REFATORAÇÃO
(SONAR e METRICS)

MÉTRICAS DE SOFTWARE

(Considerações iniciais)

MÉTRICAS DE SOFTWARE

Métricas

Métricas são usadas como indicadores de qualidade, ou seja, atestam se os objetivos definidos para um método, um módulo, ou para o projeto foram alcançados.

Por que medir?

1. Para melhorar a qualidade do produto.
2. Para possibilitar um melhor retorno de investimento.
3. Podem ser medidos: o processo, o teste em si, a automação do teste, etc.

Exemplos

1. % de defeitos encontrados.
2. % de defeitos resolvidos.
3. Métricas de código.
4. Métricas subjetivas.
5. Métrica de conferência e análise de teste.
1. Benchmark de performance. etc.

MÉTRICAS DE SOFTWARE

Atributos relacionados às métricas (MOLINARI, 2003, Pág.. 62)

1. Manutenibilidade.
2. Eficácia.
3. Confiabilidade.
4. Flexibilidade.
5. Usabilidade.
6. Robusteza.
7. Portabilidade.

MÉTRICAS

Objetivo das métricas

Auxiliar (dar suporte) a uma melhor gerência dos projetos de software.

Aumentar a qualidade dos artefatos de software (produto final).

Garantir que os usuários recebam produtos dentro das especificações e dos requerimentos. (lembrar especificação é diferente de requerimento).

Quais são as informações apresentadas nas métricas?

Tamanho do software.
Complexidade.
Quantidade de defeitos do software.
Tipos de variáveis usadas.
Coesão dos métodos.
etc.

Obs: essas informações são normalmente indicativos da qualidade do software.

MÉTRICAS

Quais são as informações apresentadas nas métricas?

Tamanho do software.
Complexidade.
Quantidade de defeitos do software.
Tipos de variáveis usadas.
Coesão dos métodos.
etc.

Obs: essas informações são normalmente indicativos da qualidade do software.

COMPLEXIDADE COGNITIVA

Complexidade Cognitiva

É uma medida de quão difícil é entender uma unidade de código. Está associada à complexidade de um software e consequentemente a sua dificuldade de manutenção.

A complexidade cognitiva indica a dificuldade de ler e entender um determinado código.

Lembre que a complexidade ciclomática também está associada à complexidade de um software e consequentemente a sua dificuldade de manutenção, ou seja, a complexidade ciclomática indica a dificuldade ou esforço para se fazer testes de unidade.

Complexidade Cognitiva
Disponível em: http://artesoftware.com.br/complexidade-cognitiva/
Acessado: Janeiro de 2019

Atividade TestePilha
(Comentários)

SONAR

```
private static class Pilha {
    public int numero;
    public double valor;
    public int pedido;
    public Pilha prox;
}
```

Problems | Javadoc | Declaration | Console | SonarLint On-The-Fly

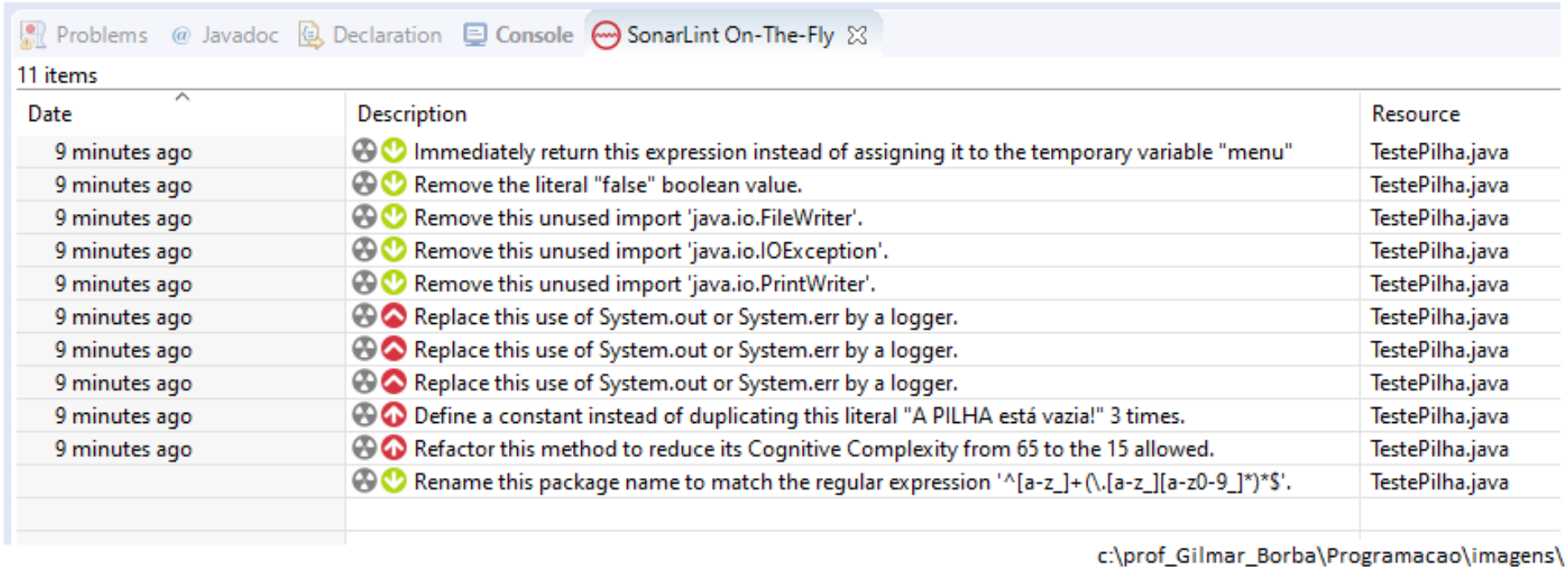
15 items

Date	Description	Resource
few seconds ago	Make numero a static final constant or non-public and provide accessors if needed.	TestePilha.java
few seconds ago	Make pedido a static final constant or non-public and provide accessors if needed.	TestePilha.java
few seconds ago	Make prox a static final constant or non-public and provide accessors if needed.	TestePilha.java
few seconds ago	Make valor a static final constant or non-public and provide accessors if needed.	TestePilha.java
2 minutes ago	Immediately return this expression instead of assigning it to the temporary variable "menu".	TestePilha.java
2 minutes ago	Remove the literal "false" boolean value.	TestePilha.java
2 minutes ago	Remove this unused import 'java.io.FileWriter'.	TestePilha.java
2 minutes ago	Remove this unused import 'java.io.IOException'.	TestePilha.java
2 minutes ago	Remove this unused import 'java.io.PrintWriter'.	TestePilha.java
2 minutes ago	Replace this use of System.out or System.err by a logger.	TestePilha.java
2 minutes ago	Replace this use of System.out or System.err by a logger.	TestePilha.java
2 minutes ago	Replace this use of System.out or System.err by a logger.	TestePilha.java
2 minutes ago	Define a constant instead of duplicating this literal "A PILHA está vazia!" 3 times.	TestePilha.java
2 minutes ago	Refactor this method to reduce its Cognitive Complexity from 65 to the 15 allowed.	TestePilha.java
	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.	TestePilha.java

c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
private static class Pilha {
    private int numero;
    private double valor;
    private int pedido;
    private Pilha prox;
}
```



SONAR

```
package pkgPilha;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.*;
public class TestePilha {
```

Problems | Javadoc | Declaration | Console | SonarLint On-The-Fly

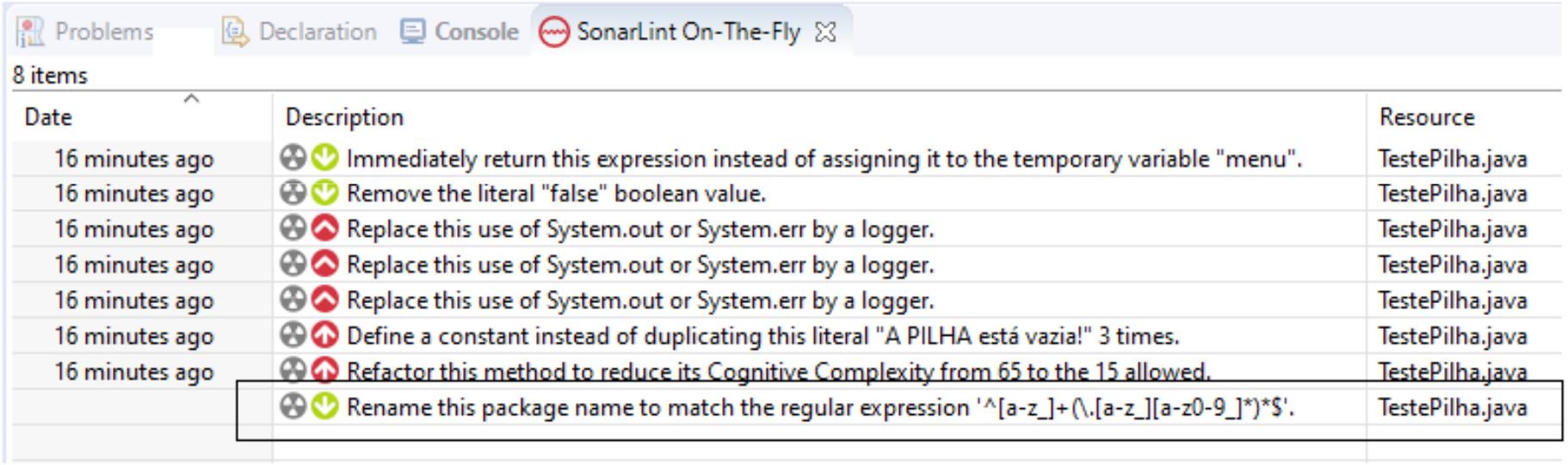
11 items

Date	Description	Resource
9 minutes ago	Immediately return this expression instead of assigning it to the temporary variable "menu".	TestePilha.java
9 minutes ago	Remove the literal "false" boolean value.	TestePilha.java
9 minutes ago	Remove this unused import 'java.io.FileWriter'.	TestePilha.java
9 minutes ago	Remove this unused import 'java.io.IOException'.	TestePilha.java
9 minutes ago	Remove this unused import 'java.io.PrintWriter'.	TestePilha.java
9 minutes ago	Replace this use of System.out or System.err by a logger.	TestePilha.java
9 minutes ago	Replace this use of System.out or System.err by a logger.	TestePilha.java
9 minutes ago	Replace this use of System.out or System.err by a logger.	TestePilha.java
9 minutes ago	Define a constant instead of duplicating this literal "A PILHA está vazia!" 3 times.	TestePilha.java
9 minutes ago	Refactor this method to reduce its Cognitive Complexity from 65 to the 15 allowed.	TestePilha.java
	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.	TestePilha.java

c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
package pkgPilha;
import javax.swing.*;
public class TestePilha {
```



c:\prof_Gilmar_Borba\Programacao\imagens\

Padrão da nomenclatura dos pacotes na linguagem JAVA

O padrão da Oracle (SUN) para dar nome aos pacotes se refere ao nome da empresa que desenvolveu a classe:

br.com.nomedaempresa.nomedoprojeto.pacote

Dessa forma a classe "calculo" desse pacote possui o nome completo:
br.com.nomedaempresa.nomedoprojeto.pacote.calculo

ATENÇÃO:
Os pacotes devem ser escritos totalmente com letras minúsculas (sugestão da Oracle), não importa quantas palavras estejam contidas nele (nesse caso não usar a notação "Camelo")

OBSERVAÇÃO:
As classes do pacote padrão de bibliotecas não seguem essa nomenclatura, que foi dada para bibliotecas de terceiros.

