



Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

Fundamentos de Sistemas Operativos - 2324SI

2º Trabalho Prático - Aula prática 3

The screenshot shows a Windows-style application window titled "Trabalho 2 - Subdito". The interface is designed for controlling a robot. At the top, there is a text input field labeled "Nome do robot" and a radio button labeled "Abrir / Fechar Bluetooth". Below this, a section titled "Controle do Robot" contains three input fields for "Raio" (set to 20), "Ângulo" (set to 90), and "Distância" (set to 30). To the right of these fields are five directional buttons: "Frente", "Esquerda", "Parar", "Direita", and "Tras". At the bottom of the window is a large empty rectangular area, likely for a video feed or log, and a button labeled "Limpar Log".

Docente Carlos Carvalho

Realizado por (Grupo 7):

Diogo Santos 48626

Pedro Silva 48965

João Fonseca 49707

26 de setembro de 2024

Conteúdo

1	Introdução	I
2	Desenvolvimento	I
2.1	Diagrama de Atividades	I
2.2	Tarefa Súbdito	II
3	Conclusões	III
4	Bibliografia	III
5	Código Java GUI_Subdito, BD_Subdito, App_Subdito	IV

1 Introdução

Esta aula consistiu em desenhar o diagrama de atividades, implementar e testar a tarefa Súbdito. O Súbdito, como no jogo "O Rei Manda" vai receber instruções, enviadas pelo Rei, através do buffer circular já desenvolvido pelos docentes, e vai realizá-las. A tarefa Súbdito está dividida em 3 classes: a BD_Subdito, a GUI_Subdito e a App_Subdito. 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 Súbdito.



Figura 1: Diagrama de atividades

Como podemos observar na imagem acima, a tarefa Súbdito inicia no estado dormir, mantendo-se no mesmo até receber uma mensagem, quando esse é caso guardamos a mensagem numa variável e verificamos se o robot está ligado, se sim enviamos o comando para o robot, se não voltamos ao estado dormir e aguardamos mais mensagens repetindo assim o loop.

2.2 Tarefa Súbdito

Vamos ter um trabalho igual ao da aula anterior para passar de um processo para uma tarefa. Para isso retiramos o método "main" e a estendemos 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 substituírmos 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 "getMensagem":

```
public Mensagem getMensagem() {
    try { ocupadaMyMensagem.acquire();
        acessoMyMensagem.acquire();
    } catch (InterruptedException e) {}
    Mensagem s = myMensagem;
    acessoMyMensagem.release();
    livreMyMensagem.release();
    return s;
}
```

Com este método vai ser possível pegar na mensagem que foi retirada do buffer e atribuída à variável myMensagem onde de seguida vai ser retornada.

Para retirarmos a mensagem do Buffer alterámos o estado dormir/busca da seguinte forma:

```
case dormir:
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    System.out.println("busca" + bufferCircular.available());
    if (bufferCircular.available() != 0) {
        try { haTrabalho.acquire();
            livreMyMensagem.acquire();
        } catch (InterruptedException e) {}
        Mensagem m = bufferCircular.removeElemento();
        try { acessoMyMensagem.acquire(); }
        catch (InterruptedException e) {}
        myMensagem = m;
        acessoMyMensagem.release();
        ocupadaMyMensagem.release();
        msg = getMensagem();
        System.out.println(bd.getMensagens());
        state = receberMensagem;
        break;}
    else
        if (bd.isLigado())
            state = esperarTempoExecucao;
        break;
```

Assim, quando entramos neste estado sinalizamos os semáforos haTrabalho e livreMyMensagem, esta última que tinha levado um release no método acima. De seguida, removemos a mensagem do buffer e guardamos-a numa variável. Temos agora que voltar a colocar os semáforos nas suas posições iniciais, ou seja, preparados para uma nova mensagem. Entramos então no método getMensagem que nos vai devolver a mensagem que foi retirada do buffer e vai levantar todos os semáforos que controlavam o acesso a esta mensagem. Adicionámos também, por sugestão do docente, um mecanismo para fazer com que a tarefa possa receber mensagens e guardá-las mesmo que o robot não esteja ligado. Isto foi conseguido ao criarmos um novo método no BufferCircular que nos vai indicar, através do semáforo elementosOcupados, se existe ou não elementos no buffer. Se sim vamos obtê-los se não verificamos se existem mensagens guardadas na lista do Súbdito. Os restantes estados mantêm-se iguais ao trabalho anterior.

