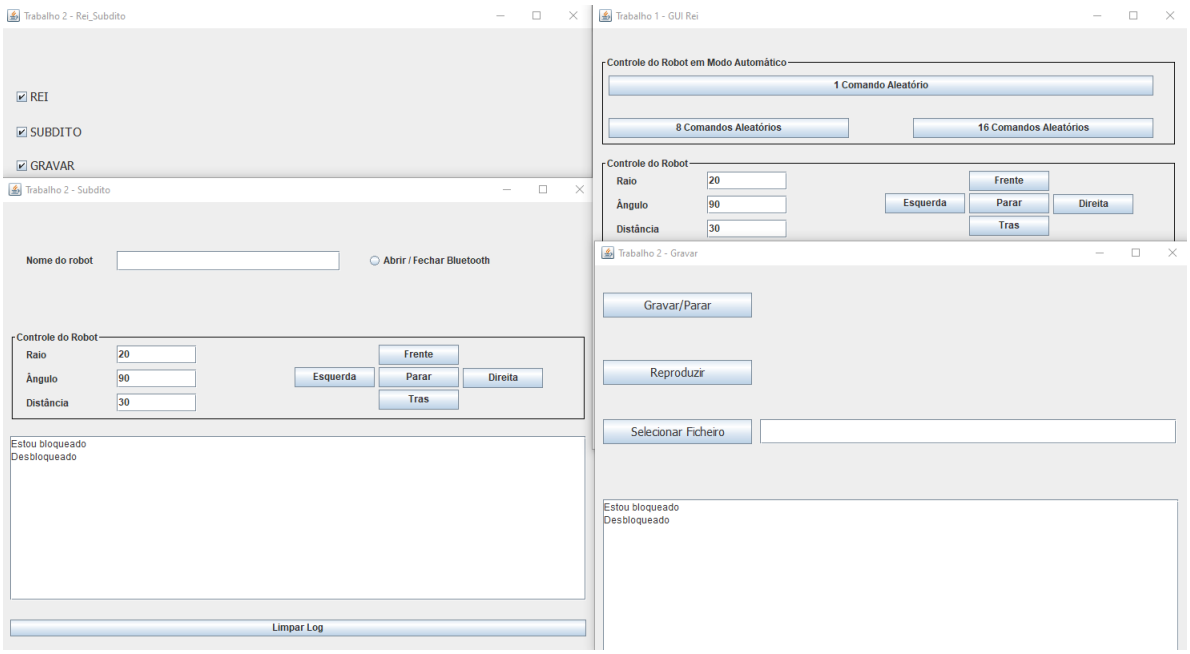
Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

Fundamentos de Sistemas Operativos - 2324SI

2º Trabalho Prático - Aula prática 6



Docente Carlos Carvalho

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

26 de setembro de 2024

Conteúdo

Lista de Figuras

1 Introdução

Esta aula consistiu no teste da aplicação multitarefa completa. Demonstração ao Professor e respetiva validação. Portanto vamos apenas descrever as alterações efetuadas para aumentar a qualidade do nosso trabalho pois este já estava completamente funcional.

2 Desenvolvimento

2.1 Estado Gravar

Não ficou mencionado no relatório da aula 4 mas vamos explicar agora o que acontece na tarefa gravar quando entramos em modo gravação. Como a gravação é feita através das mensagens que o súbdito recebe, a tarefa gravar não tem contacto direto, logo, não vai precisar de realizar nenhuma ação.

```
case gravar:
    System.out.println(bd.getGravarS().availablePermits());
    gui.write("Estou a Gravar \n");
    try {
        bd.getGravarS().acquire();
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    gui.write("Parei de gravar \n");
    state = dormir;
    break;
```

Figura 1: Estado 'Gravar' - Classe App_Gravar

Como podemos observar, quando entramos neste estado ficamos à espera do semáforo que só vai ter "permits"quando o utilizador escolher parar de gravar. Assim não estamos a gastar recursos ao computador sem necessidade.

2.2 Gravador

Adicionámos também uma mensagem à GUI_Gravar quando uma mensagem é guardada. Isto dá-nos uma confirmação extra de que as mensagens estão a ser guardadas com sucesso e a ordem pela qual isto é feito.

```
void reta(Mensagem msg)
{
    // Try-with-resources to automatically close resources (like FileWriter)
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(bdGravar.getNome(), true))) {
        // Write data to the file
        writer.write(
                ""+ msg.tipo + "," +
                "" + msg.arg1 + "," +
                "" + System.currentTimeMillis() + "" + EOL);

        System.out.println("Data has been successfully saved to the file.");
        guiGravar.write(msg.toString());

    } catch (IOException e) {
        // Handle exceptions
        System.out.println("No file.");
        e.printStackTrace();
    }
}
```

Figura 2: Método 'reta' - Classe Gravador

Foi implementado em cada um dos métodos da classe Gravador da mesma forma que na imagem acima. Quando criamos uma instância de Gravador passamos não só a base de dados como agora a GUI gravar.

3 Conclusões

Depois de termos limado as nossas arestas damos por concluído o trabalho prático 2 com sucesso. Ao longo desta disciplina conseguimos aprender os fundamentos de sistemas operativos e como estes funcionam nas nossas máquinas. Com estes dois trabalhos pudemos consolidar estes conhecimentos como por exemplo os semáforos e os monitores (apesar de não os usarmos experimentamos-los), ferramentas que nos abrem um novo mundo na linguagem Java.