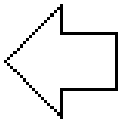
```java
// INSERIR/EMPILHAR
if (op == 1)
{
    int numero = Integer.parseInt(JOptionPane.showInputDialog("NÚMERO DA CHAPA", "0"));
    aux = topo;
    boolean achou = false;
    while (aux != null)
    {
        if (aux.numero == numero)
        {
            achou = true;
            mensagem("Esse 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;
    }
}
```

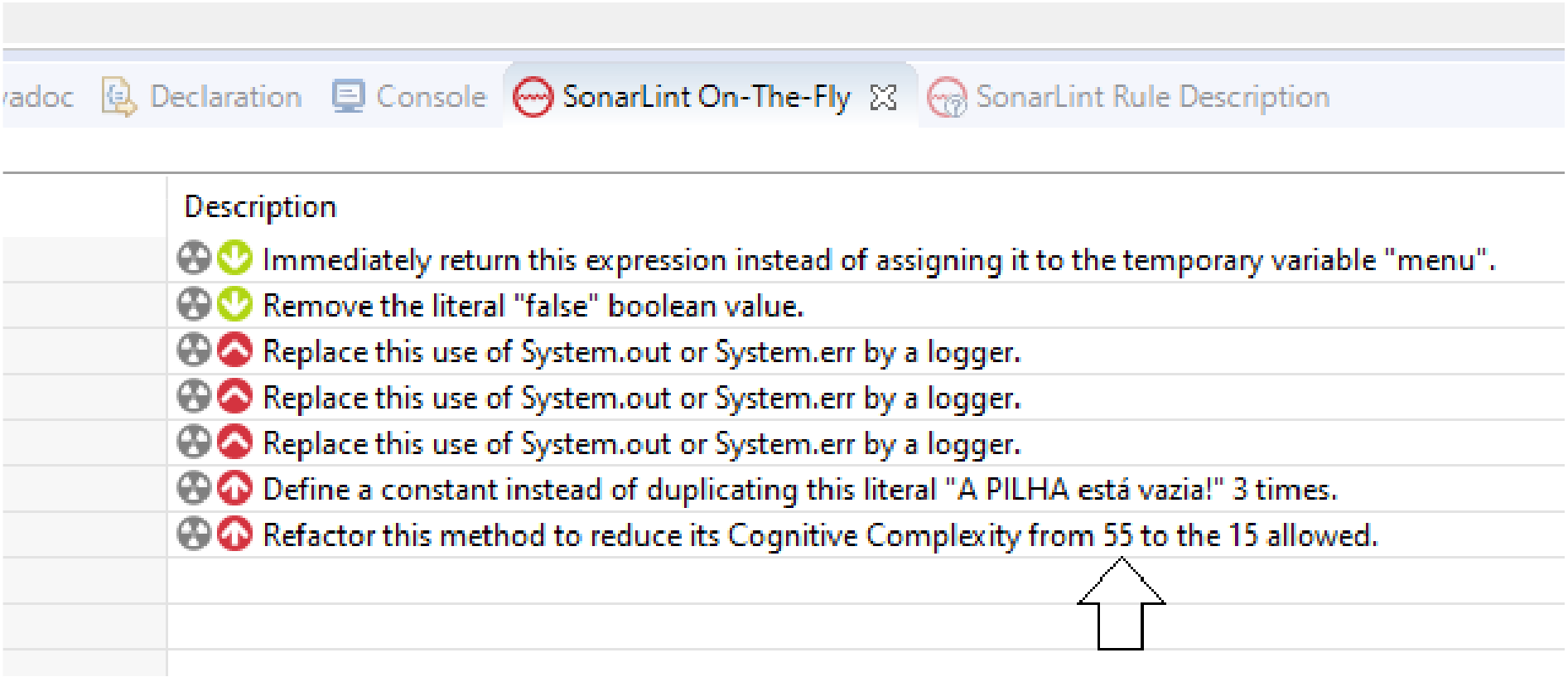
Declaration | Console | SonarLint On-The-Fly | SonarLint Rule Description

Description	Resource
Immediately return this expression instead of assigning it to the temporary variable "menu".	TestePilha.java
Remove the literal "false" boolean value.	TestePilha.java
Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java
Define a constant instead of duplicating this literal "A PILHA está vazia!" 3 times.	TestePilha.java
Refactor this method to reduce its Cognitive Complexity from 65 to the 15 allowed.	TestePilha.java

SONAR

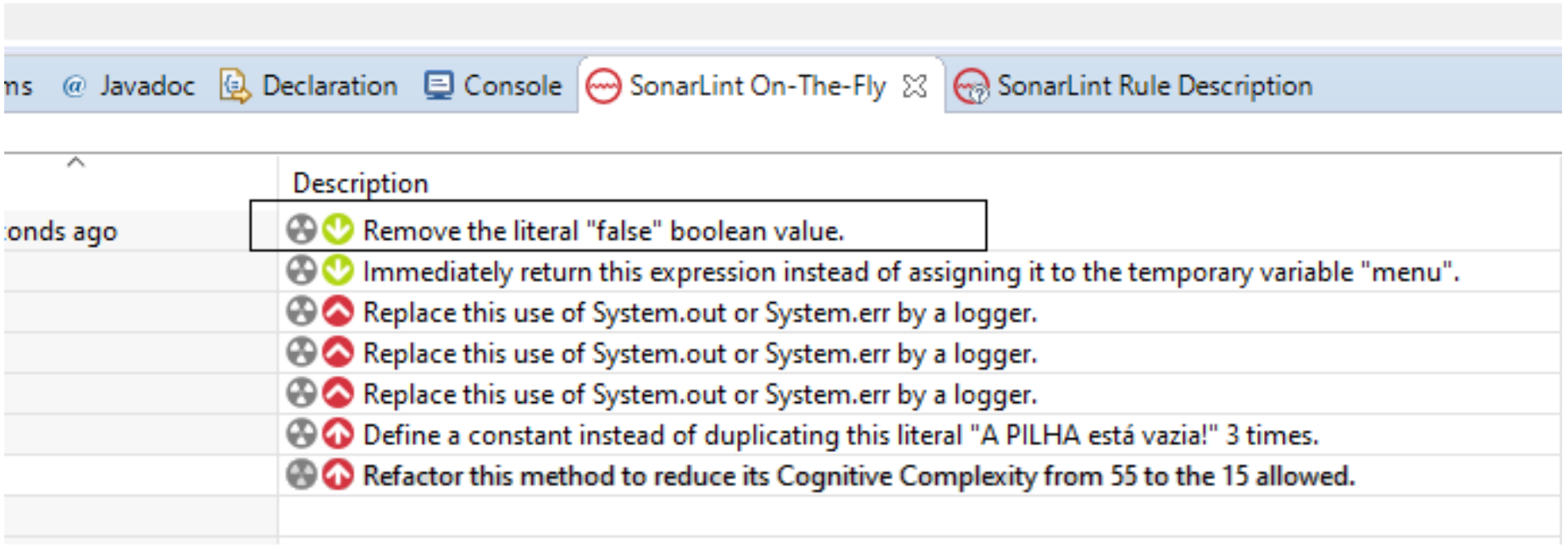
```
// INSERIR/EMPILHAR
if (op == 1) {
    topo = empilharChapas(topo);
}
```





SONAR

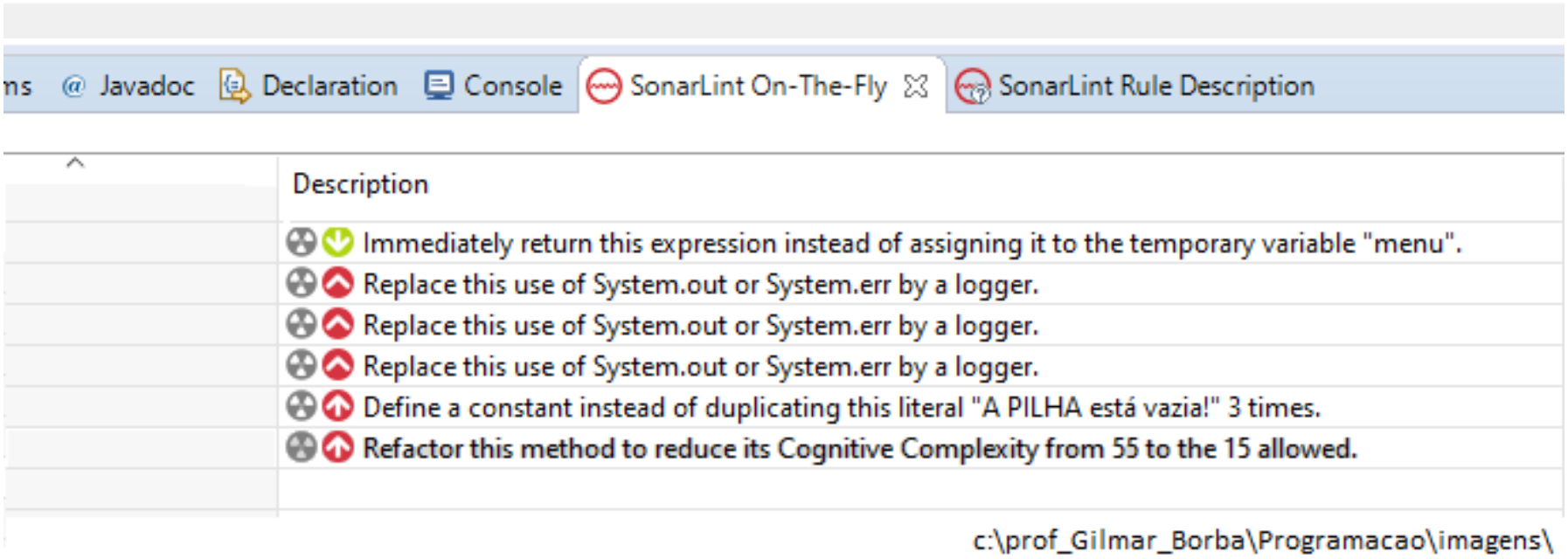
```
    if (achou==false) {
        Pilha novo = new Pilha();
        novo.numero = numero;
        novo.valor = Double.parseDouble(JOptionPane.showInputDialog("VALOR DA CHAPA", "0"));
        novo.pedido = Integer.parseInt(JOptionPane.showInputDialog("NÚMERO DO PEDIDO", "0"));
        novo.prox = topo;
        topo = novo;
    }
    return topo;
}
```



