**Aluno:** Felipe Alves Belisário

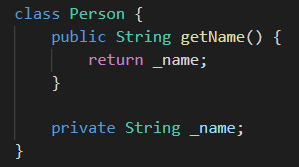
**Matrícula:** 11721BCC030

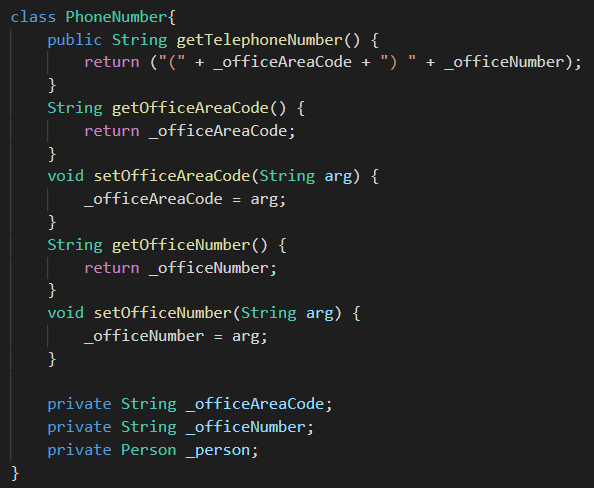
**1 –** A diferença é que na refatoração são feitas mudanças no código que não alteram o resultado final da execução do programa em relação à execução antes das mudanças, ou seja, somente se muda a forma como o código é estruturado. Já na correção de erro, dependendo do tipo de erro presente no código, o programa pode até mesmo nem conseguir executar, então aqui sim devem ser feitas alterações na implementação do mesmo para que o erro seja corrigido e o resultado seja diferente do que era antes da correção.

**2 –**

1. O *Extract Class* pode ser aplicado com a criação de uma outra classe, denominada *PhoneNumber*, sobre a qual possuirá os métodos e variáveis envolvidos com um número telefônico vindos da classe *Person*, além de também possuir a declaração de um objeto do tipo *Person* para poder saber de quem é o número telefônico.

**b)**

****

****

**3 –**

1. O que se pode fazer para que se tenha sincronização entre as duas threads é adicionar a palavra-chave *synchronized* na declaração do método “chuta” na classe “Recurso”, que fará com que enquanto uma thread esteja executando o método as demais fiquem bloequadas até essa execução se finalizar.
2. Eles poderiam ser utilizados de modo que o *wait* deixaria uma thread em estado de espera até que a outra thread que está em execução faça um *notify* para tirar a primeira do estado de espera, e logo após isso essa ultima thread também faria um *wait* para entrar em modo de espera até a outra mandar o *notify*.

**4 –**

import javax.swing.\*;  
import java.io.IOException;  
import java.io.PrintStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Iterator;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class Server extends JFrame{  
  
 public JTextArea outputArea;  
 public JButton jButton1;  
 public ArrayList<String> sum\_list;  
 public float sum = 0;  
 public int qnt = 0;  
  
 public Server(){  
 super( "Server" ); // set title of window  
  
  
 JPanel server\_panel = (JPanel) this.getContentPane();  
 server\_panel.setLayout(null);  
  
 outputArea = new JTextArea(); // create JTextArea for output  
  
 server\_panel.add(outputArea);  
 outputArea.setBounds(10,10,365,340);  
  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setSize( 400, 400 ); // set size of window  
 setVisible( true ); // show window  
  
 try {  
 ServerSocket server = new ServerSocket(12345);  
 outputArea.setText("Servidor iniciado na porta 12345\n\n");  
 outputArea.append( "Server awaiting connections\n" );  
  
 Socket cliente = server.accept();  
 outputArea.append("Cliente conectado do IP "+cliente.getInetAddress().  
 getHostAddress() + "\n\n");  
  
 Scanner entrada = new Scanner(cliente.getInputStream());  
 PrintStream saida = new PrintStream(cliente.getOutputStream());  
  
 while(entrada.hasNextLine()){  
 sum\_list = new ArrayList<String>(Arrays.*asList*(entrada.nextLine().split(" ")));  
 sum = 0;  
 qnt = 0;  
  
  
 Iterator<String> iterator = sum\_list.iterator();  
 String aux;  
 while (iterator.hasNext()) {  
 if(qnt == 0) saida.append("( ");  
 aux = iterator.next();  
  
 saida.append(aux);  
 if(iterator.hasNext()){  
 saida.append(" + ");  
 }  
 qnt++;  
 sum += Integer.*parseInt*(aux);  
 }  
 sum = sum / qnt;  
 saida.append(" ) / " + qnt + " = " + sum + "\n");  
  
 outputArea.append("Média efetuada\n");  
 }  
  
 entrada.close();  
 server.close();  
  
 } catch (IOException ex) {  
 Logger.*getLogger*(Server.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
  
 }  
  
 public static void main(String args[]){  
  
 Server server = new Server();  
  