4 Bibliografia

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

5 Código Java Todas as Classes

```
1
2 Classe App_Gravar
3
4 package ptrabalho;
5
6 import java.awt.EventQueue;
7
8 import javax.swing.JFrame;
9 import javax.swing.JPanel;
10 import javax.swing.border.EmptyBorder;
11 import javax.swing.JCheckBox;
12 import java.awt.Font;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.awt.event.WindowAdapter;
16 import java.awt.event.WindowEvent;
17 import java.util.concurrent.Semaphore;
18
19 import javax.swing.SwingConstants;
20
21 public class GUI_Rei_Subdito extends GUI_Base {
22
23     private JPanel contentPane;
24     private App_Rei appRei;
25     private App_Subdito appSub;
26     private App_Gravar appGravar;
27     private GUI_Subdito gui_Subdito;
28     private GUI_Gravar gui_Gravar;
29
30     private BD_Rei bdRei = new BD_Rei();
31     private BD_Gravar bdGravar = new BD_Gravar();
32     private Gravador gravador = new Gravador(bdGravar);
33     private BD_Subdito bdSub = new BD_Subdito(gravador);
34     private Semaphore subAvailable = new Semaphore(0);
35     private Semaphore reiAvailable = new Semaphore(0);
36     private Semaphore gravarAvailable = new Semaphore(0);
37
38     private BufferCircular bcG = new BufferCircular();
39     private Semaphore haTrabalhoG = new Semaphore(1);
40
41
42     public GUI_Rei_Subdito(BD_Base bd, BufferCircular bc, Semaphore
       ht)
43   {
44     super(bd);
45     EventQueue.invokeLater(new Runnable()
46     {
47       public void run()
48       {
49         try
50         {
51           init_Rei_Subdito(bc, ht);
52
53         } catch (Exception e)
54         {
55           e.printStackTrace();
56         }
57       }
58     });
```

```
59  }
60
61
62  public void init_Rei_Subdito(BufferCircular bc, Semaphore ht) {
63
64    setTitle("Trabalho 2 - Rei_Subdito");
65    //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66    setBounds(0, 0, 754, 600);
67
68
69
70      //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
71      getContentPane().setLayout(null);
72      JCheckBox chckbxRei = new JCheckBox("REI");
73      chckbxRei.setHorizontalAlignment(SwingConstants.LEFT);
74      chckbxRei.setFont(new Font("Tahoma", Font.PLAIN, 15));
75      chckbxRei.setBounds(16, 56, 97, 59);
76      getContentPane().add(chckbxRei);
77      JCheckBox chckbxSubdito = new JCheckBox("SUBDITO");
78      chckbxSubdito.setHorizontalAlignment(SwingConstants.LEFT);
79      chckbxSubdito.setFont(new Font("Tahoma", Font.PLAIN, 15));
80      chckbxSubdito.setBounds(16, 100, 195, 59);
81      getContentPane().add(chckbxSubdito);
82      JCheckBox chckbxGravar = new JCheckBox("GRAVAR");
83      chckbxGravar.setHorizontalAlignment(SwingConstants.LEFT);
84      chckbxGravar.setFont(new Font("Tahoma", Font.PLAIN, 15));
85      chckbxGravar.setBounds(16, 142, 108, 59);
86      getContentPane().add(chckbxGravar);
87      appRei = new App_Rei(bdRei, bc, ht, reiAvailable);
88      Thread tRei = new Thread(appRei);
89      tRei.start();
90
91      //appRei.run();
92      //gui_Subdito = new GUI_Subdito(bdSub);
93      appSub = new App_Subdito(bdSub, bc, ht, subAvailable, bcG,
    haTrabalhoG);
94      Thread tSub = new Thread(appSub);
95      tSub.start();
96      appGravar = new App_Gravar(bdGravar, gravador,
    gravarAvailable, bcG, haTrabalhoG);
97      Thread tGravar = new Thread(appGravar);
98      tGravar.start();
99
100     chckbxRei.addActionListener(new ActionListener()
101   {
102     public void actionPerformed(ActionEvent e)
103     {
104       if(chckbxRei.isSelected()) {
105         if (!appRei.gui.isVisible())
106           appRei.gui = new GUI_Rei(bdRei);
107         appRei.gui.start();
108         System.out.println(bdRei.getMensagens().size());
109         reiAvailable.release();
110         reiAvailable.release();
111         //appRei.run();
112         //t.run();
113         write("Ativei a GUI_REI \n");
114       }
115       else {
116         appRei.gui.off();
```

```
117           try {
118             reiAvailable.acquire();
119           } catch (InterruptedException e1) {
120             // TODO Auto-generated catch block
121             e1.printStackTrace();
122           }
123           write("Desativei a GUI_REI \n");
124         }
125 
126       }
127     });
128 
129       chckbxSubdito.addActionListener(new ActionListener()
130     {
131       public void actionPerformed(ActionEvent e)
132       {
133         if(chckbxSubdito.isSelected()) {
134           if (!appSub.gui.isVisible())
135             appSub.gui = new GUI_Subdito(bdSub);
136           appSub.gui.start();
137           System.out.println(bdRei.getMensagens().size());
138           subAvailable.release();
139           subAvailable.release();
140           write("Ativei a GUI_SUBDITO \n");
141         }
142         else {
143           appSub.gui.off();
144           try {
145             subAvailable.acquire();
146           } catch (InterruptedException e1) {
147             // TODO Auto-generated catch block
148             e1.printStackTrace();
149           }
150           write("Desativei a GUI_SUBDITO \n");
151         }
152 
153 
154       }
155     });
156 
157       chckbxGravar.addActionListener(new ActionListener()
158     {
159       public void actionPerformed(ActionEvent e)
160       {
161         if(chckbxGravar.isSelected()) {
162           if (!appGravar.gui.isVisible())
163             appGravar.gui = new GUI_Gravar(bdGravar);
164           appGravar.gui.start();
165           gravarAvailable.release();
166           gravarAvailable.release();
167           bdGravar.setGravarOff(true);
168           write("Ativei a GUI_GRAVAR \n");
169         }
170         else {
171           appGravar.gui.off();
172           try {
173             bdGravar.setGravarOff(false);
