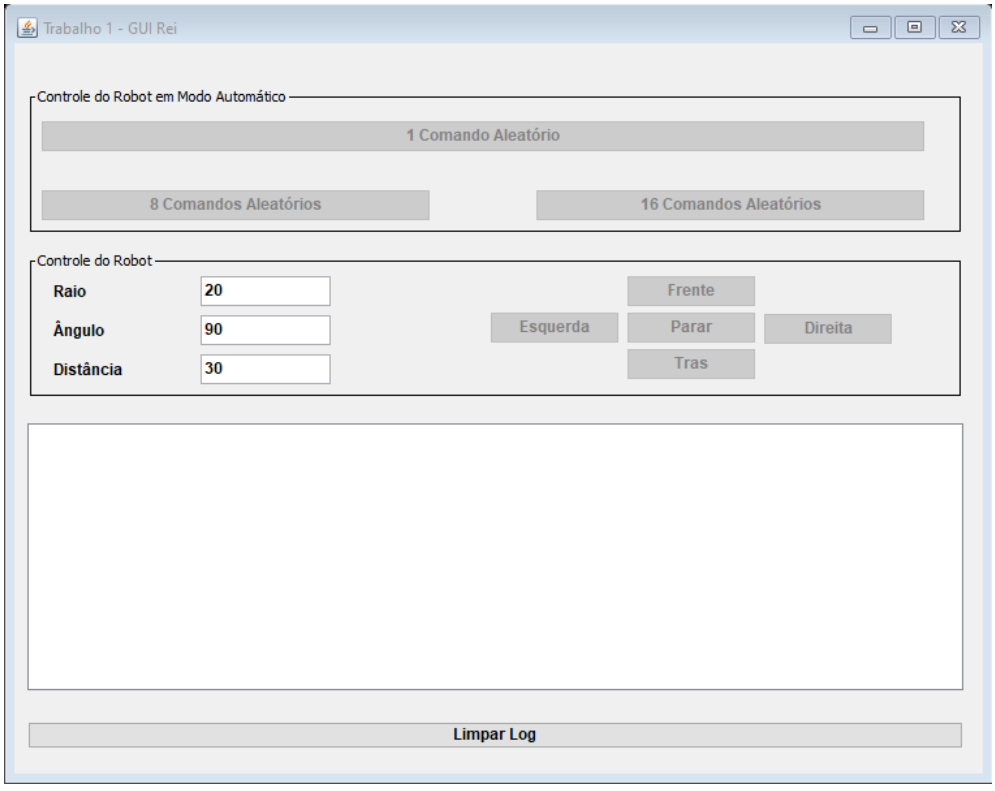
Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

Fundamentos de Sistemas Operativos - 2324SI

2º Trabalho Prático - Aula prática 2



Docente Carlos Carvalho

Realizado por (Grupo 7):
Diogo Santos 48626
Pedro Silva 48965
João Fonseca 49707

26 de setembro de 2024

Conteúdo

1 Introdução

Esta aula consistiu em desenhar o diagrama de atividades, implementar e testar a tarefa Rei. O Rei, como no jogo "O Rei Manda"vai enviar instruções, através de um buffer circular já desenvolvido pelos docentes, para o Súbdito em que este vai ter que as realizar. A tarefa Rei está dividido em 3 classes: a BD_Rei, a GUI_Rei e a App_Rei. A classe BD_Rei vai se manter igual à do trabalho passado, a GUI_Rei foi desenvolvida na aula anterior e a App_Rei vai ter que acomodar o tal buffer circular em vez do canal de comunicação.

2 Desenvolvimento

2.1 Diagrama de Atividades

Começamos pelo desenvolvimento do diagrama de atividades. Este permite-nos estruturar o resto da aula pois conseguimos estabelecer claramente quais são os objetivos e a forma como opera a tarefa Rei .