c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
    if ( !achou )    {
        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;
    }
    return topo;
}
```



SONAR

```
// INSERIR/EMPILHAR
if (op == 1) {
    topo = empilharChapas(topo);
}
// CONSULTAR CHAPAS CADASTRADAS
if (op == 2) {
    consultar(topo);
}
// DESEMPILHAR CHAPA
if (op == 3) {
    topo = desempilhar(topo);
}
// ESVAZIAR PÁTIO
if (op == 4) {
    topo = esvaziar(topo);
}
// QUANTIDADE E CHAPAS
if (op == 5) {
    quantidade(topo);
}
// FILTRAR CHAPAS CADASTRADAS - DESAFIO 3
if (op == 6) {
    filtrar(topo);
}
```

Description
Remove this unused "aux" local variable.
Immediately return this expression instead of assigning it to the temporary variable "menu".
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Define a constant instead of duplicating this literal "A PILHA esta vazia!" 3 times.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

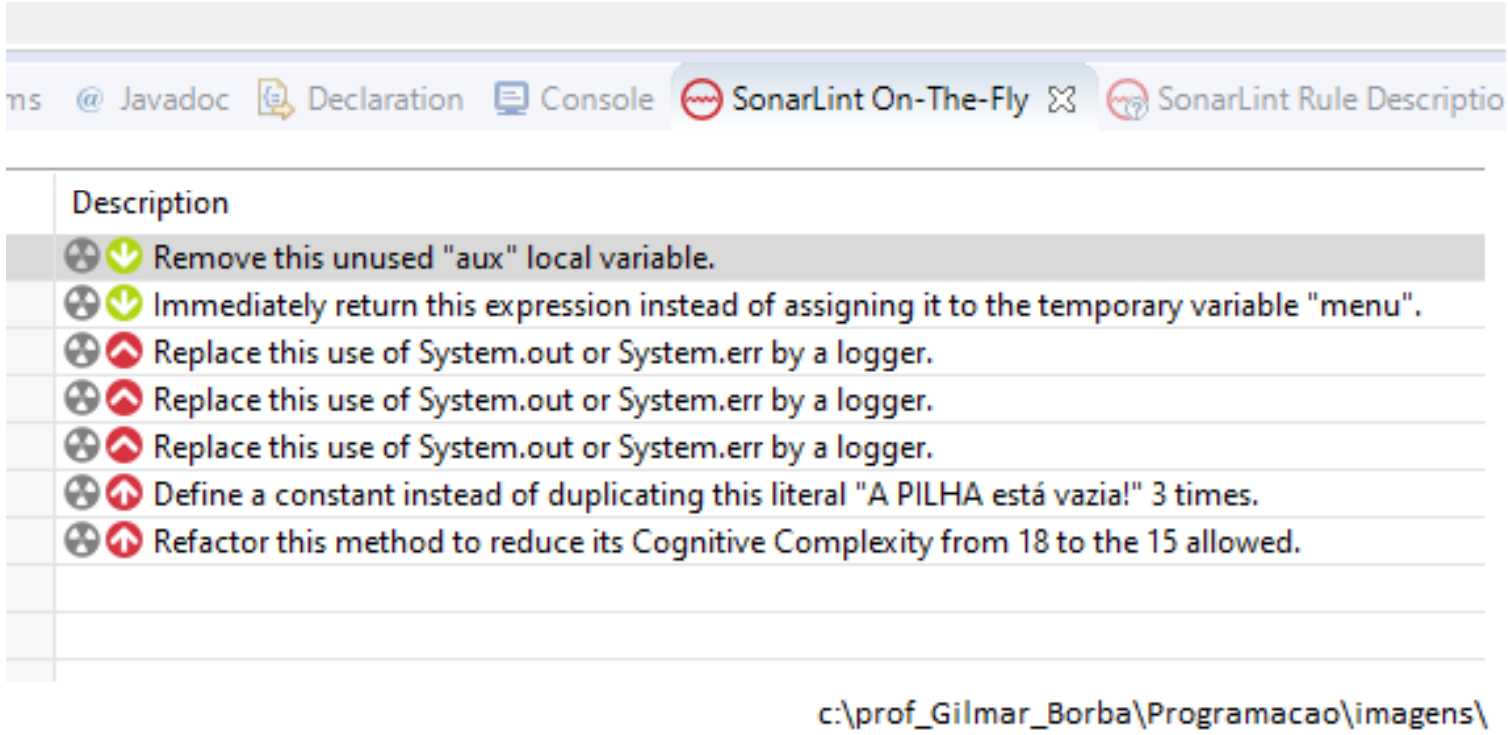
SONAR

```
// INSERIR/EMPILHAR
if (op == 1) {
    topo = empilharChapas(topo);
}
// CONSULTAR CHAPAS CADASTRADAS
if (op == 2) {
    consultar(topo);
}
// DESEMPILHAR CHAPA
if (op == 3) {
    topo = desempilhar(topo);
}
// ESVAZIAR PÁTIO
if (op == 4) {
    topo = esvaziar(topo);
}
// QUANTIDADE E CHAPAS
if (op == 5) {
    quantidade(topo);
}
// FILTRAR CHAPAS CADASTRADAS - DESAFIO 3
if (op == 6) {
    filtrar(topo);
}
```

Description
Remove this unused "aux" local variable.
Immediately return this expression instead of assigning it to the temporary variable "menu".
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Define a constant instead of duplicating this literal "A PILHA esta vazia!" 3 times.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

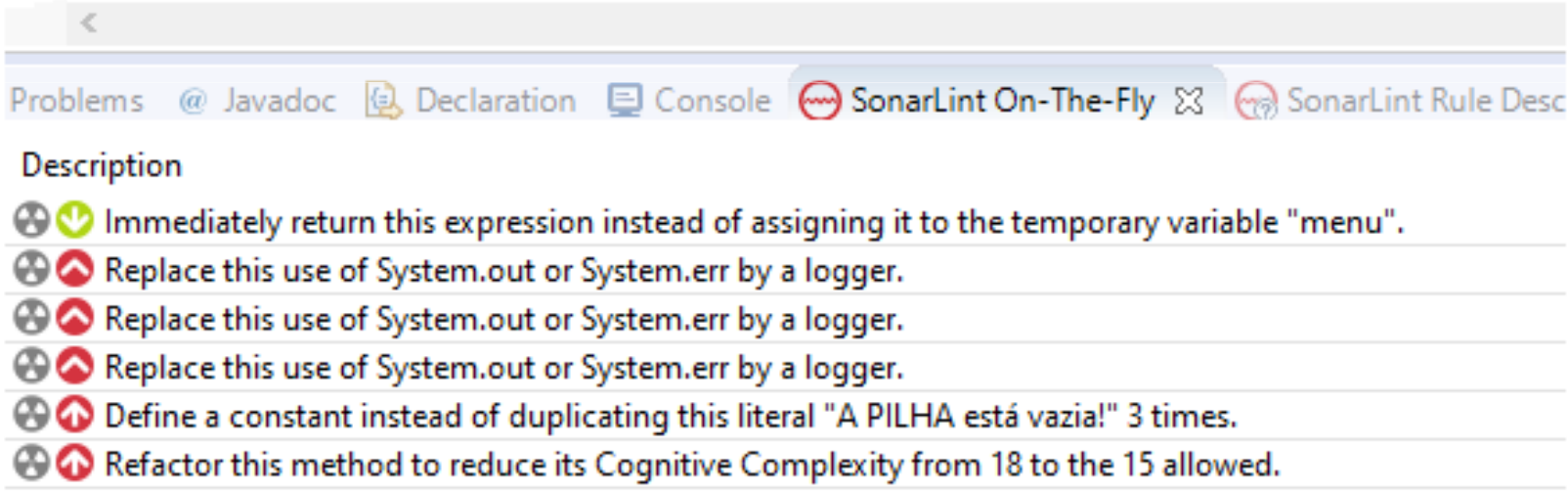
SONAR

```
public static void main(String[] args) {
    Pilha topo = null;
    Pilha aux;
    int op = 0;
```



SONAR

```
    private Pilha prox;
}
static Pilha topo = null;
static Pilha aux;

public static void main(String[] args) {

    int op = 0;
```



c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```java
// STRING - OPÇÕES MENU
private static String opcoesMenu() {
    String menu = "\nMENU DE OPCÕES\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;
}
```

Description	Resource
Immediately return this expression instead of assigning it to the temporary variable "menu".	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java
Define a constant instead of duplicating this literal "A PILHA está vazia!" 3 times.	TestePilha.java
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.	TestePilha.java

c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
// STRING - OPÇÕES MENU
private static String opcoesMenu() {
    return  "\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";
}
```

Description
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Define a constant instead of duplicating this literal "A PILHA está vazia!" 3 times.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
private static void filtrar(Pilha topo) {
    Pilha aux;
    if (topo == null) {
        mensagem("A PILHA está vazia!");
```