174             gravarAvailable.acquire();
175           } catch (InterruptedException e1) {
176             // TODO Auto-generated catch block
```

```
177             e1.printStackTrace();
178           }
179           write("Desativei a GUI_GRAVAR \n");
180         }
181 
182 
183       }
184     });
185 
186       addWindowListener(new WindowAdapter(){
187           public void windowClosing(WindowEvent e){
188               if (appSub.getBD().isLigado())
189               {
190                 appSub.gui.write(" Desconectando o robot ... \n");
191                 write(" Desconectando o robot ... \n");
192                 appSub.getBD().getRobot().getRobot().CloseEV3();
193                 appSub.getBD().setLigado(false);
194                 System.exit(0);
195               }
196               else {
197                   System.out.println("Closing program");
198                   System.exit(0);
199               }
200           }
201       });
202 
203       setVisible(true);
204   }
205 
206 
207 }
208 
209 Classe App_Rei_Subdito
210 
211 package ptrabalho;
212 
213 import java.util.concurrent.Semaphore;
214 
215 public class App_Rei_Subdito
216 {
217     @SuppressWarnings("unused")
218     private GUI_Rei_Subdito gui;
219     private BD_Base bd = new BD_Base();
220     private BufferCircular bc = new BufferCircular();
221     private Semaphore ht = new Semaphore(1);
222 
223     public App_Rei_Subdito(String[] args)
224     {
225        gui = new GUI_Rei_Subdito(bd, bc, ht);
226     }
227 
228 
229 
230     public void run() throws InterruptedException
231     {
232         while(true) {
233           //System.out.println("sai");
234             Thread.sleep(100);
235         }
236 
```

```
237
238     }
239
240     public static void main(String[] args) throws
        InterruptedException
241     {
242
243       App_Rei_Subdito app = new App_Rei_Subdito(args);
244         System.out.println("A aplicação começou.");
245         app.run();
246         System.out.println("A aplicação terminou.");
247     }
248
249 }
250
251 Classe App_Rei
252
253 package ptrabalho;
254
255 import java.util.concurrent.Semaphore;
256
257 public class App_Rei extends Thread
258 {
259     @SuppressWarnings("unused")
260     protected GUI_Rei gui;
261     private BD_Rei bd;
262     Mensagem msg;
263     Mensagem myMensagem = null;
264     private int state = 2;
265     private int counter = 0;
266     private final int escreverMensagem = 1;
267     private final int dormir = 2;
268     private final int bloqueado = 4;
269
270     BufferCircular bufferCircular;
271     Semaphore haTrabalho, livreMyMensagem, ocupadaMyMensagem,
        acessoMyMensagem, reiAvailable;
272
273
274     public App_Rei(BD_Rei bdRei, BufferCircular bc, Semaphore ht,
        Semaphore ra)
275     {
276       bd = bdRei;
277         gui = new GUI_Rei(bd);
278         bufferCircular= bc;
279       haTrabalho= ht;
280       reiAvailable = ra;
281       myMensagem = null;
282       livreMyMensagem= new Semaphore(1);
283       ocupadaMyMensagem= new Semaphore(0);
284       acessoMyMensagem= new Semaphore(1);
285     }
286
287     public BD_Rei getBD()
288     {
289         return bd;
290     }
291
292     public void setMensagem(Mensagem m)
293     {
```

```
294       try {
295       livreMyMensagem.acquire();
296       acessoMyMensagem.acquire();
297       } catch (InterruptedException e) {}
298       myMensagem= m;
299       acessoMyMensagem.release();
300       ocupadaMyMensagem.release();
301     }
302
303
304     public void run()
305     {
306         while(true) {
307
308           switch (state) {
309
310             case escreverMensagem:
311               System.out.println("escreve");
312               System.out.println("Mensagens à espera: " + bd.
    getMensagens().size());
313               msg = bd.getMensagens().get(0);
314               msg.setId(counter);
315               setMensagem(msg);
316               try {
317                 ocupadaMyMensagem.acquire();
318                 acessoMyMensagem.acquire();
319               } catch (InterruptedException e) {}
320                 bufferCircular.inserirElemento(myMensagem);
321                 acessoMyMensagem.release();
322                 livreMyMensagem.release();
323                 haTrabalho.release();
324                 System.out.println("ht:" + haTrabalho.
    availablePermits());
325                 gui.write(" Enviei = " + msg + "\n");
326                 counter =++ counter % bd.getNMensagens();
327                 bd.removeMensagem();
328                 state = dormir;
329                 break;
330
331             case dormir:
332               //System.out.println("sleep");
333               try {
334                 Thread.sleep(1000);
335                 //System.out.println("permits" + reiAvailable.
    availablePermits());
336                 if(reiAvailable.availablePermits() == 0) {
337                   state = bloqueado;
338                   break;
339                 }
340               } catch (InterruptedException e) {
341                 // TODO Auto-generated catch block
342                 e.printStackTrace();
343               }
344               //System.out.println(bd.getMensagens().size());
345               if(bd.getMensagens().size() !=0) {
346                 state = escreverMensagem;
347                 break;
348               }
349               else {break;}
350
```

```
351             case bloqueado:
352               try {
353                 gui.write("Estou bloqueado \n");
354                 reiAvailable.acquire();
355                 gui.write("Desbloqueado \n");
356                 state = dormir;
357                 break;
358               } catch (InterruptedException e) {
359                 // TODO Auto-generated catch block
