

UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
ENGENHARIA DE COMPUTAÇÃO

ANTÔNIO LUÍS PEREIRA JUNIOR
GUILHERME ALEXANDER DA SILVA RIBEIRO
ITALO DOMINGOS RODRIGUES MATOS
LARYSSA RIBEIRO SOUSA
RICARDO GONÇALVES CARDOSO

TUTORIAL SOBRE THREADS E AWT/SWING

São Luís

2020

Threads em Java

Uma *thread* é considerada uma sucessão de comandos sendo executados por um programa em paralelo a outras instruções. De forma convencional podemos dizer que *thread* é uma maneira de um software executar diversas tarefas internas simultaneamente. Quando se fala em *threads*, o objetivo é trabalhar de forma paralela.

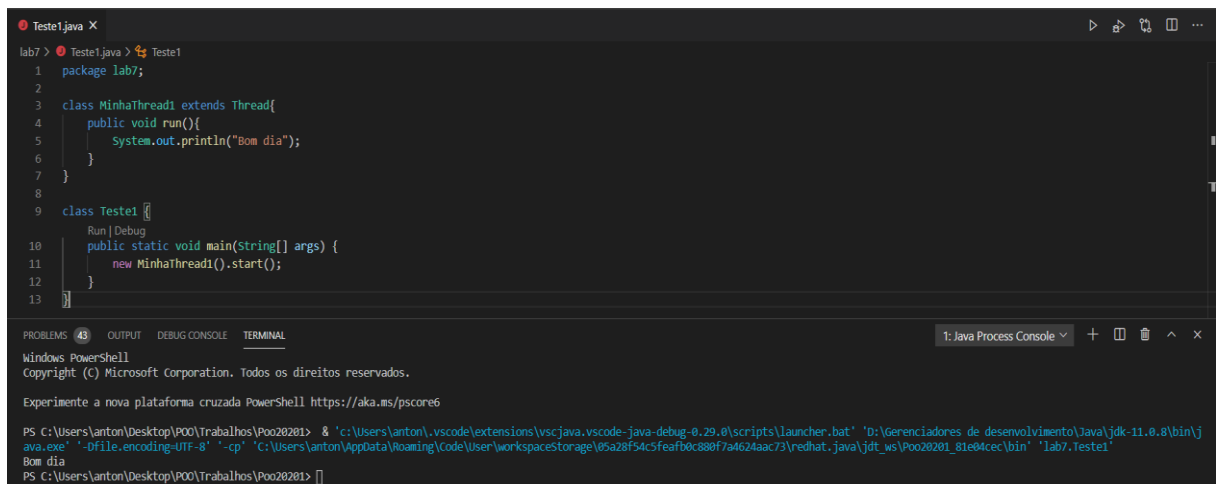
Threads Java são implementadas pela classe *Thread* do pacote *java.lang*. Esta classe implementa um encapsulamento independente de sistema, isto é, a implementação real de *threads* é oferecida pelo sistema operacional. A classe *Thread* oferece uma interface unificada para todos os sistemas. Portanto, uma mesma implementação da *Thread* pode fazer com que a aplicação proceda de forma diferentes em cada sistema.

As *threads* possuem os seguintes estados:

- Bloqueado: Ocorre no momento que uma solicitação do executável não pode ser finalizada por espera de um determinado recurso, logo a mesma deve esperar até que o recurso esteja disponível.
- Novo: Quando um thread é criada e está aguardando para ser executada.
- Executável: Thread está executando suas operações.
- Espera: Ocorre quando uma *thread* está aguardando outra thread executar sua tarefa.
- Terminado: Quando todas as suas operações foram finalizadas pelo fim das instruções ou por um erro.

Existem duas formas de criarmos uma *thread* em Java. Pode-se criar uma classe para herdar os atributos e métodos da classe *Thread*, ou criar uma classe que implementa a interface *Runnable*.

- Por herança:



```
Teste1.java X
lab7 > Teste1.java > Teste1
1 package lab7;
2
3 class MinhaThread1 extends Thread{
4     public void run(){
5         System.out.println("Bom dia");
6     }
7 }
8
9 class Teste1 {
10     Run | Debug
11     public static void main(String[] args) {
12         new MinhaThread1().start();
13     }
14 }

PROBLEMS 43 OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\anton\Desktop\P00\Trabalhos\P0020201> & 'c:\Users\anton\.vscode\extensions\vscode-java-debug-0.29.0\scripts\launcher.bat' 'D:\Gerenciadores de desenvolvimento\java\jdk-11.0.8\bin\java.exe' -Dfile.encoding=UTF-8 -cp 'C:\Users\anton\AppData\Local\Code\User\workspaceStorage\05a28f34c3feaf80c88077a4624aac73\redhat.java\jdt_ws\P0020201_81e04cec\bin' 'lab7.Teste1'
Bom dia
PS C:\Users\anton\Desktop\P00\Trabalhos\P0020201>
```

Figura 1: Implementação da Thread por Herança

- Por interface:

```
package Thread;

public class Executal {
    Run | Debug
    public static void main(String[] args){
        Thread executarThreadTeste = new Thread(new ThreadTeste());
        executarThreadTeste.start();
        for(int i = 0; i < 50; i++){
            System.out.println("Contador Principal: " + i);
        }
    }
}

class ThreadTeste implements Runnable{
    public void run(){
        for(int i = 0; i < 50; i++){
            System.out.println("Contador Thread: " + i);
        }
    }
}
```