Description
Define a constant instead of duplicating this literal "A PILHA está vazia!" 3 times.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
private static void filtrar(Pilha topo) {
    Pilha aux;
    if (topo == null) {
        mensagem(PILHAVAZIA);
```

Description
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

c:\prof_Gilmar_Borba\Programacao\imagens\

SONAR

```
    } while (op != 7);

    Svstem.out.println(">>> PROGRAMA FINALIZADO!");

}
```

Description
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

SONAR

```
} while (op != 7);

JOptionPane.showMessageDialog(null,"PROGRAMA FINALIZADO",
                  "Mensagem" , JOptionPane.INFORMATION_MESSAGE);
```

Description
Replace this use of System.out or System.err by a logger.
Replace this use of System.out or System.err by a logger.
Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

Criando um arquivo de LOG (opcional)

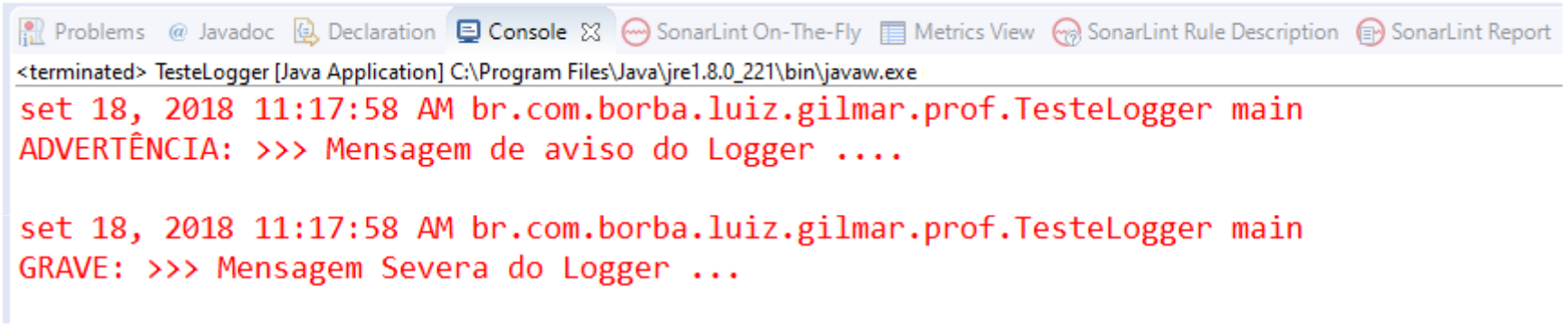
Refatoração

```
package br.com.borba.luiz.gilmar.prof;

import java.io.File;
import java.io.IOException;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class Log {
    public Logger logger;
    FileHandler manipuladorArquivo;

    public Log(String nomeArquivo) throws SecurityException, IOException {
        File f = new File(nomeArquivo);
        if (!f.exists()) {
            f.createNewFile();
        }
        manipuladorArquivo = new FileHandler(nomeArquivo, true);
        logger = Logger.getLogger("test");
        logger.addHandler(manipuladorArquivo);
        SimpleFormatter formatter = new SimpleFormatter();
        manipuladorArquivo.setFormatter(formatter);
    }
}
```

Refatoração

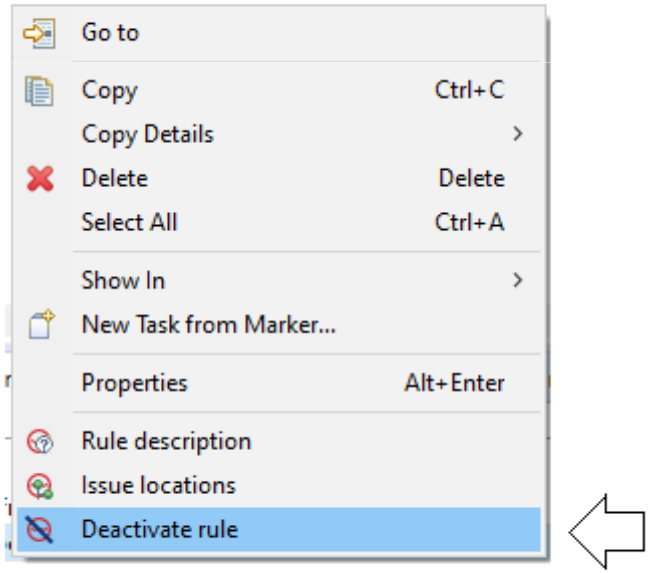
```
package br.com.borba.luiz.gilmar.prof;

import java.util.logging.Level;

public class TesteLogger {
    public static void main(String args[]) {
        try {
            Log meuLog = new Log("log.txt");
            meuLog.logger.setLevel(Level.WARNING);
            meuLog.logger.info("Informações do fornecedor: LOG NERVOSO!");
            meuLog.logger.info("Prof. Gilmar Borba");
            meuLog.logger.warning(">>> Mensagem de aviso do Logger ....\n");
            meuLog.logger.severe(">>> Mensagem Severa do Logger ...\n");
        } catch (Exception e) {
        }
    }
}
```

Refatoração



Refatoração

Caso a sugestão (como LOGGER) não tenha aplicabilidade, ela pode ser ignorada *(escolha: desativar regra)*.



c:\prof_Gilmar_Borba\Programacao\imagens\

Alteração de “if” por “switch/Case”

Refatoração