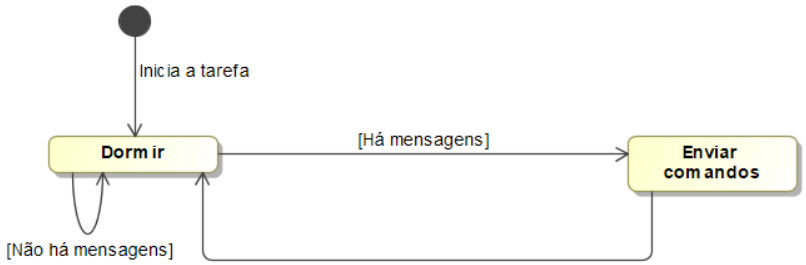


Figura 1: Diagrama de atividades

Como podemos observar na imagem acima, a tarefa Rei inicia no estado dormir, mantendo-se no mesmo até existirem mensagens para enviar, quando esse é o enviamos as mensagens através do buffer circular até este ficar cheio ou até não termos mais mensagens para enviar.

2.2 Buffer Circular

Esta classe, disponibilizada pelos docentes onde tivemos que modificar para aceitar Mensagens e não Strings, permite-nos a comunicação entre as nossas duas tarefas Rei e Súbdito. Através da utilização de semáforos conseguimos manter a integridade e não perda de informação na manipulação do Buffer. Estes semáforos funcionam de modo a que cada vez que vamos mexer em informação do Buffer, quer seja adicionar ou remover, impossibilitamos que qualquer outra tarefa possa aceder a esta informação até que nós tenhamos acabado.

```
public void inserirElemento(Mensagem msg){
    try {
        elementosLivres.acquire();
        acessoElemento.acquire();
        bufferCircular[putBuffer]= msg;
        System.out.print(msg);
        putBuffer= ++putBuffer % dimensaoBuffer;
        acessoElemento.release();
    } catch (InterruptedException e) {}
        elementosOcupados.release();
    }

public Mensagem removerElemento() {
     Mensagem msg= null;
     try {
     elementosOcupados.acquire();
     acessoElemento.acquire();
     } catch (InterruptedException e) {}

     msg = bufferCircular[getBuffer];
     getBuffer= ++getBuffer % dimensaoBuffer;
     acessoElemento.release();
     elementosLivres.release();
     return msg;
    }
```

Como podemos observar nas duas funções acima, fazemos um "aquire"à variável acessoElemento nos seus inícios o que vai indicar a todas as tarefas que queiram usar o buffer, este está a ser utilizado. Os semáforos funcionam de tal forma que apenas permitem o acesso a determinada variável, função etc. a uma tarefa bloqueando o acesso a todas as outras se esta estiver a ser utilizada e é libertada quando acabar.

2.3 Tarefa Rei

Comparando com o trabalho passado começámos por alterar o Rei para passar de um processo para uma tarefa ao retirarmos o método "main"e a estendermos a classe Java "Thread". Ao estendermos esta classe e com o método "run"implementado bastou-nos criar uma instância desta tarefa e começá-la quando o utilizador ativá-la através da GUI principal.

Para substituirmos o canal de comunicação e utilizar o buffer circular, tivemos de criar um novo método para verificar que o acesso à mensagem desejada é único. Este método é o "setMensagem":

```
public void setMensagem(Mensagem m)
{
    try {
    livreMyMensagem.acquire();
    acessoMyMensagem.acquire();
    } catch (InterruptedException e) {}
    myMensagem= m;
    acessoMyMensagem.release();
    ocupadaMyMensagem.release();
}
```

Com este método vai ser possível pegar na mensagem recebida e atribuí-la à mensagem que queremos enviar sem qualquer impedimento.