360                 e.printStackTrace();
361               }
362
363       }
364     }
365     /*System.out.println("sai");
366         try {
367       Thread.sleep(100);
368     } catch (InterruptedException e) {
369       // TODO Auto-generated catch block
370       e.printStackTrace();
371     }*/
372
373     }
374
375 }
376
377
378 Classe App_Subdito
379
380 package ptrabalho;
381 import java.lang.Math;
382 import java.util.concurrent.Semaphore;
383
384 public class App_Subdito extends Thread
385 {
386     @SuppressWarnings("unused")
387     protected GUI_Subdito gui;
388     private BD_Subdito bd;
389     Mensagem msg = null;
390     Mensagem myMensagem = null;
391     private int state = 2;
392     private int counter = 0;
393     private final int receberMensagem = 1;
394     private final int dormir = 2;
395     private final int esperarTempoExecucao = 3;
396     private final int bloqueado = 4;
397     private final int reproduzir = 5;
398     private final int RETA = 1;
399     private final int CURVARDIR = 2;
400     private final int CURVARESQ = 3;
401     private final int PARAR = 4;
402     private boolean aReproduzir = false;
403
404     BufferCircular bufferCircular, bufferCircularReproduzir;
405     Semaphore haTrabalho, livreMyMensagem, ocupadaMyMensagem,
    acessoMyMensagem, subAvailable, haTrabalhoG;
406
407     public App_Subdito(BD_Subdito bdSub, BufferCircular bc,
    Semaphore ht, Semaphore sub, BufferCircular bcG, Semaphore htG)
408     {
```

```
409       bd = bdSub;
410
411         gui = new GUI_Subdito(bd);
412
413         subAvailable = sub;
414         bufferCircular= bc;
415         bufferCircularReproduzir = bcG;
416         haTrabalho= ht;
417         haTrabalhoG= htG;
418         myMensagem= null;
419         livreMyMensagem= new Semaphore(1);
420         ocupadaMyMensagem= new Semaphore(0);
421         acessoMyMensagem= new Semaphore(1);
422     }
423
424     public BD_Subdito getBD()
425     {
426         return bd;
427     }
428
429     public Mensagem getMensagem() {
430       try { ocupadaMyMensagem.acquire();
431       acessoMyMensagem.acquire();
432       } catch (InterruptedException e) {}
433       Mensagem s= myMensagem;
434       acessoMyMensagem.release();
435       livreMyMensagem.release();
436       return s;
437       }
438
439
440     public void run()
441     {
442           while(!bd.getTerminar()) {
443
444               switch (state) {
445
446               case receberMensagem:
447                 System.out.println("recebe");
448                 gui.write(" Recebi = " + msg + "\n");
449                 bd.addMensagem(msg);
450                 if (bd.isLigado())
451                   state = esperarTempoExecucao;
452                 else
453                   state = dormir;
454                 break;
455
456           case dormir:
457             try {
458             Thread.sleep(100);
459             //System.out.println("permits" + subAvailable.
    availablePermits());
460                 if(subAvailable.availablePermits() == 0) {
461                   state = bloqueado;
462                   break;
463                 }
464           } catch (InterruptedException e) {
465             // TODO Auto-generated catch block
466             e.printStackTrace();
467           }
```

```
468             //System.out.println("busca" + aReproduzir );
469             if(bufferCircularReproduzir.available() != 0 ||
    aReproduzir)
470             {
471               System.out.println("REPRODUZIR");
472               state = reproduzir;
473               break;
474             }
475             if (bufferCircular.available() != 0) {
476               //System.out.println(haTrabalho.availablePermits());
477               try {
478                 haTrabalho.acquire();
479                 livreMyMensagem.acquire();
480                 } catch (InterruptedException e) {}
481               Mensagem m= bufferCircular.removerElemento();
482               try { acessoMyMensagem.acquire(); }
483               catch (InterruptedException e) {}
484               myMensagem = m;
485               acessoMyMensagem.release();
486               ocupadaMyMensagem.release();
487               msg = getMensagem();
488               //System.out.println(bd.getNMensagens());
489               state = receberMensagem;
490               break;
491             }
492             else
493               if (bd.isLigado() && bd.getMensagens().size() != 0)
494                 state = esperarTempoExecucao;
495               break;
496
497
498           case esperarTempoExecucao:
499             msg = bd.getMensagens().get(0);
500             //System.out.println("esperaExec id:" + msg.getId());
501             int tipo = msg.getTipo();
502             //System.out.println("Tipo:" + tipo);
503             try {
504             switch (tipo) {
505               case RETA:
506                 gui.write(" O robot avançou " + msg.getArg1() + "\n
    ");
507                 bd.getRobot().reta(msg);
508                 bd.getRobot().parar(msg);
509                 Thread.sleep((long) ((Math.abs(msg.getArg1())) /
    0.03));
510                 break;
511               case CURVARDIR:
512                 gui.write(" O robot virou direita com raio = " +
    msg.getArg1() +" e angulo = " + msg.getArg2() + "\n");
513                 bd.getRobot().curvarDireita(msg);
514                 bd.getRobot().parar(msg);
515                 Thread.sleep((long) ((msg.getArg1() * (msg.getArg2
    () * 0.017)) / 0.03));
516                 break;
517               case CURVARESQ:
518                 gui.write(" O robot virou esquerda com raio = " +
    msg.getArg1() +" e angulo = " + msg.getArg2() + "\n");
519                   bd.getRobot().curvarEsquerda(msg);
520                   bd.getRobot().parar(msg);