3 Conclusões

Esta aula, muito como a anterior, resumiu-se a substituir tudo o que estava relacionado com o Canal de Comunicação pelo Buffer Circular, que nos era disponibilizado pelos docentes. Como a comunicação com o robot estava a funcionar no último trabalho não tivemos que alterar nada nesse aspeto. Também adicionamos o mecanismo mencionado anteriormente o que tornou esta tarefa mais robusta e menos propensa a erros.

4 Bibliografia

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

5 Código Java GUI_Subdito, BD_Subdito, App_Subdito

```
1
2 Classe GUI_Subdito
3
4 package ptrabalho;
5
6 import java.awt.EventQueue;
7
8 import javax.swing.BorderFactory;
9 import javax.swing.JFileChooser;
10 import javax.swing.JFrame;
11 import javax.swing.JPanel;
12 import javax.swing.border.Border;
13 import javax.swing.border.EmptyBorder;
14 import javax.swing.border.LineBorder;
15 import javax.swing.border.TitledBorder;
16
17 import java.awt.Color;
18 import javax.swing.JLabel;
19 import java.awt.Font;
20 import java.awt.event.ActionEvent;
21 import java.awt.event.ActionListener;
22 import java.awt.event.WindowAdapter;
23 import java.awt.event.WindowEvent;
24
25 import javax.swing.SwingConstants;
26 import javax.swing.JTextField;
27 import javax.swing.JRadioButton;
28 import javax.swing.JTextArea;
29
30 public class GUI_Subdito extends GUI_BaseRS
31 {
32
33     private JPanel contentPane;
34     private JTextField txtNome;
35
36
37     /**
38      * Launch the application.
39      */
40     public GUI_Subdito(BD_Subdito bd)
41     {
42         super(bd);
43         EventQueue.invokeLater(new Runnable()
44         {
45             public void run()
46             {
47                 try
48                 {
49                     init_Subdito(bd);
50
51                 } catch (Exception e)
52                 {
53                     e.printStackTrace();
54                 }
55             }
56         });
57     }
58
59     /**
```

```

60  * Create the frame.
61  */
62  public void init_Subdito(BD_Subdito bd)
63  {
64      setTitle("Trabalho 2 - Subdito");
65
66      //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
67      setBounds(50, 400, 760, 600);
68
69
70      JLabel lblNomeDoRobot = new JLabel("Nome do robot");
71      lblNomeDoRobot.setHorizontalAlignment(SwingConstants.LEFT);
72      lblNomeDoRobot.setFont(new Font("Arial", Font.BOLD, 12));
73      lblNomeDoRobot.setBounds(30, 61, 103, 25);
74      getContentPane().add(lblNomeDoRobot);
75
76      JRadioButton rdbtnAbrirFecharBlt = new JRadioButton("Abrir /
Fechar Bluetooth");
77      /*rdbtnAbrirFecharBlt.addActionListener(new ActionListener()
78      {
79          public void actionPerformed(ActionEvent e) {
80              if (bd.isLigado())
81              {
82                  System.out.println("Desligando...");
83                  bd.getRobot().CloseEV3();
84                  bd.setLigado(false);
85                  if(rdbtnAtivarDesativarComp.isSelected() ) {
86                      btnFrt.setEnabled(false);
87                      btnEsq.setEnabled(false);
88                      btnParar.setEnabled(false);
89                      btnDir.setEnabled(false);
90                      btnTras.setEnabled(false);
91                  }
92
93              } else
94              {
95                  System.out.println("Ligando...");
96                  bd.setLigado(bd.getRobot().OpenEV3(bd.getNome()));
97                  System.out.println(bd.isLigado());
98                  if (!bd.isLigado())
99                      rdbtnAbrirFecharBlt.setSelected(false);
100                  if(rdbtnAtivarDesativarComp.isSelected() ) {
101                      btnFrt.setEnabled(true);
102                      btnEsq.setEnabled(true);
103                      btnParar.setEnabled(true);
104                      btnDir.setEnabled(true);
105                      btnTras.setEnabled(true);}
106
107                      //bd.setLigado(true);
108              }
109          }
110      });*/
111
112
113      txtNome = new JTextField();
114      /*txtNome.addActionListener(new ActionListener()
115      {
116          public void actionPerformed(ActionEvent e)
117          {
118              bd.setNome(txtNome.getText());