Para adicionarmos a mensagem ao Buffer alterámos o estado escrever mensagem da seguinte forma:

```
case escreverMensagem:
    System.out.println("escreve");
    System.out.println("Mensagens à espera: " + bd.getMensagens().size());
    msg = bd.getMensagens().get(0);
    msg.setId(counter);
    setMensagem(msg);
    try {
        ocupadaMyMensagem.acquire();
        acessoMyMensagem.acquire();
    } catch (InterruptedException e) {}
        bufferCircular.inserirElemento(myMensagem);
        acessoMyMensagem.release();
        livreMyMensagem.release();
        haTrabalho.release();
        gui.txtLog.append(" Enviei = " + msg + "\n");
        counter =++ counter % bd.getNMensagens();
        bd.removeMensagem();
        state = dormir;
        break;
```

Assim, quando entramos neste estado guardamos a mensagem que queremos enviar, criada na GUI e adicionada à lista de mensagens na base de dados, utilizando o método acima. De seguida, fazemos "acquire"aos semáforos ocupado e acesso. Para o semáforo ocupado isto significa que a nossa mensagem já não está ocupada porque este semáforo começa a 0 e é lhe dado um "release"no método acima. No semáforo acesso é o caso oposto pois este começa a 1 e quando sai do método "setMensagem"continua com 1. É inserida a mensagem no buffer e libertos todos os semáforos acionados quer no método acima quer no método "inserirElemento". O resto deste estado mantém-se igual ao primeiro trabalho.

3 Conclusões

Devido a este trabalho depender tanto do último estivemos a colher os frutos do nosso trabalho pois bastou-nos substituir tudo o que estava relacionado com o Canal de Comunicação pelo Buffer Circular, que nos era disponibilizado pelos docentes. Esta aula permitiu-nos familiarizar com a nova matéria, como os semáforos e as threads, o que foi um bom complemento da aula teórica.