```
switch (op) {
case 1:
    topo = empilharChapas(topo);
    break;
case 2:
    if (topo == null) {
        JOptionPane.showMessageDialog(null, "Mensagem",
                "Pilha Vazia!", JOptionPane.INFORMATION_MESSAGE);
    } else {
        consultar(topo);
    }
    break;
case 3:
    topo = desempilhar(topo);
    break;
case 4:
    topo = esvaziar(topo);
    break;
case 5:
    quantidade(topo);
    break;
case 6:
    filtrar(topo);
    break;
case 7:
    System.exit(0);
    break;
default:
    mensagem("Opção Inválida!");
}
```

Description	Resource
Replace this use of System.out or System.err by a logger.	TestePilha.java
Replace this use of System.out or System.err by a logger.	TestePilha.java

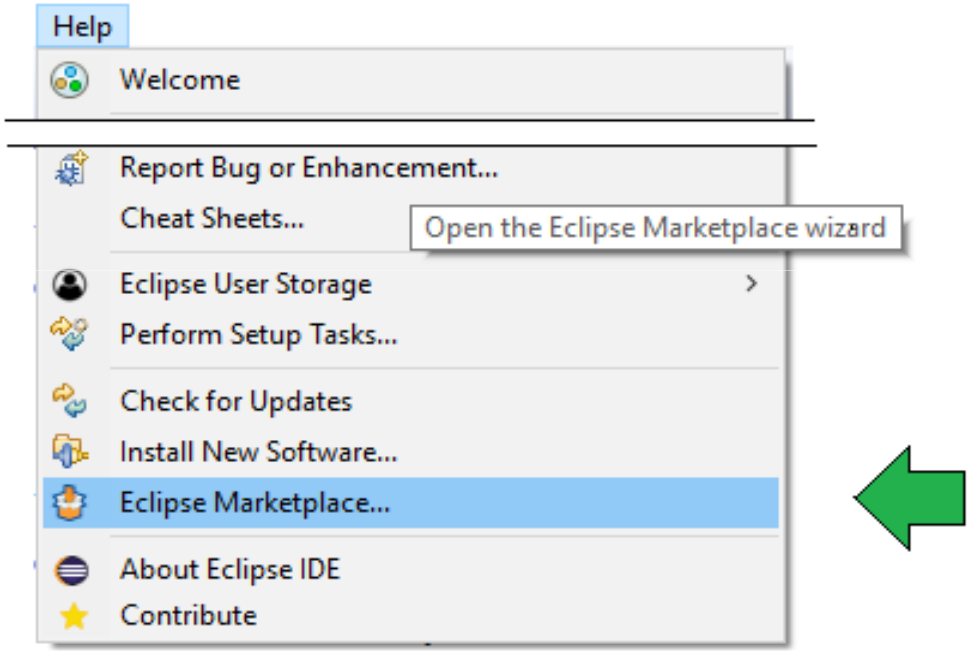
c:\prof_Gilmar_Borba\Programacao\imagens\

INSTALANDO SONAR

(Eclipse)

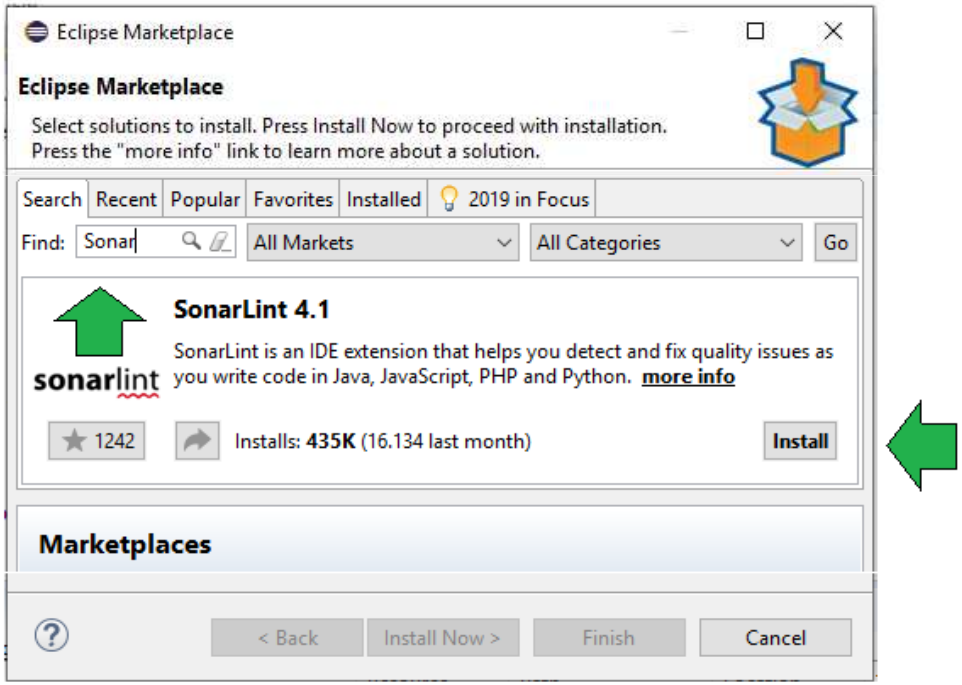
Instalando o Plugin Sonar Lint

Escolha a opção Help, em seguida Eclipse Marketplace



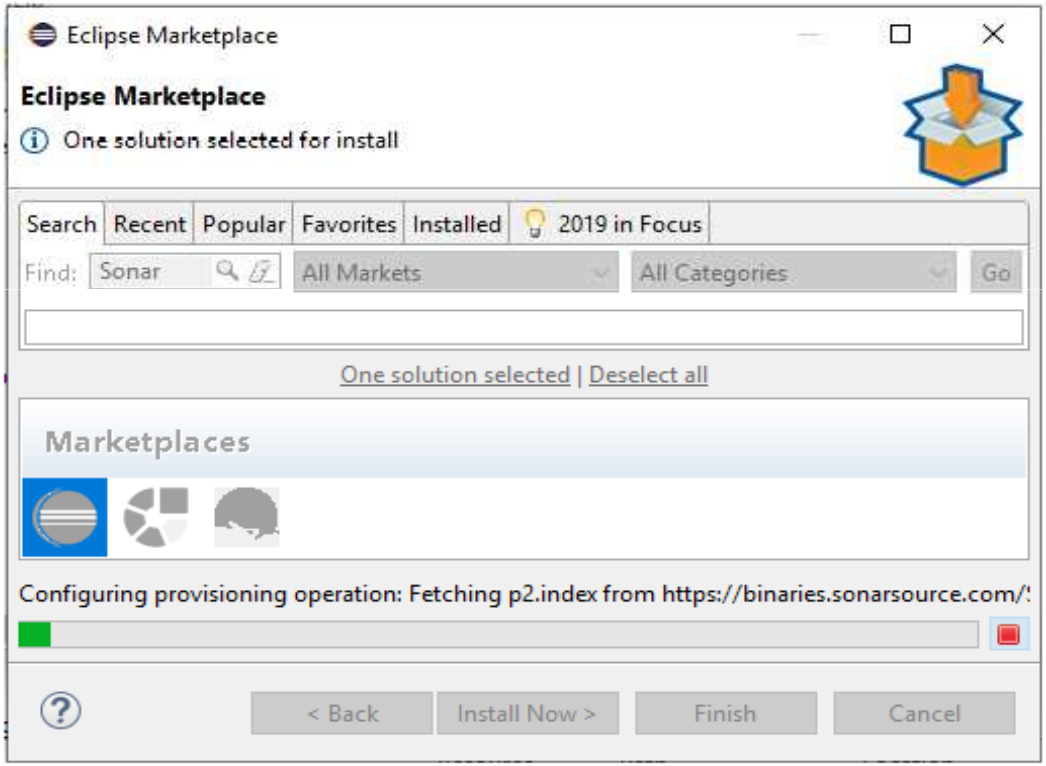
Instalando o Plugin Sonar Lint

Escolha Localizar, digite “Sonar”, veja a imagem, escolher instalar.



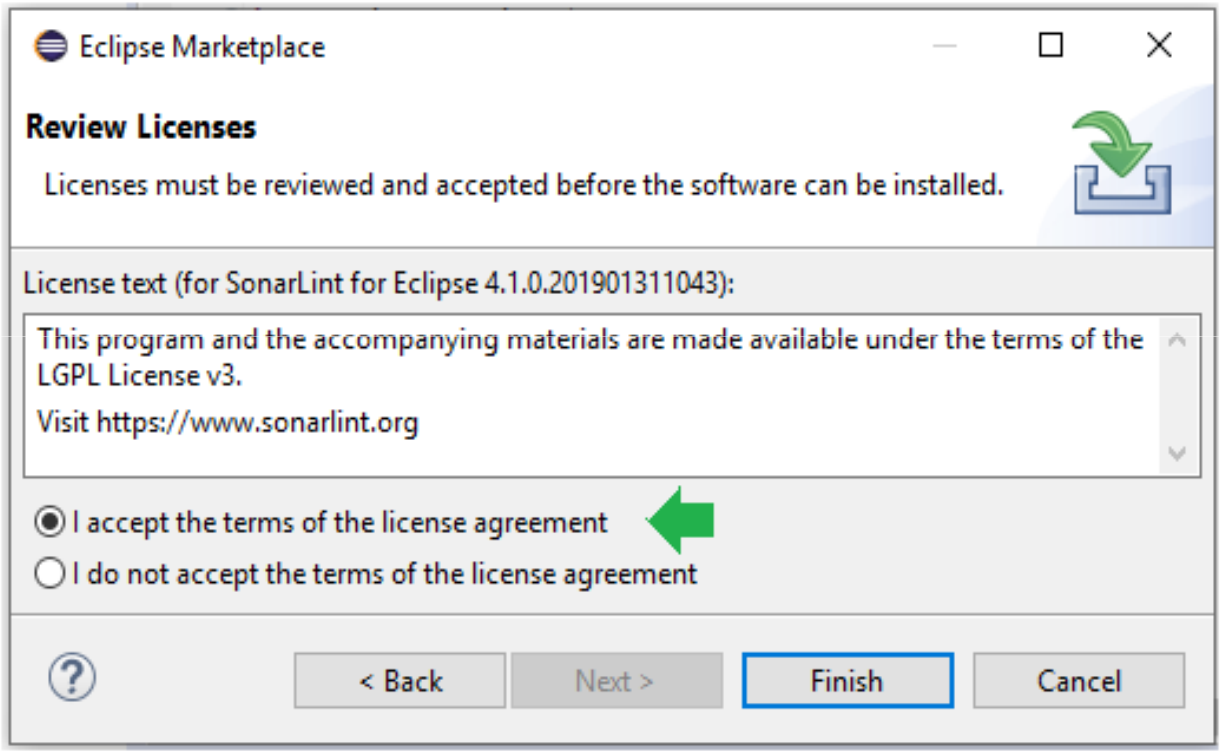
Instalando o Plugin Sonar Lint

Escolha Localizar, digite “Sonar”, veja a imagem.



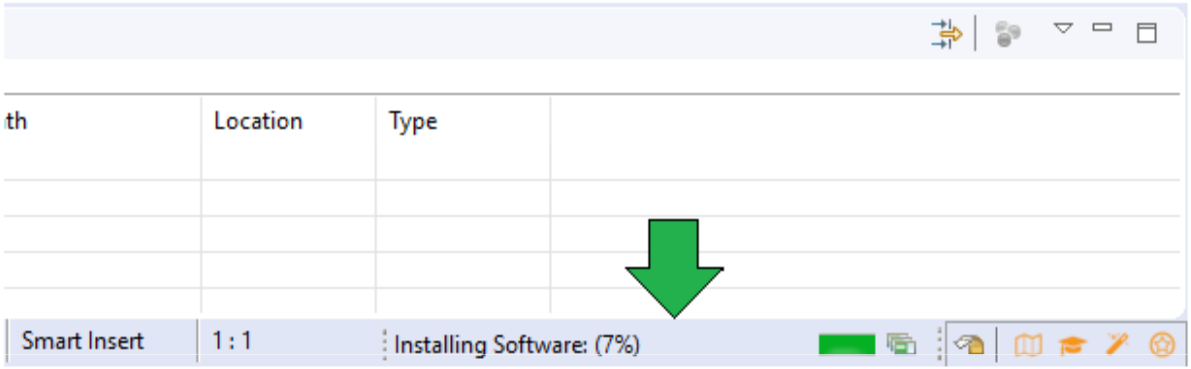
Instalando o Plugin Sonar Lint

Aceite os termos da licença do Plugin.

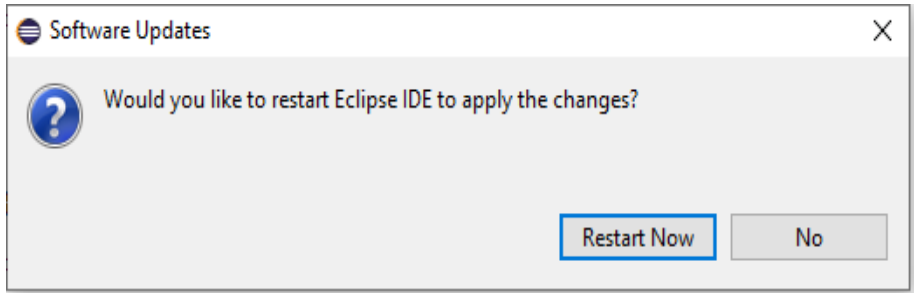


Instalando o Plugin Sonar Lint

Aguarde a instalação, alguns minutos.

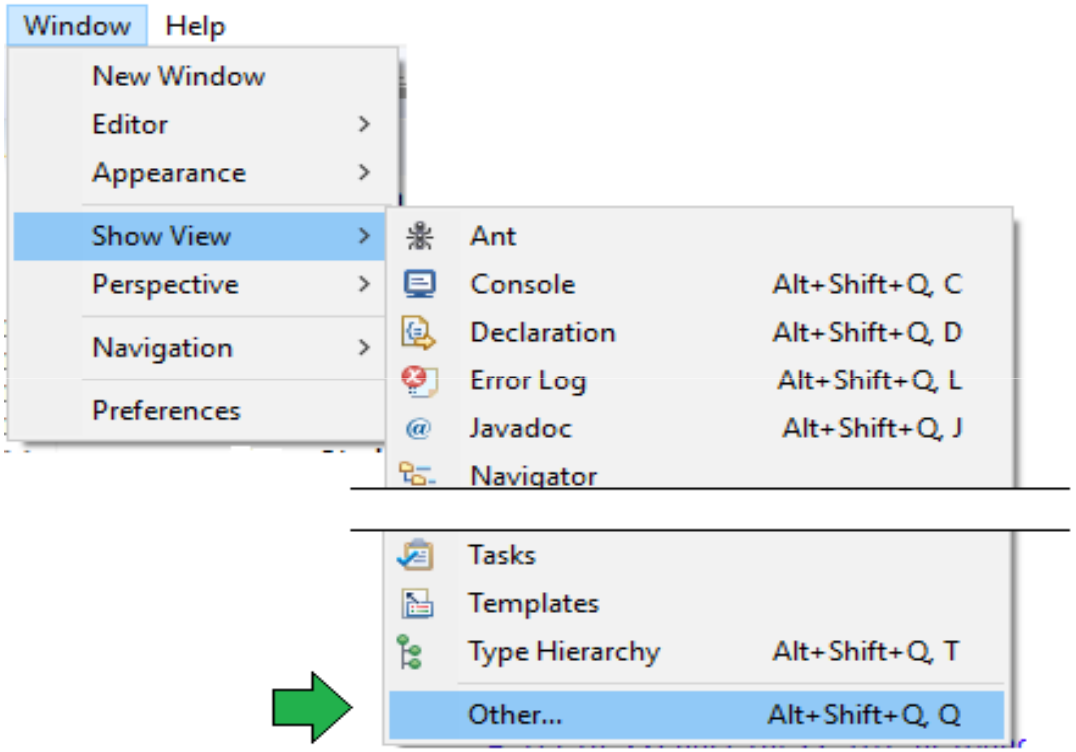


Confirme a restauração do Windows.



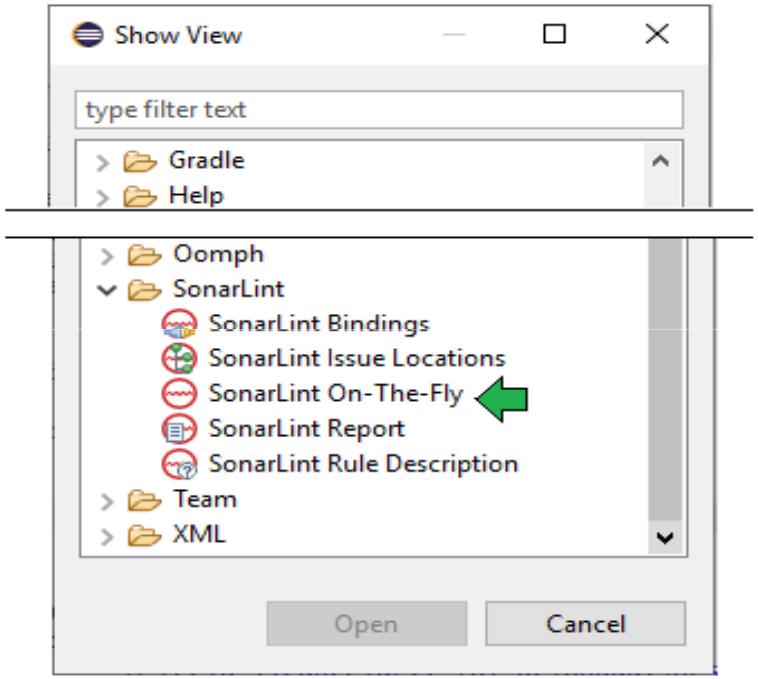
Instalando o Plugin Sonar Lint

Habilitar o Sonar Lint On-The-Fly. Escolher Windows + Show View + Other.



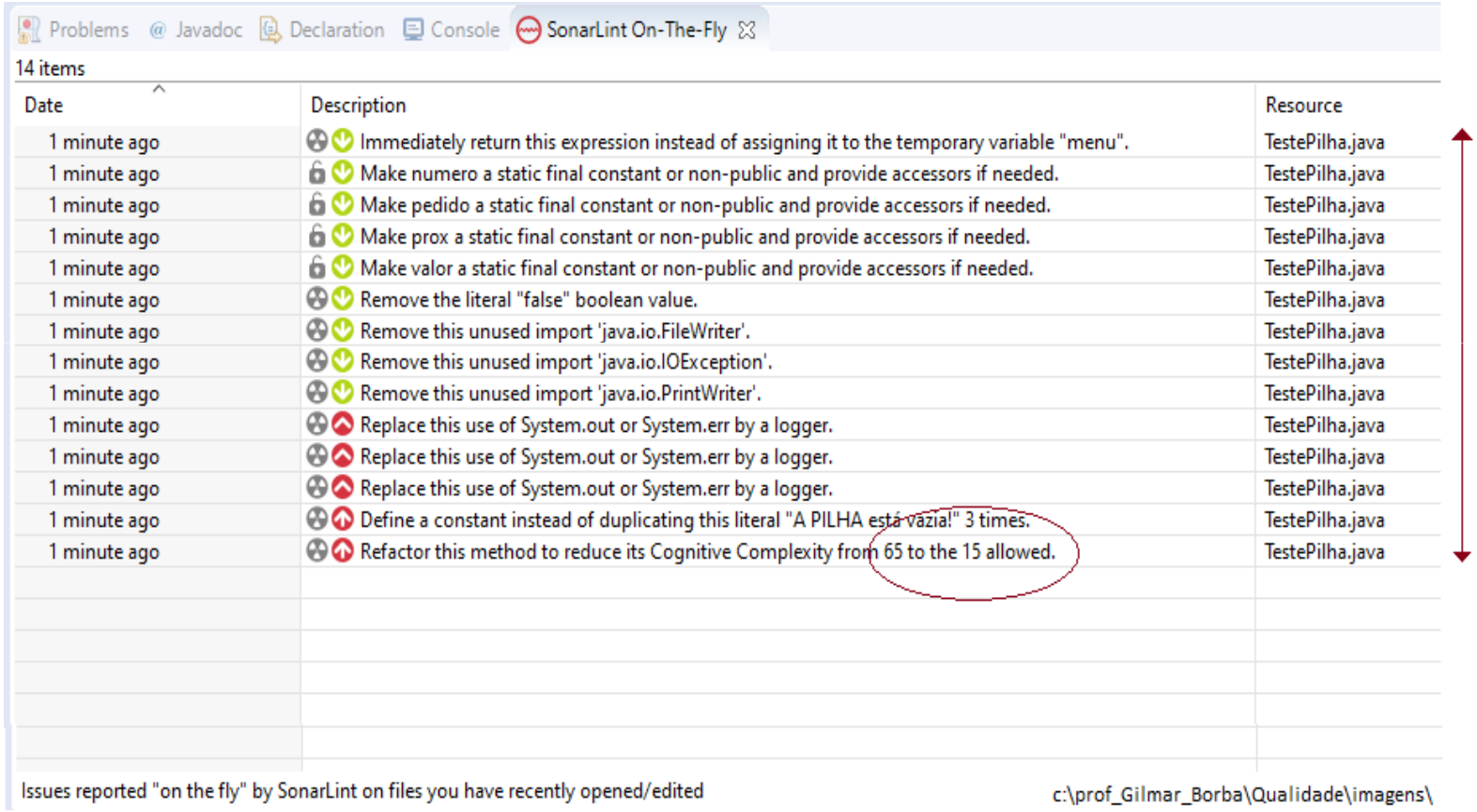
Instalando o Plugin Sonar Lint

Escolher Sonar Lint On-The-Fly.



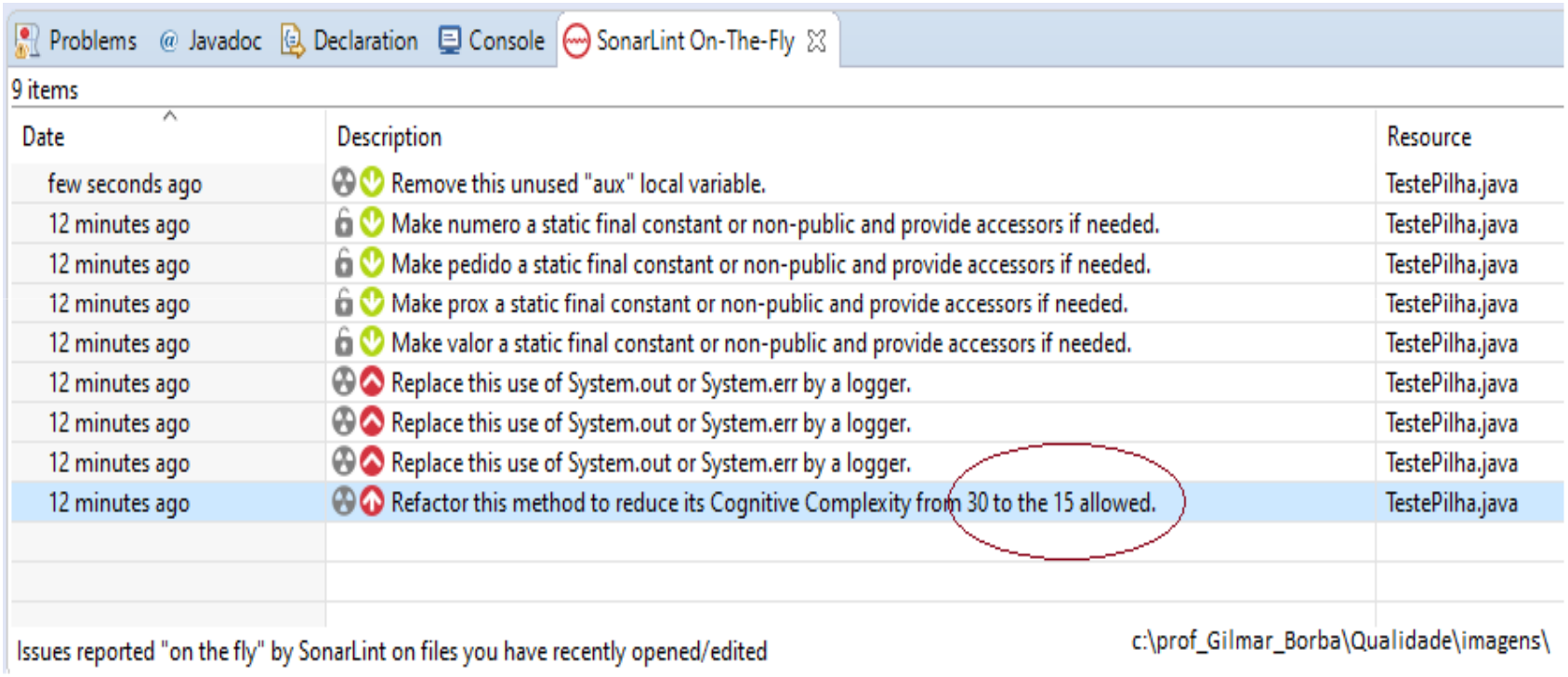
Instalando o Plugin Sonar Lint

Visualize as indicações do SonarLint.



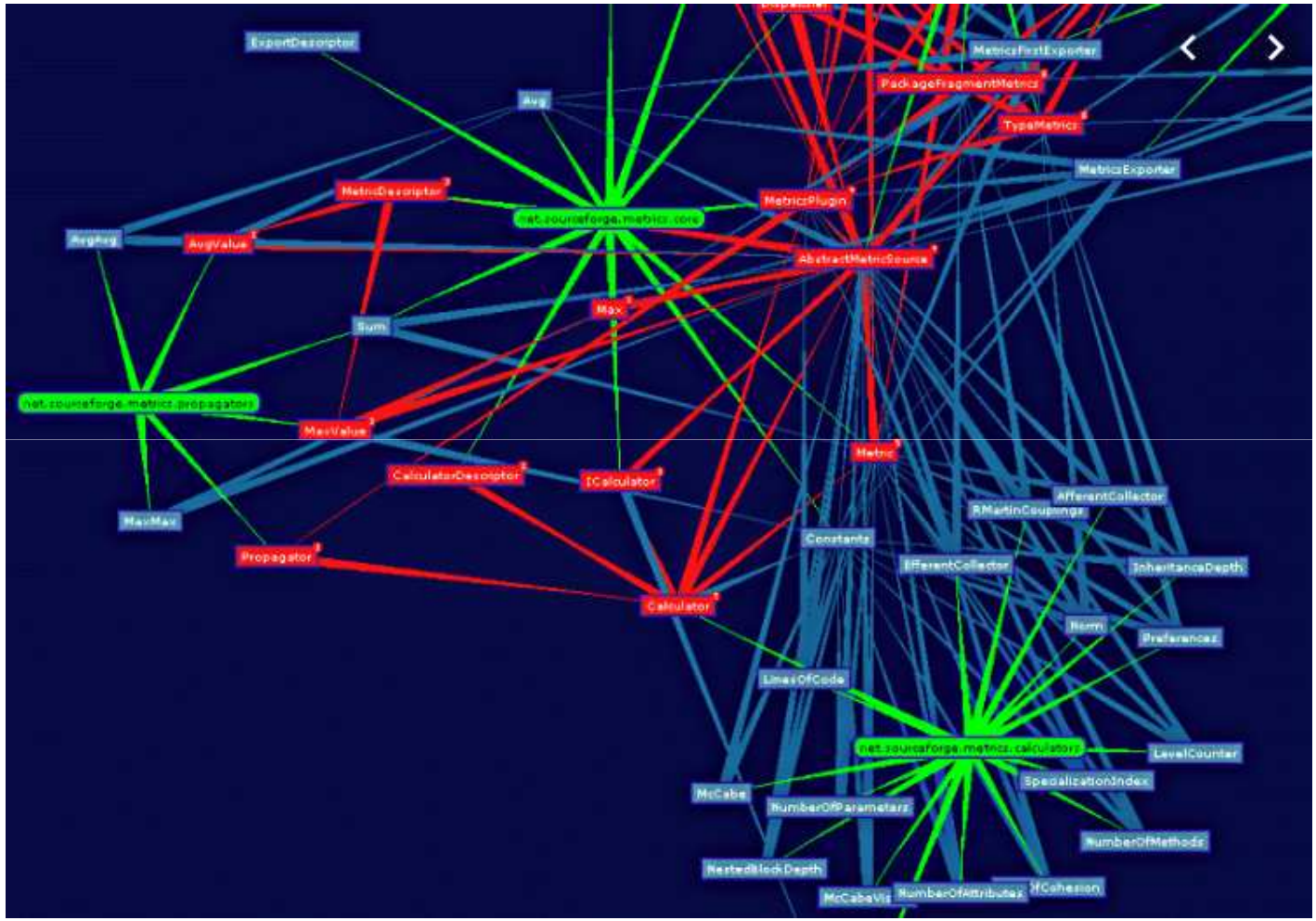
Instalando o Plugin Sonar Lint

Faça as refatorações com base nas indicações do SonarLint.



Complexidade Ciclomática

METRICS



Exibição da complexidade de código no Metrics (Dependency Graph View)

MÉTRICS (plugin)

Complexidade ciclomática
(complexidade condicional)

É uma métrica que indica a complexidade de um programa de computador.

Mede o número de caminhos independentes de um programa/código de software.

É baseada a partir de um grafo de fluxo de controle. Os nós correspondem a grupos indivisíveis de comandos.

É aplicada a módulos de um programa, métodos e classes.

Foi desenvolvida por Thomas J. McCabe em 1976.

MÉTRICS (plugin)

Complexidade ciclomática

(complexidade condicional)

“Sabemos que um dos grandes vilões na hora de mantermos um sistema legado são os trechos de código difíceis ou complicados de entender. Todo programador já se deparou com [...] códigos com péssimos nomes de variáveis ou muitas linhas de código em um único método, sem contar o excesso de responsabilidade. [...] um dos maiores responsáveis pela dificuldade em entender o que um código faz é o simples if *(na: case, switch, while ...)*