```

```

119         txtLog.append(" O nome do Robot é " + bd.getNome() + "\n");
120     }
121 });*/
122
123
124     txtNome.setFont(new Font("Arial", Font.BOLD, 12));
125     txtNome.setColumns(10);
126     txtNome.setBounds(143, 61, 279, 25);
127     getContentPane().add(txtNome);
128
129
130
131     rdbtnAbrirFecharBlt.setFont(new Font("Arial", Font.BOLD, 12));
132     rdbtnAbrirFecharBlt.setBounds(455, 61, 188, 25);
133     getContentPane().add(rdbtnAbrirFecharBlt);
134
135     setVisible(true);
136 }
137
138 }
139
140
141 Classe BD_Subdito
142
143 package ptrabalho;
144 import robot.RobotLegoEV3;
145
146 public class BD_Subdito extends BD_Base
147 {
148     private RobotLegoEV3 robot;
149     private boolean terminar;
150     private boolean ligado;
151
152     private String nome;
153
154     public BD_Subdito()
155     {
156         super();
157         robot = new RobotLegoEV3();
158         terminar = false;
159         ligado = false;
160
161     }
162
163     public RobotLegoEV3 getRobot()
164     {
165         return robot;
166     }
167
168     public boolean getTerminar()
169     {
170         return terminar;
171     }
172
173     public void setTerminar(boolean b)
174     {
175         terminar = b;
176     }
177
178     public boolean isLigado()

```



```

179     {
180         return ligado;
181     }
182
183     public void setLigado(boolean b)
184     {
185         ligado = b;
186     }
187
188     public void setNome(String n)
189     {
190         nome = n;
191     }
192
193     public String getNome()
194     {
195         return nome;
196     }
197 }
198
199
200 Classe App_Subdito
201
202 package ptrabalho;
203 import java.lang.Math;
204 import java.util.concurrent.Semaphore;
205
206 public class App_Subdito extends Thread
207 {
208     @SuppressWarnings("unused")
209     protected GUI_Subdito gui;
210     private BD_Subdito bd;
211     Mensagem msg = null;
212     Mensagem myMensagem = null;
213     private int state = 2;
214     private int counter = 0;
215     private final int receberMensagem = 1;
216     private final int dormir = 2;
217     private final int esperarTempoExecucao = 3;
218     private final int RETA = 1;
219     private final int CURVARDIR = 2;
220     private final int CURVARESQ = 3;
221
222     BufferCircular bufferCircular;
223     Semaphore haTrabalho, livreMyMensagem, ocupadaMyMensagem,
    acessoMyMensagem;
224
225     public App_Subdito(BD_Subdito bdSub, BufferCircular bc,
    Semaphore ht)
226     {
227         bd = bdSub;
228
229         gui = new GUI_Subdito(bd);
230
231         bufferCircular= bc;
232         haTrabalho= ht;
233         myMensagem= null;
234         livreMyMensagem= new Semaphore(1);
235         ocupadaMyMensagem= new Semaphore(0);
236         acessoMyMensagem= new Semaphore(1);