4 Bibliografia

1. Folhas de Computação Física - Jorge Pais, 2023/2024

5 Código Java GUI_Rei, BD_Rei, App_Rei, Rei_Subdito

```
1
2  Classe GUI_Rei
3
4  package ptrabalho;
5
6  import java.awt.Color;
7  import java.awt.EventQueue;
8  import java.awt.Font;
9  import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.util.Random;
12
13 import javax.swing.BorderFactory;
14 import javax.swing.JButton;
15 import javax.swing.JFrame;
16 import javax.swing.JPanel;
17 import javax.swing.border.Border;
18 import javax.swing.border.EmptyBorder;
19 import javax.swing.border.LineBorder;
20 import javax.swing.border.TitledBorder;
21
22 public class GUI_Rei extends GUI_BaseRS
23 {
24   private int id = 0;
25   protected JButton btn8com;
26   protected JButton btn16com;
27   protected JButton btn1com;
28   Mensagem msg = null;
29
30
31   public GUI_Rei(BD_Rei bd)
32   {
33     super(bd);
34     EventQueue.invokeLater(new Runnable()
35     {
36       public void run()
37       {
38         try
39         {
40           init_Rei(bd);
41
42         } catch (Exception e)
43         {
44           e.printStackTrace();
45         }
46       }
47     });
48   }
49
50   /**
51    * Create the frame.
52    */
53   public void init_Rei(BD_Rei bd)
54   {
55
56     setTitle("Trabalho 1 - GUI Rei");
57     //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
58     setBounds(1000, 0, 760, 600);
59
```

```
60      Border cost = BorderFactory.createLineBorder(new Color(0,0,0)
        ,1);
61      TitledBorder borda_rei = BorderFactory.createTitledBorder(cost,
        "Controle do Robot em Modo Automático");
62      JPanel panel_1_1 = new JPanel();
63      panel_1_1.setLayout(null);
64      panel_1_1.setName("Controle do Robot em Modo Automático");
65      panel_1_1.setBorder(new LineBorder(new Color(0, 0, 0)));
66      panel_1_1.setBounds(10, 34, 719, 113);
67      panel_1_1.setBorder(borda_rei);
68      getContentPane().add(panel_1_1);
69
70      btn8com = new JButton("8 Comandos Aleatórios");
71      btn8com.setEnabled(false);
72      btn8com.setFont(new Font("Arial", Font.BOLD, 12));
73      btn8com.setBounds(10, 78, 300, 25);
74      panel_1_1.add(btn8com);
75
76      btn16com = new JButton("16 Comandos Aleatórios");
77      btn16com.setEnabled(false);
78      btn16com.setFont(new Font("Arial", Font.BOLD, 12));
79      btn16com.setBounds(390, 78, 300, 25);
80      panel_1_1.add(btn16com);
81
82      btn1com = new JButton("1 Comando Aleatório");
83      btn1com.setFont(new Font("Arial", Font.BOLD, 12));
84      btn1com.setEnabled(false);
85      btn1com.setBounds(10, 25, 680, 25);
86      panel_1_1.add(btn1com);
87      contentPane = new JPanel();
88      contentPane.setBorder(new EmptyBorder(100, 100,100, 100));
89
90      btn1com.addActionListener(new ActionListener()
91      {
92        public void actionPerformed(ActionEvent e)
93        {
94          msg = gerarRandomMensagem();
95          bd.addMensagem(msg);
96          txtLog.append(" Criei a mensagem " + msg + "\n");
97        }
98      });
99
100
101     btn8com.addActionListener(new ActionListener()
102     {
103       public void actionPerformed(ActionEvent e)
104       {
105         for (int i = 0; i<8; i++)
106         {
107           msg = gerarRandomMensagem();
108           bd.addMensagem(msg);
109           txtLog.append(" Criei a mensagem " + msg + "\n");
110
111         }
112       }
113     });
114
115     btn16com.addActionListener(new ActionListener()
116     {
117       public void actionPerformed(ActionEvent e)
```

```
118       {
119         for (int i = 0; i<16; i++)
120         {
121           msg = gerarRandomMensagem();
122           bd.addMensagem(msg);
123           txtLog.append(" Criei a mensagem " + msg + "\n");
124         }
125 
126       }
127     });
128 
129 
130     btnFrt.addActionListener(new ActionListener()
131     {