521                 Thread.sleep((long) ((msg.getArg1() * (msg.getArg2
```

```
() * 0.017)) / 0.03));
522                 break;
523               case PARAR:
524                 gui.write(" O robot parou \n");
525                   bd.getRobot().parar(msg);
526                 Thread.sleep(100);
527                 break;
528             }
529             } catch (InterruptedException e) {
530             // TODO Auto-generated catch block
531             e.printStackTrace();
532           }
533             bd.removeMensagem();
534             state = dormir;
535             break;
536
537           case bloqueado:
538               try {
539                 gui.write("Estou bloqueado \n");
540                 subAvailable.acquire();
541                 gui.write("Desbloqueado \n");
542                 state = dormir;
543                 break;
544               } catch (InterruptedException e) {
545                 // TODO Auto-generated catch block
546                 e.printStackTrace();
547               }
548
549           case reproduzir:
550
551             try {
552               haTrabalhoG.acquire();
553             System.out.println("haT");
554             livreMyMensagem.acquire();
555           } catch (InterruptedException e) {}
556           Mensagem m= bufferCircularReproduzir.removerElemento();
557
558           try { acessoMyMensagem.acquire(); }
559           catch (InterruptedException e) {}
560           myMensagem = m;
561           acessoMyMensagem.release();
562           ocupadaMyMensagem.release();
563           msg = getMensagem();
564           //System.out.println(bd.getNMensagens());
565
566           if (msg.getTipo() != 5)
567             gui.write(" Vou reproduzir = " + msg + "\n");
568           else
569             gui.write(" Acabou a reprodução \n");
570
571                 aReproduzir = true;
572                 //if (bd.isLigado())
573                 if(msg.tipo == 5) {
574                   aReproduzir = false;
575                   state = dormir;
576                   break;
577                 }
578                 bd.addMensagem(msg);
579                 state = esperarTempoExecucao;
580           break;
```

```
            }
        }
        System.out.println("sai");

    }

}



Classe BD_Base

package ptrabalho;

import java.util.ArrayList;
import java.util.List;

public class BD_Base {

  protected int distance;
  protected int angulo;
    protected int raio;
    protected String file = "";
    protected int nMensagens = 8;
    private List<Mensagem> mensagens = new ArrayList<>();

  public BD_Base() {
    distance = 30;
        angulo = 90;
        raio = 20;
  }

  public int getDist()
    {
        return distance;
    }

    public void setDist(int i)
    {
        distance = i;
    }

    public int getAng()
    {
        return angulo;
    }

    public void setAng(int i)
    {
        angulo = i;
    }

    public int getRaio()
    {
        return raio;
    }

    public void setRaio(int i)
    {
```

```
641         raio = i;
642     }
643 
644     public String getFile()
645     {
646         return file;
647     }
648 
649     public void setFile(String f)
650     {
651         file = f;
652     }
653 
654     public int getNMensagens()
655     {
656         return nMensagens;
657     }
658 
659     public void setNMensagens(int i)
660     {
661       nMensagens = i;
662     }
663 
664     public void addMensagem(Mensagem msg)
665     {
666       mensagens.add(msg);
667     }
668 
669     public void removeMensagem()
670     {
671       mensagens.remove(0);
672     }
673 
674     public List<Mensagem> getMensagens()
675     {
676     return mensagens;
677     }
678 
679 
680 }
681 
682 
683 Classe BD_Gravar
684 
685 package ptrabalho;
686 
687 import java.util.concurrent.Semaphore;
688 
689 public class BD_Gravar extends BD_Base {
690 
691 
692   private String nome = null;
693   private boolean gravar, reproduzir, gravarOff = false;
694   private Semaphore gravarS = new Semaphore(0);
695 
696   public BD_Gravar()
697     {
698       super();
699 
700     }
```

```
701
702   public void setNome(String n)
703     {
704         nome = n;
705     }
706
707     public String getNome()
708     {
709         return nome;
710     }
711
712     public void setGravar(boolean g) throws InterruptedException
713     {
714         gravar = g;
715         if (!g)
716           gravarS.release();
717
718     }
719
720     public boolean getGravar()
721     {
722         return gravar;
723     }
724
725     public Semaphore getGravarS()
726     {
727       return gravarS;
728     }
729
730     public void setReproduzir(boolean r)
731     {
732         reproduzir = r;
733     }
734
735     public boolean getReproduzir()
736     {
737         return reproduzir;
738     }
739
740     public void setGravarOff(boolean g)
741     {
742       gravarOff = g;
743     }
744
745     public boolean getGravarOff()
746     {
747       return gravarOff;
748     }
749
750
751 }
752
753 Classe BD_Rei
754
755 package ptrabalho;
756
757
758 //import robot.RobotLegoEV3;
759
760 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;
    }

    public void setNome(String n)
    {
        nome = n;
    }

    public String getNome()
    {
        return nome;
    }



}

Classe BD_Subdito

package ptrabalho;
import robot.RobotLegoEV3;

public class BD_Subdito extends BD_Base
{
  private myRobotLegoEV3 myRobot;
    private RobotLegoEV3 robot;
```

```
821     private boolean terminar;
822     private boolean ligado;
823 
824     private String nome;
825 
826     public BD_Subdito(Gravador g)
827     {
828       super();
829       myRobot = new myRobotLegoEV3(g);
830         //robot = new RobotLegoEV3();
831         terminar = false;
832         ligado = false;
833 
834     }
835 
836     public myRobotLegoEV3 getRobot()