Com essa simples instrução, o desenvolvedor faz com que o mesmo método possa responder de duas, três, quatro, N, ..., maneiras diferentes, de acordo com certas condições! E pior, o número de caminhos diferentes pode crescer muito rápido!. ”

Complexidade ciclomática

(complexidade condicional)

"Devemos, quase sempre, manter esse número de caminhos **o menor possível**."

"Não há um número ideal para a complexidade de um método. Mas fato é que uma complexidade ciclomática de 20 ou 30 é demais e esse método deve ser reescrito. Mesmo quebrando-o em dois, é interessante observar a complexidade dos métodos de uma classe em conjunto." Maurício AnicheMsc. Ciência da Computação – USP

Fonte: **Medindo a complexidade do seu código**. Maurício Aniche. Postado em 01/03/2012.
Disponível em: http://blog.caelum.com.br/medindo-a-complexidade-do-seu-codigo/
Acessado em: agosto de 2015 / Janeiro de 2019

MÉTRICS (plugin)

Complexidade ciclomática

ReservaPassagemArea (Sem refatorar)

Problems | Javadoc | Declaration | Console | Metrics - TestePilha.java

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	

MÉTRICS (plugin)

Complexidade ciclomática

ReservaPassagemArea (refatorado)

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 packageFragr	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	

MÉTRICS (plugin)

Instalando Plugin do Metrics no Eclipse

No menu **HELP**

Escolher: Install New Software

Clicar sobre o botão **ADD** (na caixa de texto Working With)

Na caixa de diálogo: **Add Repository**

Informar *Metrics* (na opção name)

Informar: *http://metrics2.sourceforge.net/update* (na opção location)

Acionar **OK**

Instalando o Metrics

MÉTRICS (plugin)

Instalando plugin do *METRICS* no Eclipse

Acionar Select All

Acionar NEXT (Ainda na caixa de diálogo Available Software)
(se a opção NEXT estiver desabilitada selecione e tire a seleção de Hide items that are already installed)

Acionar NEXT (Na caixa de detalhe Install Details)

Observe que Metrics plugin for Eclipse version 1.3.6 foi reconhecido

Aceitar os termos de uso da licença (Na caixas de diálogo Review Licenses) Acionar FINISH

... O Plugin será instalado em alguns segundos
... talvez um minuto ...

Continuar (OK) a instalação caso ocorra a mensagem Warning! Acione **YES** para reiniciar o Eclipse.

MÉTRICS (plugin)

Para Visualizar o Plugin o *METRICS* no Projeto

Acesse: Menu Window -> Show View -> Other
Expanda o ítem da árvore: Metrics
Selecione: Metrics View
Clique: **Ok**
Irá aparecer uma janela no rodapé da IDE

Para Ativar o Plugin do *METRICS* no Projeto

Clique com o botão direito do mouse sobre o nome do projeto no Explorer do Eclipse