132       public void actionPerformed(ActionEvent e)
133       {
134         Mensagem mensagem = new Mensagem(id,1,bd.getDist(),0);
135         bd.addMensagem(mensagem);
136         txtLog.append(" Criei a mensagem " + mensagem + "\n");
137       }
138     });
139 
140     btnEsq.addActionListener(new ActionListener()
141     {
142       public void actionPerformed(ActionEvent e)
143       {
144         Mensagem mensagem = new Mensagem(id,3,bd.getRaio(),bd.
    getAng());
145         bd.addMensagem(mensagem);
146         txtLog.append(" Criei a mensagem " + mensagem + "\n");
147       }
148     });
149 
150     btnDir.addActionListener(new ActionListener()
151     {
152       public void actionPerformed(ActionEvent e)
153       {
154         Mensagem mensagem = new Mensagem(id,2,bd.getRaio(),bd.
    getAng());
155         bd.addMensagem(mensagem);
156         txtLog.append(" Criei a mensagem " + mensagem + "\n");
157       }
158     });
159 
160     btnTras.addActionListener(new ActionListener()
161     {
162       public void actionPerformed(ActionEvent e)
163       {
164         Mensagem mensagem = new Mensagem(id,1,-bd.getDist(),0);
165         bd.addMensagem(mensagem);
166         txtLog.append(" Criei a mensagem " + mensagem + "\n");
167       }
168     });
169 
170     btnParar.addActionListener(new ActionListener()
171     {
172       public void actionPerformed(ActionEvent e)
173       {
174         Mensagem mensagem = new Mensagem(id,0,0,0);
175         bd.addMensagem(mensagem);
```

```
176         txtLog.append(" Criei a mensagem " + mensagem + "\n");
177       }
178     });
179 
180     setVisible(true);
181   }
182 
183   private Mensagem gerarRandomMensagem() {
184     Mensagem m = new Mensagem();
185     Random rn = new Random();
186     int[] variaveis = new int[4]; // array de 4 variaveis
187     int tipoMensagem = rn.nextInt(3); // random between 0-2
188     if(id==8)
189       id=0;
190     variaveis = gerarVariaveis(tipoMensagem);
191     m.setId(variaveis[0]);
192     m.setTipo(variaveis[1]);
193     m.setArg1(variaveis[2]);
194     m.setArg2(variaveis[3]);
195     return m;
196   }
197 
198   /*
199    * Metodo auxiliar ao gerarRandomMensagem()
200    *
201    * @param tMsg - tipo de mensagem
202    */
203   private int[] gerarVariaveis(int tMsg) {
204     Random rn = new Random();
205     int[] variaveis = new int[4];
206     int variavel;
207     if (tMsg == 0) { // para tMsg 0 faz reta
208       variaveis[0] = id;
209       variaveis[1] = 1; // 1 equivale a reta na minha mensagem
210 
211       variavel = rn.nextInt(45) + 5; // random between 5-50 cm reta
212       int sinal = rn.nextInt(2);
213       if (sinal == 1)
214         variavel*=-1;
215       variaveis[2] = variavel;
216       variaveis[3] = 0;
217     } else if (tMsg == 1) { // para tMsg 1 faz curva direita
218       variaveis[0] = id;
219       variaveis[1] = 2; // 2 equivale a reta na minha mensagem
220       variavel = rn.nextInt(30); // random between 0-30 raio
221       variaveis[2] = variavel;
222       variavel = rn.nextInt(70) + 20;// random between 20-90 angulo
223       variaveis[3] = variavel;
224     } else { // para tMsg 0 faz curva esquerda
225       variaveis[0] = id;
226       variaveis[1] = 3; // 3 equivale a reta na minha mensagem
227       variavel = rn.nextInt(30); // random between 0-30 raio
228       variaveis[2] = variavel;
229       variavel = rn.nextInt(70) + 20;// random between 20-90 angulo
230       variaveis[3] = variavel;
231     }
232     return variaveis;
233   }
234 
235   protected void start()
```

```
  {
    super.start();
    btn1com.setEnabled(true);
    btn16com.setEnabled(true);
    btn8com.setEnabled(true);
  }

  protected void off()
  {
    super.off();
    btn1com.setEnabled(false);
    btn16com.setEnabled(false);
    btn8com.setEnabled(false);
  }
}
```

