JOptionPane.showMessageDialog(null, scroll,
                            "CONSULTAR CHAPAS CADASTRADAS",
                            JOptionPane.INFORMATION_MESSAGE);
                }
            }

        } while (op != 7);
        System.out.println(">>> PROGRAMA FINALIZADO!");

    } // FIM MÉTODO MAIN()

    // STRING - OPÇÕES MENU
    private static String opcoesMenu() {
        String menu = "\nMENU DE OPÇÕES\n"
                + "\n1 - Empilhar Chapas."
                + "\n2 - Consultar Todas as Chapas."
                + "\n3 - Desempilhar Chapas."
                + "\n4 - Remover Todas as Chapas Da Pilha."
                + "\n5 - Verificar Quantidade de Chapas."
                + "\n6 - Filtrar Chapas Por Pedido."
                + "\n7 - Sair";
        return menu;
    }
    // MÉTODO VOID MENSAGEM()

    private static void mensagem(String a) {
        JOptionPane.showMessageDialog(null, a, "Mensagem do Programa",
JOptionPane.INFORMATION_MESSAGE);
    }
} // FIM CLASSE
```