837     {
838         return myRobot;
839     }
840 
841     public boolean getTerminar()
842     {
843         return terminar;
844     }
845 
846     public void setTerminar(boolean b)
847     {
848         terminar = b;
849     }
850 
851     public boolean isLigado()
852     {
853         return ligado;
854     }
855 
856     public void setLigado(boolean b)
857     {
858         ligado = b;
859     }
860 
861     public void setNome(String n)
862     {
863         nome = n;
864     }
865 
866     public String getNome()
867     {
868         return nome;
869     }
870 
871 }
872 
873 Classe Buffer_Circular
874 
875 package ptrabalho;
876 
877 import java.util.concurrent.Semaphore;
878 
879 public class BufferCircular {
880   final int dimensaoBuffer= 8;
```

```
881   Mensagem[] bufferCircular;
882   int putBuffer, getBuffer;
883   // o semáforo elementosLivres indica se há posições livres para
       inserir Strings
884   // o semáforo acessoElemento garante exclusão mútua no acesso a
       um elemento
885   // o semáforo elementosOcupados indica se há posições com Strings
       válidas
886   Semaphore elementosLivres, acessoElemento, elementosOcupados;
887
888   public BufferCircular(){
889     bufferCircular= new Mensagem[dimensaoBuffer];
890     putBuffer= 0;
891     getBuffer= 0;
892     elementosLivres= new Semaphore(dimensaoBuffer);
893     elementosOcupados= new Semaphore(0);
894     acessoElemento= new Semaphore(1);
895   }
896
897   public void inserirElemento(Mensagem msg){
898     try {
899       elementosLivres.acquire();
900       acessoElemento.acquire();
901       bufferCircular[putBuffer]= msg;
902       System.out.print(msg);
903       putBuffer= ++putBuffer % dimensaoBuffer;
904       acessoElemento.release();
905     } catch (InterruptedException e) {}
906       elementosOcupados.release();
907     }
908
909   public Mensagem removerElemento() {
910      Mensagem msg= null;
911      try {
912      elementosOcupados.acquire();
913      acessoElemento.acquire();
914      } catch (InterruptedException e) {}
915
916      msg = bufferCircular[getBuffer];
917      getBuffer= ++getBuffer % dimensaoBuffer;
918      acessoElemento.release();
919      elementosLivres.release();
920      return msg;
921     }
922
923   public int available()
924   {
925     return elementosOcupados.availablePermits();
926   }
927 }
928
929 Classe Gravador
930
931 package ptrabalho;
932
933 import java.io.BufferedWriter;
934 import java.io.FileWriter;
935 import java.io.IOException;
936
937 public class Gravador {
```

```
938
939
940   private boolean gravar;
941   protected BD_Gravar bdGravar;
942   protected GUI_Gravar guiGravar;
943   private final String EOL = System.lineSeparator();
944
945   public Gravador(BD_Gravar bd, GUI_Gravar gui)
946   {
947     gravar = false;
948     bdGravar = bd;
949     guiGravar = gui;
950   }
951
952   public void setGravar(boolean g)
953   {
954     gravar = g;
955   }
956
957   public boolean getGravar()
958   {
959     return bdGravar.getGravar();
960   }
961
962   void reta(Mensagem msg)
963   {
964     // Try-with-resources to automatically close resources (like
       FileWriter)
965         try (BufferedWriter writer = new BufferedWriter(new
       FileWriter(bdGravar.getNome(), true))) {
966             // Write data to the file
967             writer.write(
968                 ""+ msg.tipo + "," +
969                 "" + msg.arg1 + "," +
970                 "" + System.currentTimeMillis() + "" + EOL);
971
972             System.out.println("Data has been successfully saved to
       the file.");
973             guiGravar.write(msg.toString());
974
975         } catch (IOException e) {
976             // Handle exceptions
977           System.out.println("No file.");
978             e.printStackTrace();
979         }
980   }
981
982   void curvarDireita(Mensagem msg)
983   {
984     // Try-with-resources to automatically close resources (like
       FileWriter)
985         try (BufferedWriter writer = new BufferedWriter(new
       FileWriter(bdGravar.getNome(), true))) {
986             // Write data to the file
987             writer.write(
988                 "" + msg.tipo + "," +
989                 "" + msg.arg1 + "," +
990                 "" + msg.arg2 + "," +
991                 "" + System.currentTimeMillis() + "" + EOL);
992
```

```
            System.out.println("Data has been successfully saved to the file.");
            guiGravar.write(msg.toString());
        } catch (IOException e) {
            // Handle exceptions
            e.printStackTrace();
        }
  }

  void curvarEsquerda(Mensagem msg)
  {
    // Try-with-resources to automatically close resources (like FileWriter)
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(bdGravar.getNome(), true))) {
            // Write data to the file
            writer.write(
                "" + msg.tipo + "," +
                "" + msg.arg1 + "," +
                "" + msg.arg2 + "," +
                "" + System.currentTimeMillis() + "" + EOL);

            System.out.println("Data has been successfully saved to the file.");
            guiGravar.write(msg.toString());
        } catch (IOException e) {
            // Handle exceptions
            e.printStackTrace();
        }
  }

  void parar(Mensagem msg)
  {
    // Try-with-resources to automatically close resources (like FileWriter)
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(bdGravar.getNome(), true))) {
            // Write data to the file
            writer.write("4" + EOL);
            System.out.println("Data has been successfully saved to the file.");
            guiGravar.write(msg.toString());
        } catch (IOException e) {
            // Handle exceptions
            e.printStackTrace();
        }
  }
}
```