 }  
  
}

import javax.naming.LimitExceededException;  
import javax.swing.\*;  
import java.io.IOException;  
import java.io.PrintStream;  
import java.net.ConnectException;  
import java.net.Socket;  
import java.rmi.ConnectIOException;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
*/\*\*  
 \*  
 \** ***@author*** *ronaldo  
 \*/*public class FClient extends javax.swing.JFrame {  
  
 private Socket cliente;  
 private javax.swing.JButton jButton1;  
 private javax.swing.JScrollPane jScrollPane1;  
 private javax.swing.JTextArea jTextArea1;  
 private boolean possuiLetras = false;  
  
 */\*\*  
 \* Creates new form FClient  
 \*/* public FClient() {  
 super( "Client" ); // set title of window  
  
 initComponents();  
 initCliente();  
 }  
  
 private void initCliente(){  
 try {  
 cliente = new Socket("127.0.0.1",12345);  
  
 JOptionPane.*showMessageDialog*(new JFrame(), "Connected to the server", "INFO",  
 JOptionPane.*INFORMATION\_MESSAGE*);  
 } catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(new JFrame(), "Connection Refused", "ERROR",  
 JOptionPane.*ERROR\_MESSAGE*);  
 System.*exit*(0);  
 }  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 // <editor-fold defaultstate="collapsed" desc="Generated Code">   
 private void initComponents() {  
  
  
  
 jScrollPane1 = new javax.swing.JScrollPane();  
 jTextArea1 = new javax.swing.JTextArea();  
 jButton1 = new javax.swing.JButton();  
  
 JPanel controlPanel = new JPanel();  
 controlPanel.add(jButton1);  
 controlPanel.add(jTextArea1);  
  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 jTextArea1.setColumns(20);  
 jTextArea1.setRows(5);  
 jScrollPane1.setViewportView(jTextArea1);  
  
 jTextArea1.setText("(Apague esse comentário e digite um numero\napos o outro separados com um espaço)");  
  
 jButton1.setText("Efetuar media");  
 jButton1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton1ActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 getContentPane().setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jScrollPane1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 376, Short.*MAX\_VALUE*)  
 .addGroup(layout.createSequentialGroup()  
 .addComponent(jButton1)  
 .addGap(0, 0, Short.*MAX\_VALUE*)))  
 .addContainerGap())  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 228, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addComponent(jButton1)  
 .addContainerGap(25, Short.*MAX\_VALUE*))  
 );  
  
 pack();  
 }// </editor-fold>   
  
 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
 try {  
  
 String aux = jTextArea1.getText();  
 for(int i=0; i < aux.length(); i++){  
 if( aux.charAt(i) >= 'a' && aux.charAt(i) <= 'z'){  
 possuiLetras = true;  
 }  
 }  
 if(possuiLetras == false) {  
 PrintStream saida = new PrintStream(cliente.getOutputStream());  
 saida.println(jTextArea1.getText());  
  
 Scanner entrada = new Scanner(cliente.getInputStream());  
  
 jTextArea1.setText(entrada.nextLine());  
  
  
 }  
 else{  
 jTextArea1.setText("");  
  
 JOptionPane.*showMessageDialog*(new JFrame(), "Favor inserir apenas números!", "INFO",  
 JOptionPane.*INFORMATION\_MESSAGE*);  
  
 possuiLetras = false;  
 }  
 } catch (IOException ex) {  
 Logger.*getLogger*(FClient.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 }  
  
 */\*\*  
 \** ***@param*** *args the command line arguments  
 \*/* public static void main(String args[]) {  
 /\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
 \*/  
 try {  
 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
 if ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
 break;  
 }  
 }  
 } catch (ClassNotFoundException ex) {  
 java.util.logging.Logger.*getLogger*(FClient.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (InstantiationException ex) {  
 java.util.logging.Logger.*getLogger*(FClient.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (IllegalAccessException ex) {  
 java.util.logging.Logger.*getLogger*(FClient.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(FClient.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 //</editor-fold>  
  
 /\* Create and display the form \*/  
 java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new FClient().setVisible(true);  
 }  
 });  
 }  
  
}

**5 –**

1. Os métodos utilizados dependem se o comando terá retorno de dados ou não, por exemplo, o comando *select* que possui uma tabela como retorno é necessário a utilização do método *executeQuery* enquanto um *update*, *create table*, *alter table* e dentre outros que não possuem um retorno deve-se utilizar os método *executeUpdate* e *execute*.
2. O que deve ser alterado é algo bem simples, deve-se alterar somente o driver que será utilizado tanto na url do mesmo quanto na url de conexão, além de também ter que alterar o nome do novo SGBD a ser utilizado nessa última url.
3. É necessário a classe *Connection* para fazer a conexão com o banco de dados e um driver para facilitar o acesso e interação com esse banco através de linhas de código da linguagem escolhida.