**Resolução – 7° Lista de Exercícios – Threads com Swing**

**1)**

O método “SwingUtilities.invokeLater” permite que a função “run()” de dentro do método seja executada **assíncronamente** na Thread de expedição de evento da AWT. Ou seja: ele retornará o programa (“Thread.sleep(1000)” na linha 34) e executará o método “run()” na Thread de expedição de eventos, evitando dessa forma o “travamento” da interface gráfica (a não atualização do “paintComponent”).

**2)**

A chamada do método “pBar.setValue(cont);” (linha 22) **não** necessita do “SwingUtilities.invokeLater” por conta de estar realizando uma operação **rápida** e **simples**, com atribuição de um **valor constante** (linha 21: cont = 0) ao JProgressBar. Por conta disso, não há a preocupação de tratar um possível travamento da interface gráfica.

**3)**

Sim, poderíamos executar “pBar.setValue(cont++);” (linha 31) sem usar o “SwingUtilities.invokeLater” **neste caso**, pois a operação de incrementação do “cont” é sempre realizada em uma Thread **diferente** (criada e iniciada quando pressionado o botão). Além disso, a operação sendo **simples** (um contador que incrementa de 0 até 10) não impossibilita a atualização da interface gráfica (paint()) pelo tempo de execução dessa incrementação.

**4)**

Sim, é possível substituir o método “SwingUtilities.invokeLater” pelo método “SwingUtilities.invokeAndWait”, ficando dessa forma:

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.lang.reflect.InvocationTargetException;

public class TestaThread extends JFrame implements Runnable, ActionListener {

JProgressBar pBar = new JProgressBar(JProgressBar.HORIZONTAL, 0, 10);

JButton b = new JButton("Inicia");

int cont = 0;

TestaThread() {

super("Thread com Swing");

setLayout(new FlowLayout());

add(b);

add(pBar);

b.addActionListener(this);

pack();

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

cont = 0;

pBar.setValue(cont);

new Thread(this).start();

}

public void run() {

try {

while (cont <= 10) {

try {

SwingUtilities.invokeAndWait(new Runnable() {

public void run() {

pBar.setValue(cont++);

}

});

} catch (InvocationTargetException e) {

e.printStackTrace();

}

Thread.sleep(1000);

}

} catch (InterruptedException e) {}

}

static public void main(String[] args) {

new TestaThread();

}

}

A principal diferença entre os dois métodos consiste em: enquanto o “SwingUtilities.invokeLater” permite que a função “run()” de dentro dele seja executada **assíncronamente** na Thread de expedição de evento da AWT, o “SwingUtilities.invokeAndWait” faz essa permissão de maneira **síncrona**. Ou seja: o “SwingUtilities.invokeAndWait” **somente** retornará para a Thread anterior após a execução do método “run()” contido nele, enquanto que o “SwingUtilities.invokeLater” retorna **imediatamente** e executa o método “run()” contido nele de maneira separada da Thread.

**5)**

Após a compilação na linha de comando, são gerados os arquivos: “TestaThread.class” (Classe principal) e “TestaThread$1.class” (Runnable criado internamente no “run()” do “SwingUtilities.invokeLater”).