Classe GUI_Base

```
package ptrabalho;

import java.awt.EventQueue;
import java.awt.Font;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollBar;
```

```
import javax.swing.border.EmptyBorder;
import javax.swing.JTextArea;
import java.awt.ScrollPane;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JToggleButton;
import javax.swing.JScrollPane;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;

public class GUI_Base extends JFrame {

  protected JPanel contentPane;
  protected JTextArea txtLog;
  protected JScrollPane scrollPane;

  public GUI_Base(BD_Base bd) {
      //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setBounds(0, 0, 800, 500);
      contentPane = new JPanel();
      contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        txtLog = new JTextArea();
        txtLog.setBounds(1, 1, 719, 203);
        contentPane.add(txtLog);

        scrollPane = new JScrollPane(txtLog);
        scrollPane.setBounds(10, 292, 719, 205);

        JButton btnLimpaLog = new JButton("Limpar Log");
        btnLimpaLog.setBounds(10, 521, 719, 21);
    btnLimpaLog.setFont(new Font("Arial", Font.BOLD, 12));
    contentPane.add(scrollPane);
    contentPane.add(btnLimpaLog);

        btnLimpaLog.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        txtLog.setText("");
      }
    });
    }

  public void write(String txt)
  {
    EventQueue.invokeLater(new Runnable()
    {

      @Override
      public void run() {
        txtLog.append(txt);

        // Set the caret position to the end of the text
              txtLog.setCaretPosition(txtLog.getDocument().
   getLength());
```

```
1105
1106                // Set the caret position to the end of the text
1107                txtLog.setCaretPosition(txtLog.getDocument().
    getLength());
1108
1109                // Adjust the scrollbar to show the latest content
1110                JScrollBar verticalScrollBar = scrollPane.
    getVerticalScrollBar();
1111                verticalScrollBar.setValue(verticalScrollBar.
    getMaximum());
1112
1113                // Repaint the JTextArea and its parent
1114                txtLog.revalidate();
1115                txtLog.repaint();
1116
1117       }});
1118   }
1119 }
1120
1121 Classe GUI_BaseRS
1122
1123 package ptrabalho;
1124
1125 import java.awt.Color;
1126 import java.awt.Font;
1127 import java.awt.event.ActionEvent;
1128 import java.awt.event.ActionListener;
1129
1130 import javax.swing.BorderFactory;
1131 import javax.swing.JButton;
1132 import javax.swing.JLabel;
1133 import javax.swing.JPanel;
1134 import javax.swing.JTextField;
1135 import javax.swing.SwingConstants;
1136 import javax.swing.border.Border;
1137 import javax.swing.border.LineBorder;
1138 import javax.swing.border.TitledBorder;
1139
1140 public class GUI_BaseRS extends GUI_Base{
1141
1142   protected JPanel contentPane;
1143   protected JTextField txtRaio;
1144   protected JTextField txtAng;
1145   protected JTextField txtDist;
1146   protected JTextField txtFile;
1147   protected JButton btnFrt;
1148   protected JButton btnEsq;
1149   protected JButton btnDir;
1150   protected JButton btnParar;
1151   protected JButton btnTras;
1152
1153   public GUI_BaseRS(BD_Base bd) {
1154
1155     super(bd);
1156     Border borda_cont_robot = BorderFactory.createLineBorder(new
    Color(0,0,0),1);
1157     TitledBorder borda1 = BorderFactory.createTitledBorder(
    borda_cont_robot,"Controle do Robot");
1158
1159     JLabel lblRaio = new JLabel("Raio");
```

```
1160        lblRaio.setBounds(30, 179, 103, 25);
1161        getContentPane().add(lblRaio);
1162        lblRaio.setHorizontalAlignment(SwingConstants.LEFT);
1163        lblRaio.setFont(new Font("Arial", Font.BOLD, 12));
1164
1165        txtRaio = new JTextField();
1166        txtRaio.setBounds(143, 179, 100, 23);
1167        getContentPane().add(txtRaio);
1168        txtRaio.setText("20");
1169
1170        txtRaio.setFont(new Font("Arial", Font.BOLD, 12));
1171        txtRaio.setColumns(10);
1172
1173        JLabel lblngulo = new JLabel("Ângulo");
1174        lblngulo.setBounds(30, 209, 103, 25);
1175        getContentPane().add(lblngulo);
1176        lblngulo.setHorizontalAlignment(SwingConstants.LEFT);
1177        lblngulo.setFont(new Font("Arial", Font.BOLD, 12));
1178
1179        JLabel lblDistncia = new JLabel("Distância");
1180        lblDistncia.setBounds(30, 239, 103, 25);
1181        getContentPane().add(lblDistncia);
1182        lblDistncia.setHorizontalAlignment(SwingConstants.LEFT);
1183        lblDistncia.setFont(new Font("Arial", Font.BOLD, 12));
1184
1185        txtAng = new JTextField();
1186        txtAng.setBounds(143, 209, 100, 23);
1187        getContentPane().add(txtAng);
1188        txtAng.setText("90");
1189        txtAng.setFont(new Font("Arial", Font.BOLD, 12));
1190        txtAng.setColumns(10);
1191
1192        txtDist = new JTextField();
1193        txtDist.setBounds(143, 239, 100, 23);
1194        getContentPane().add(txtDist);
1195        txtDist.setText("30");
1196        txtDist.setFont(new Font("Arial", Font.BOLD, 12));
1197        txtDist.setColumns(10);
1198
1199        btnParar = new JButton("Parar");
1200        btnParar.setBounds(470, 206, 100, 25);
1201        getContentPane().add(btnParar);
1202        btnParar.setEnabled(false);
1203        btnParar.setFont(new Font("Arial", Font.BOLD, 12));
1204
1205        btnFrt = new JButton("Frente");
1206        btnFrt.setBounds(470, 178, 100, 25);
1207        getContentPane().add(btnFrt);
1208        btnFrt.setEnabled(false);
1209        btnFrt.setFont(new Font("Arial", Font.BOLD, 12));
1210
1211        btnDir = new JButton("Direita");
1212        btnDir.setBounds(575, 207, 100, 25);
1213        getContentPane().add(btnDir);
1214        btnDir.setEnabled(false);
1215        btnDir.setFont(new Font("Arial", Font.BOLD, 12));
1216
1217        btnEsq = new JButton("Esquerda");
1218        btnEsq.setBounds(365, 206, 100, 25);
1219        getContentPane().add(btnEsq);