Classe BD_Rei

```
package ptrabalho;


//import robot.RobotLegoEV3;

public class BD_Rei extends BD_Base
{
    //private RobotLegoEV3 robot;
    private boolean terminar;
    private boolean ligado;

    private String nome;


    public BD_Rei()
    {
      super();
        terminar = false;
        ligado = false;

    }


    public boolean getTerminar()
    {
        return terminar;
    }

    public void setTerminar(boolean b)
    {
        terminar = b;
    }

    public boolean isLigado()
    {
        return ligado;
    }

    public void setLigado(boolean b)
    {
        ligado = b;
    }
```

```
296
297     public void setNome(String n)
298     {
299         nome = n;
300     }
301
302     public String getNome()
303     {
304         return nome;
305     }
306
307
308
309 }
310
311 Classe App_Rei
312
313 package ptrabalho;
314
315 import java.util.concurrent.Semaphore;
316
317 public class App_Rei extends Thread
318 {
319     @SuppressWarnings("unused")
320     protected GUI_Rei gui;
321     private BD_Rei bd;
322     Mensagem msg;
323     Mensagem myMensagem = null;
324     private int state = 2;
325     private int counter = 0;
326     private final int escreverMensagem = 1;
327     private final int dormir = 2;
328     private final int esperarTempoExecucao = 3;
329
330     BufferCircular bufferCircular;
331     Semaphore haTrabalho, livreMyMensagem, ocupadaMyMensagem,
    acessoMyMensagem;
332
333
334     public App_Rei(BD_Rei bdRei, BufferCircular bc, Semaphore ht)
335     {
336       bd = bdRei;
337         gui = new GUI_Rei(bd);
338         bufferCircular= bc;
339       haTrabalho= ht;
340       myMensagem = null;
341       livreMyMensagem= new Semaphore(1);
342       ocupadaMyMensagem= new Semaphore(0);
343       acessoMyMensagem= new Semaphore(1);
344     }
345
346     public BD_Rei getBD()
347     {
348         return bd;
349     }
350
351     public void setMensagem(Mensagem m)
352     {
353       try {
354       livreMyMensagem.acquire();
```

```
355       acessoMyMensagem.acquire();
356       } catch (InterruptedException e) {}
357       myMensagem= m;
358       acessoMyMensagem.release();
359       ocupadaMyMensagem.release();
360     }
361
362
363     public void run()
364     {
365         while(true) {
366
367           switch (state) {
368
369             case escreverMensagem:
370               System.out.println("escreve");
371               System.out.println("Mensagens à espera: " + bd.
    getMensagens().size());
372               msg = bd.getMensagens().get(0);
373               msg.setId(counter);
374               setMensagem(msg);
375               try {
376                 ocupadaMyMensagem.acquire();
377                 acessoMyMensagem.acquire();
378               } catch (InterruptedException e) {}
379                 bufferCircular.inserirElemento(myMensagem);
380                 acessoMyMensagem.release();
381                 livreMyMensagem.release();
382                 haTrabalho.release();
383                 gui.txtLog.append(" Enviei = " + msg + "\n");
384                 counter =++ counter % bd.getNMensagens();
385                 bd.removeMensagem();
386                 state = dormir;
387                 break;
388
389             case dormir:
390               //System.out.println("sleep");
391               try {
392                 Thread.sleep(1000);
393               } catch (InterruptedException e) {
394                 // TODO Auto-generated catch block
395                 e.printStackTrace();
396               }
397               System.out.println(bd.getMensagens().size());
398               if(bd.getMensagens().size() !=0) {
399                 state = escreverMensagem;
400                 break;
401               }
402               else {break;}
403
404       }
405     }
406     /*System.out.println("sai");
407         try {
408       Thread.sleep(100);
409     } catch (InterruptedException e) {
410       // TODO Auto-generated catch block
411       e.printStackTrace();
412     }*/
413
```

```
414     }
415
416     /*public static void main(String[] args) throws
        InterruptedException
417     {
418
419         App_Rei app = new App_Rei();
420         System.out.println("A aplicação começou.");
421         app.run();
422         System.out.println("A aplicação terminou.");
423     }*/
424
425 }
426
427 Classe Rei_Subdito
428
429 package ptrabalho;
430
431 import java.awt.EventQueue;
432
433 import javax.swing.JFrame;
434 import javax.swing.JPanel;