Figura 2: Implementação da Thread por Interface

```
Contador Principal: 0
Contador Principal: 1
Contador Principal: 2
Contador Thread: 0
Contador Thread: 1
Contador Principal: 3
Contador Thread: 2
Contador Thread: 3
Contador Thread: 4
Contador Principal: 4
Contador Thread: 5
Contador Thread: 6
Contador Thread: 7
Contador Thread: 8
Contador Thread: 9
Contador Thread: 10
Contador Thread: 11
```

Figura 3: Saída da Implementação da Thread por Interface

Alguns dos principais métodos da classe *Thread* podem ser vistos abaixo:

- **run():** É o método que executa as atividades de uma *thread*. Quando este método finaliza, a *thread* também termina.
- **start():** Método que dispara a execução de uma *thread*. Este método chama o método run() antes de terminar.
- **sleep(int x):** Método que coloca a *thread* para dormir por x milissegundos.
- **join():** Método que espera o término da *thread* para qual foi enviada a mensagem para ser liberada.
- **interrupt():** Método que interrompe a execução de uma *thread*.
- **interrupted():** Método que testa se uma *thread* está ou não interrompida.

AWT/Swing

Java Swing faz parte do Java Foundation Classes (JFC), usado para criar aplicativos baseados em janelas. Ele é construído na parte superior da AWT (Abstract Windowing Toolkit), é inteiramente escrito em Java.

Contrário do AWT, o Java Swing fornece componentes independentes e leves da plataforma. O pacote *javax.swing* fornece classes da API de *swing java* como *JButton*, *TextField*, *TextArea*, *JRadioButton*, *JCheckbox*, etc. Contrário do AWT, o Java Swing fornece componentes independentes e leves da plataforma.

O pacote *javax.swing* fornece classes para API de *swing java*, como *JButton*, *TextField*, *TextArea*, *JRadioButton*, *JCheckbox*, *JMenu*, *JColorChooser*, etc.

Diferença entre AWT e Swing

- Os componentes do AWT são dependentes de plataforma, já os componentes do Java Swing independentes de plataforma.
- Os componentes AWT são pesados, já os componentes do Java Swing leves.
- A AWT não suporta aparência e sensação plugáveis, o Swing suporta aparência e sensação plugáveis.
- O AWT fornece menos componentes do que o Swing, já o Swing fornece componentes mais poderosos, como tabelas, listas, rolagem, colorchooser, tabbepane etc.
- O AWT não segue o MVC(Model View Controller) onde o modelo representa dados, a visualização representa a apresentação e o controlador atua como interface entre o modelo e visualização, o Swing segue MVC.

JFC

As Classes da Fundação Java (JFC) são um conjunto de componentes de GUI que simplificam o desenvolvimento de aplicativos de desktop. As Classes Java Swing possuem a seguinte hierarquia.

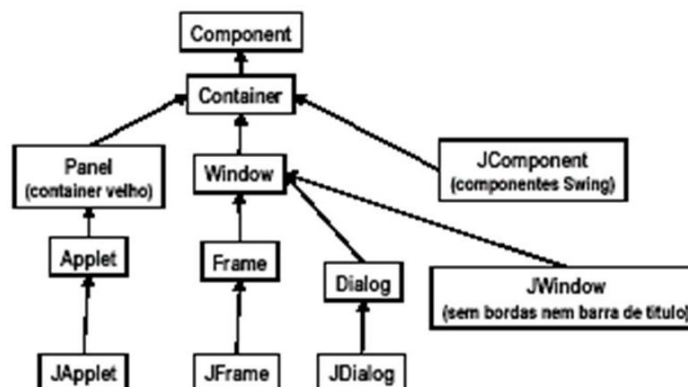


Figura 4: Hierarquia das Classes Swing

Métodos comumente usados da classe Componente

Os métodos da classe *Component* são amplamente utilizados no Java Swing que são dados abaixo:

public void add(Component c): adiciona um componente em outro componente.

public void setSize(int width, int height): define o tamanho do componente.

public void setLayout(LayoutMenager m): define o gerenciador de layout para o componente.

public void setVisible(boolean b): define a visibilidade do componente. Por padrão é falso.

Exemplos de Java Swing

Existem duas maneiras de criar um quadro: criando o objeto de classe *Frame* (associação) ou estendendo a classe *Frame* (herança). Podemos escrever o código de Java Swing dentro da *main()*, construtor ou qualquer outro método. Veremos um simples exemplo de Swing onde criaremos um botão e o adicionando no objeto *JFrame* dentro do método *main()*.

```
FirstSwingExample.java X
lab7 > FirstSwingExample.java > ...
1  package lab7;
2
3  import javax.swing.*;
4
5  public class FirstSwingExample {
6      Run | Debug
7      public static void main(String[] args) {
8          JFrame f = new JFrame();
9          JButton b = new JButton("click");
10         b.setBounds(130, 100, 100, 40);
11         f.add(b);
12         f.setSize(400, 500);
13         f.setLayout(null);
14         f.setVisible(true);
15     }
16 }
```

Figura 5: Código de Exemplo do Java Swing

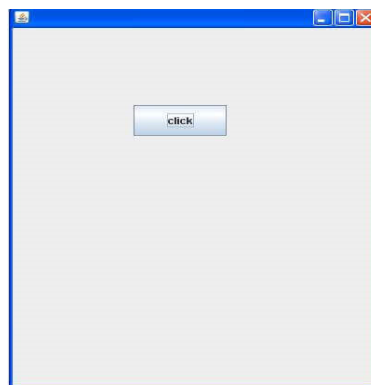


Figura 6: Execução do Código de Exemplo do Java Swing