Clique: Properties
Clique no Metrics na árvore de propriedades
Selecione o checkbox: **Enable** Metrics

SONAR x METRICS

Complexidade Cognitiva X Complexidade Ciclomática

MÉTRICS (plugin) X Sonar

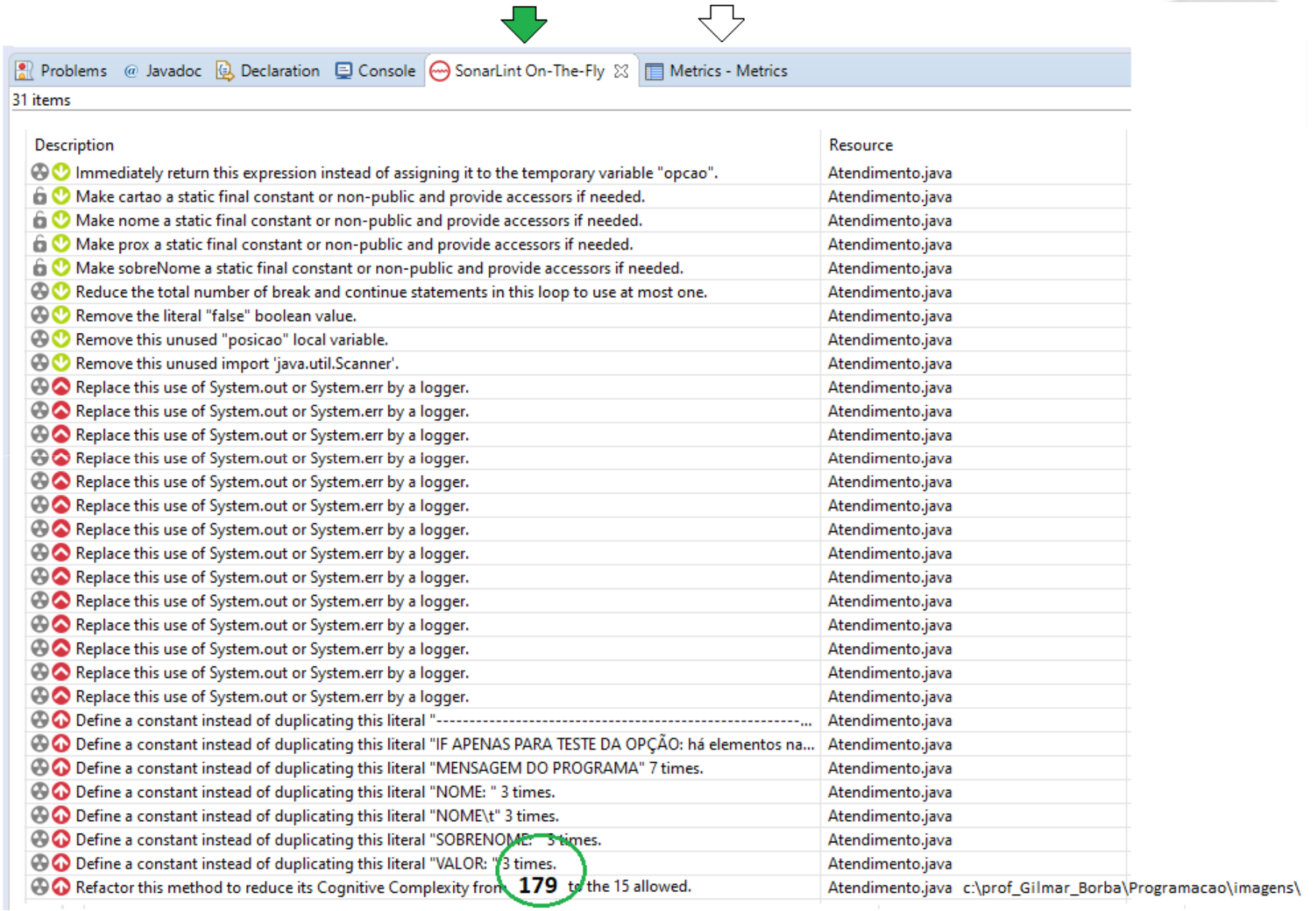
Complexidade cognitiva (Sonar)

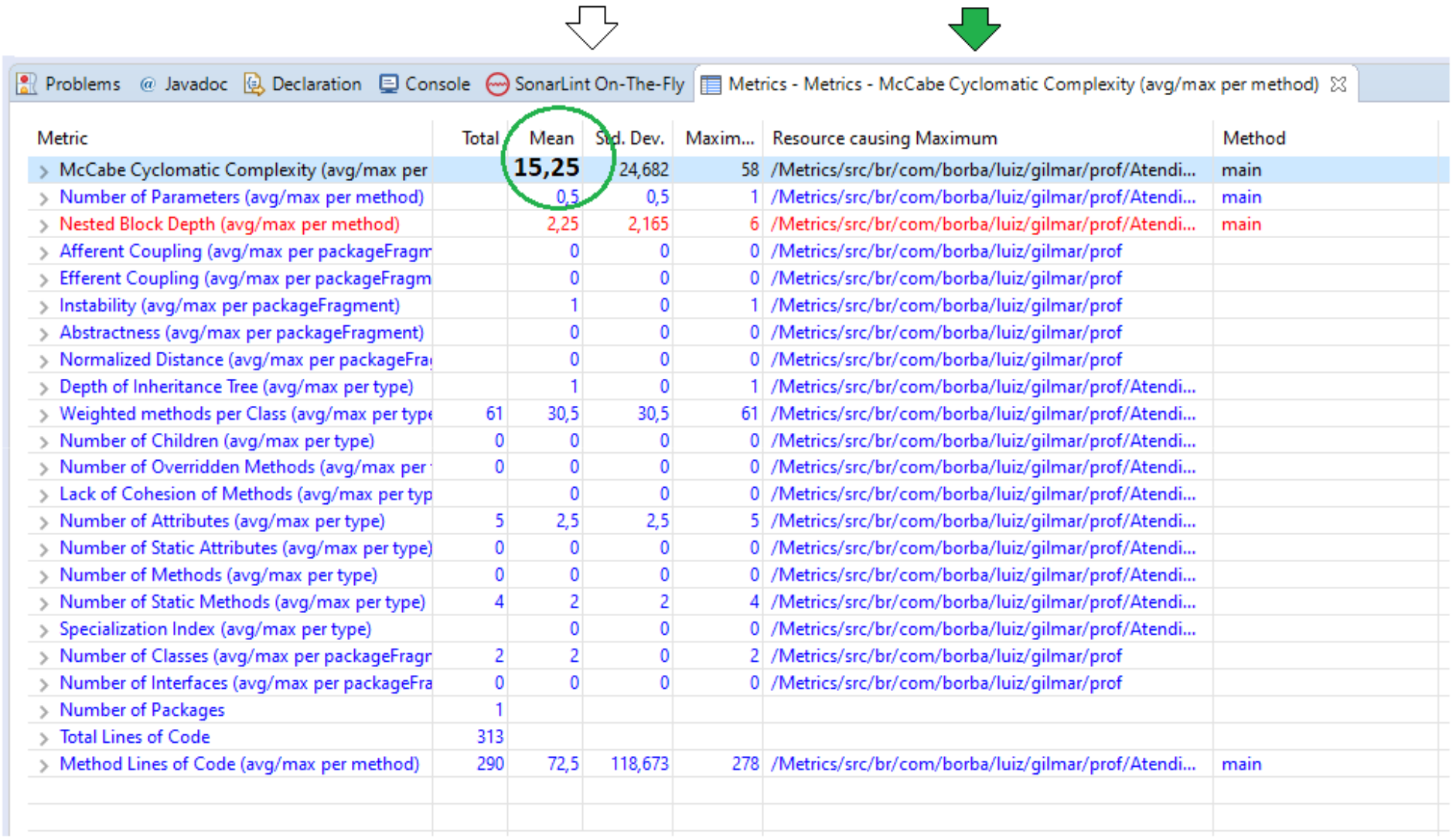
- É uma técnica de medida de qualidade recente.
- Medida de quão difícil é entender uma unidade de código.
- Está associada à complexidade de um software.
- Está associada a dificuldade de manutenção.
- Está associada a dificuldade de ler e entender um determinado código.

Complexidade ciclomática (Metrics)

- É uma técnica de medida de qualidade mais antiga.
- Está associada a dificuldade ou esforço para se fazer testes de unidade.
- É uma técnica precisa para indicar dificuldade de "*testabilidade*".
- Métodos com a mesma complexidade ciclomática podem apresentar diferenças na dificuldade de leitura e entendimento.

Fonte: Complexidade Cognitiva
Disponível em: http://artesoftware.com.br/complexidade-cognitiva/
Acessado: Janeiro de 2019


Problems
Javadoc
Declaration
Console
SonarLint On-The-Fly
Metrics - Metrics
31 items
Description
Resource
Immediately return this expression instead of assigning it to the temporary variable "opcao". Atendimento.java
Make cartao a static final constant or non-public and provide accessors if needed. Atendimento.java
Make nome a static final constant or non-public and provide accessors if needed. Atendimento.java
Make prox a static final constant or non-public and provide accessors if needed. Atendimento.java
Make sobreNome a static final constant or non-public and provide accessors if needed. Atendimento.java
Reduce the total number of break and continue statements in this loop to use at most one. Atendimento.java
Remove the literal "false" boolean value. Atendimento.java
Remove this unused "posicao" local variable. Atendimento.java
Remove this unused import 'java.util.Scanner'. Atendimento.java
Replace this use of System.out or System.err by a logger. Atendimento.java
Define a constant instead of duplicating this literal "---------- Atendimento.java
Define a constant instead of duplicating this literal "IF APENAS PARA TESTE DA OPÇÃO: há elementos na... Atendimento.java
Define a constant instead of duplicating this literal "MENSAGEM DO PROGRAMA" 7 times. Atendimento.java
Define a constant instead of duplicating this literal "NOME: " 3 times. Atendimento.java
Define a constant instead of duplicating this literal "NOME\t" 3 times. Atendimento.java
Define a constant instead of duplicating this literal "SOBRENOME: " 3 times. Atendimento.java
Define a constant instead of duplicating this literal "VALOR: " 3 times. Atendimento.java
Refactor this method to reduce its Cognitive Complexity from 179 to the 15 allowed. Atendimento.java c:\prof_Gilmar_Borba\Programacao\imagens\



Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per		15,25	24,682	58	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	main
Number of Parameters (avg/max per method)		0,5	0,5	1	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	main
Nested Block Depth (avg/max per method)		2,25	2,165	6	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	main
Afferent Coupling (avg/max per packageFragm		0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Efferent Coupling (avg/max per packageFragm		0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Instability (avg/max per packageFragment)		1	0	1	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Abstractness (avg/max per packageFragment)		0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Normalized Distance (avg/max per packageFra		0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Depth of Inheritance Tree (avg/max per type)		1	0	1	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Weighted methods per Class (avg/max per type	61	30,5	30,5	61	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Children (avg/max per type)	0	0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Overridden Methods (avg/max per	0	0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Lack of Cohesion of Methods (avg/max per typ		0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Attributes (avg/max per type)	5	2,5	2,5	5	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Static Attributes (avg/max per type)	0	0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Methods (avg/max per type)	0	0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Static Methods (avg/max per type)	4	2	2	4	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Specialization Index (avg/max per type)		0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	
Number of Classes (avg/max per packageFragr	2	2	0	2	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/Metrics/src/br/com/borba/luiz/gilmar/prof	
Number of Packages	1					
Total Lines of Code	313					
Method Lines of Code (avg/max per method)	290	72,5	118,673	278	/Metrics/src/br/com/borba/luiz/gilmar/prof/Atendi...	main

c:\prof_Gilmar_Borba\Programacao\imagens\

REFERÊNCIAS PRINCIPAIS

Complexidade Cognitiva
Disponível em: http://artesoftware.com.br/complexidade-cognitiva/
Acessado: Janeiro de 2019

Medindo a complexidade do seu código. Maurício Aniche. Postado em 01/03/2012.
Disponível em: http://blog.caelum.com.br/medindo-a-complexidade-do-seu-codigo/
Acessado em: agosto de 2015 / Janeiro de 2019

Refatoração de Software. DCC/IME-USP. Prof. Dr. Marcos L. Chain - EACH - USP
Disponível em: http://www.ic.unicamp.br
Acessado em: janeiro/2015

Refatoração: Melhorando a Qualidade de código Pré-Existente. Cursos de Verão 2007 - IME/USP. Alexandre Freire e&Paulo Cheque.
Disponível em: http://ccsl.ime.usp.br
Acessado em: Dezembro/2014

Refatoração

fim ...