```

```

237     }
238
239     public BD_Subdito getBD()
240     {
241         return bd;
242     }
243
244     public Mensagem getMensagem() {
245         try { ocupadaMyMensagem.acquire();
246             acessoMyMensagem.acquire();
247         } catch (InterruptedException e) {}
248         Mensagem s= myMensagem;
249         acessoMyMensagem.release();
250         livreMyMensagem.release();
251         return s;
252     }
253
254
255     public void run()
256     {
257         while(!bd.getTerminar()) {
258
259             switch (state) {
260
261                 case receberMensagem:
262                     System.out.println("recebe");
263                     gui.txtLog.append(" Recebi = " + msg + "\n");
264                     bd.addMensagem(msg);
265                     state = dormir;
266                     break;
267
268                 case dormir:
269                     try {
270                         Thread.sleep(1000);
271                     } catch (InterruptedException e) {
272                         // TODO Auto-generated catch block
273                         e.printStackTrace();
274                     }
275                     System.out.println("busca");
276                     try { haTrabalho.acquire();
277                         livreMyMensagem.acquire();
278                     } catch (InterruptedException e) {}
279                     myMensagem= bufferCircular.removerElemento();
280                     try { acessoMyMensagem.acquire(); }
281                     catch (InterruptedException e) {}
282
283                     acessoMyMensagem.release();
284                     ocupadaMyMensagem.release();
285                     msg = getMensagem();
286                     if(msg.getTipo() != 4) {
287                         System.out.println("existe msg");
288
289                         state = receberMensagem;
290                         break;
291                     }
292                     else {
293                         if(bd.getMensagens().size()!=0 && bd.isLigado()) {
294                             state = esperarTempoExecucao;
295                             break;
296                         }

```

```

297     }
298
299
300     case esperarTempoExecucao:
301         msg = bd.getMensagens().get(0);
302         System.out.println("esperaExec id:" + msg.getId());
303         int tipo = msg.getTipo();
304         System.out.println("Tipo:" + tipo);
305         try {
306             switch (tipo) {
307                 case RETA:
308                     gui.txtLog.append(" 0 robot avançou " + msg.getArg1
309 () + "\n");
310                     bd.getRobot().Reta(msg.getArg1());
311                     bd.getRobot().Parar(false);
312                     Thread.sleep((long) ((Math.abs(msg.getArg1())) /
313 0.03));
314                     break;
315                 case CURVARDIR:
316                     gui.txtLog.append(" 0 robot virou direita com raio
317 = " + msg.getArg1() + " e angulo = " + msg.getArg2() + "\n");
318                     bd.getRobot().CurvarDireita(msg.getArg1(), msg.
319 getArg2());
320                     bd.getRobot().Parar(false);
321                     Thread.sleep((long) ((msg.getArg1() * (msg.getArg2
322 () * 0.017)) / 0.03));
323                     break;
324                 case CURVARESQ:
325                     gui.txtLog.append(" 0 robot virou esquerda com raio
326 = " + msg.getArg1() + " e angulo = " + msg.getArg2() + "\n");
327                     bd.getRobot().CurvarEsquerda(msg.getArg1(), msg.
328 getArg2());
329                     bd.getRobot().Parar(false);
330                     Thread.sleep((long) ((msg.getArg1() * (msg.getArg2
331 () * 0.017)) / 0.03));
332                     break;
333             }
334         } catch (InterruptedException e) {
335             // TODO Auto-generated catch block
336             e.printStackTrace();
337         }
338         bd.removeMensagem();
339         state = dormir;
340         break;
341     }
342 }
343 System.out.println("sai");
344
345 }
346
347
348 /*public static void main(String[] args) throws
349 InterruptedException
350 {
351     App_Subdito app = new App_Subdito();
352     System.out.println("A aplicação começou.");
353     app.run();
354     System.out.println("A aplicação terminou.");
355 }*/
356
357 }

```