435 import javax.swing.border.EmptyBorder;
436 import javax.swing.JCheckBox;
437 import java.awt.Font;
438 import java.awt.event.ActionEvent;
439 import java.awt.event.ActionListener;
440 import java.util.concurrent.Semaphore;
441
442 import javax.swing.SwingConstants;
443
444 public class Rei_Subdito extends GUI_Base {
445
446     private JPanel contentPane;
447     private App_Rei appRei;
448     private App_Subdito appSub;
449     private GUI_Subdito gui_Subdito;
450     private Gravar gui_Gravar;
451     private BD_Rei bdRei = new BD_Rei();
452     private BD_Subdito bdSub = new BD_Subdito();
453
454
455
456     public Rei_Subdito(BD_Base bd, BufferCircular bc, Semaphore ht)
457   {
458     super(bd);
459     EventQueue.invokeLater(new Runnable()
460     {
461       public void run()
462       {
463         try
464         {
465           init_Rei_Subdito(bc, ht);
466
467         } catch (Exception e)
468         {
469           e.printStackTrace();
470         }
471       }
472     });
```

```
473   }
474
475
476   public void init_Rei_Subdito(BufferCircular bc, Semaphore ht) {
477
478     setTitle("Trabalho 2 - Rei_Subdito");
479     //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
480     setBounds(0, 0, 754, 600);
481
482       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
483       getContentPane().setLayout(null);
484       JCheckBox chckbxRei = new JCheckBox("REI");
485       chckbxRei.setHorizontalAlignment(SwingConstants.LEFT);
486       chckbxRei.setFont(new Font("Tahoma", Font.PLAIN, 15));
487       chckbxRei.setBounds(16, 56, 97, 59);
488       getContentPane().add(chckbxRei);
489       JCheckBox chckbxSubdito = new JCheckBox("SUBDITO");
490       chckbxSubdito.setHorizontalAlignment(SwingConstants.LEFT);
491       chckbxSubdito.setFont(new Font("Tahoma", Font.PLAIN, 15));
492       chckbxSubdito.setBounds(16, 100, 195, 59);
493       getContentPane().add(chckbxSubdito);
494       JCheckBox chckbxGravar = new JCheckBox("GRAVAR");
495       chckbxGravar.setHorizontalAlignment(SwingConstants.LEFT);
496       chckbxGravar.setFont(new Font("Tahoma", Font.PLAIN, 15));
497       chckbxGravar.setBounds(16, 142, 108, 59);
498       getContentPane().add(chckbxGravar);
499       appRei = new App_Rei(bdRei, bc, ht);
500       Thread tRei = new Thread(appRei);
501       tRei.start();
502
503       //appRei.run();
504       //gui_Subdito = new GUI_Subdito(bdSub);
505       appSub = new App_Subdito(bdSub, bc, ht);
506       Thread tSub = new Thread(appSub);
507       tSub.start();
508       gui_Gravar = new Gravar(bdRei);
509
510       chckbxRei.addActionListener(new ActionListener()
511     {
512       public void actionPerformed(ActionEvent e)
513       {
514         if(chckbxRei.isSelected()) {
515           if (!appRei.gui.isVisible())
516             appRei.gui = new GUI_Rei(bdRei);
517           appRei.gui.start();
518           System.out.println(bdRei.getMensagens().size());
519           //appRei.run();
520           //t.run();
521           txtLog.append("Ativei a GUI_REI \n");
522         }
523         else {
524           //gui_Rei.off();
525           txtLog.append("Desativei a GUI_REI \n");
526         }
527
528       }
529     });
530
531       chckbxSubdito.addActionListener(new ActionListener()
532     {
```

```
533         public void actionPerformed(ActionEvent e)
534         {
535           if(chckbxSubdito.isSelected()) {
536             if (!appSub.gui.isVisible())
537               appSub.gui = new GUI_Subdito(bdSub);
538             appSub.gui.start();
539             System.out.println(bdRei.getMensagens().size());
540             txtLog.append("Ativei a GUI_SUBDITO \n");
541           }
542           else {
543             //gui_Subdito.off();
544             txtLog.append("Desativei a GUI_SUBDITO \n");
545           }
546 
547 
548         }
549       });
550 
551         chckbxGravar.addActionListener(new ActionListener()
552       {
553         public void actionPerformed(ActionEvent e)
554         {
555           if(chckbxGravar.isSelected()) {
556             if (!gui_Subdito.isVisible())
557               gui_Gravar = new Gravar(bdRei);
558             gui_Gravar.start();
559             txtLog.append("Ativei a GUI_GRAVAR \n");
560           }
561           else {
562             gui_Gravar.off();
563             txtLog.append("Desativei a GUI_GRAVAR \n");
564           }
565 
566 
567         }
568       });
569 
570         setVisible(true);
571     }
572 
573 
574 }
```