```

```
1220      btnEsq.setEnabled(false);
1221      btnEsq.setFont(new Font("Arial", Font.BOLD, 12));
1222
1223      btnTras = new JButton("Tras");
1224      btnTras.setBounds(470, 234, 100, 25);
1225      getContentPane().add(btnTras);
1226      btnTras.setEnabled(false);
1227      btnTras.setFont(new Font("Arial", Font.BOLD, 12));
1228      JPanel panel_1 = new JPanel();
1229      panel_1.setBounds(10, 160, 719, 113);
1230      getContentPane().add(panel_1);
1231      panel_1.setFont(new Font("Arial", Font.BOLD, 12));
1232      panel_1.setName("Controle do Robot");
1233      panel_1.setBorder(new LineBorder(new Color(0, 0, 0)));
1234      panel_1.setBorder(borda1);
1235
1236      txtDist.addActionListener(new ActionListener()
1237      {
1238        public void actionPerformed(ActionEvent e)
1239        {
1240          bd.setDist(Integer.parseInt(txtDist.getText()));
1241          write(" A distancia é " + bd.getDist() + "\n");
1242
1243        }
1244      });
1245
1246      txtAng.addActionListener(new ActionListener()
1247      {
1248        public void actionPerformed(ActionEvent e)
1249        {
1250          bd.setAng(Integer.parseInt(txtAng.getText()));
1251          write(" O angulo é " + bd.getAng() + "\n");
1252        }
1253      });
1254
1255      txtRaio.addActionListener(new ActionListener()
1256      {
1257        public void actionPerformed(ActionEvent e)
1258        {
1259          bd.setRaio(Integer.parseInt(txtRaio.getText()));
1260          write(" A raio é " + bd.getRaio() + "\n");
1261        }
1262      });
1263
1264
1265    }
1266
1267    protected void start()
1268    {
1269      btnFrt.setEnabled(true);
1270      btnEsq.setEnabled(true);
1271      btnParar.setEnabled(true);
1272      btnDir.setEnabled(true);
1273      btnTras.setEnabled(true);
1274    }
1275
1276    protected void off()
1277    {
1278      btnFrt.setEnabled(false);
1279      btnEsq.setEnabled(false);
```

```
1280      btnParar.setEnabled(false);
1281      btnDir.setEnabled(false);
1282      btnTras.setEnabled(false);
1283    }
1284
1285  }
1286
1287  Classe GUI_Gravar
1288
1289  package ptrabalho;
1290
1291  import java.awt.EventQueue;
1292  import java.awt.event.ActionEvent;
1293  import java.awt.event.ActionListener;
1294  import java.io.BufferedWriter;
1295  import java.io.FileWriter;
1296  import java.io.IOException;
1297  import java.nio.file.Files;
1298  import java.nio.file.Path;
1299  import java.nio.file.Paths;
1300  import java.nio.file.StandardOpenOption;
1301
1302  import javax.swing.JFileChooser;
1303  import javax.swing.JFrame;
1304  import javax.swing.JPanel;
1305  import javax.swing.border.EmptyBorder;
1306  import javax.swing.JToggleButton;
1307  import javax.swing.WindowConstants;
1308
1309  import java.awt.Font;
1310  import javax.swing.JTextField;
1311  import javax.swing.JButton;
1312
1313  public class GUI_Gravar extends GUI_Base {
1314
1315      private JPanel contentPane;
1316      private JTextField txtFile;
1317      private JToggleButton tglGravar;
1318      JToggleButton tglReproduzir;
1319      protected JButton btnFile;
1320      protected BD_Gravar bdG;
1321
1322
1323      public GUI_Gravar(BD_Gravar bd)
1324    {
1325      super(bd);
1326      bdG = bd;
1327      EventQueue.invokeLater(new Runnable()
1328      {
1329        public void run()
1330        {
1331          try
1332          {
1333            init_Gravar(bd);
1334
1335          } catch (Exception e)
1336          {
1337            e.printStackTrace();
1338          }
1339        }
```

```
1340      });
1341    }
1342
1343    public void init_Gravar(BD_Gravar bd) {
1344
1345      setTitle("Trabalho 2 - Gravar");
1346      setDefaultCloseOperation(WindowConstants. DO_NOTHING_ON_CLOSE);
1347        setBounds(1000, 400, 760, 600);
1348        txtFile = new JTextField();
1349        txtFile.setBounds(206, 192, 520, 31);
1350        getContentPane().add(txtFile);
1351        txtFile.setColumns(10);
1352
1353        tglGravar = new JToggleButton("Gravar/Parar");
1354        tglGravar.setEnabled(false);
1355        tglGravar.setFont(new Font("Tahoma", Font.PLAIN, 15));
1356        tglGravar.setBounds(10, 34, 186, 31);
1357        getContentPane().add(tglGravar);
1358
1359        tglReproduzir = new JToggleButton("Reproduzir");
1360        tglReproduzir.setEnabled(false);
1361        tglReproduzir.setFont(new Font("Tahoma", Font.PLAIN, 15));
1362        tglReproduzir.setBounds(10, 118, 186, 31);
1363        getContentPane().add(tglReproduzir);
1364
1365        btnFile = new JButton("Selecionar Ficheiro");
1366        btnFile.setEnabled(false);
1367        btnFile.setFont(new Font("Tahoma", Font.PLAIN, 15));
1368        btnFile.setBounds(10, 192, 186, 31);
1369        getContentPane().add(btnFile);
1370        btnFile.addActionListener(new ActionListener()
1371        {
1372          public void actionPerformed(ActionEvent e)
1373          {
1374            JFileChooser fileChooser = new JFileChooser(System.
    getProperty("user.dir"));
1375                  if (fileChooser.showSaveDialog(null) ==
    JFileChooser.APPROVE_OPTION)
1376                  {
1377                    String file = fileChooser.getSelectedFile().
    getAbsolutePath();
1378                    bd.setNome(file);
1379                    write(" O nome do ficheiro é " + file + "\n");
1380                    txtFile.setText(file);
1381                  }
1382           }
1383       });
1384
1385        txtFile.addActionListener(new ActionListener()
1386     {
1387        public void actionPerformed(ActionEvent e)
1388        {
1389          bd.setNome(txtFile.getText());
1390          write(" O nome do ficheiro é " + bd.getNome() + "\n");
1391          //tglReproduzir.setEnabled(true);
1392          //tglGravar.setEnabled(true);
1393        }
1394     });
1395
1396        tglGravar.addActionListener(new ActionListener() {
```

```
1397            public void actionPerformed(ActionEvent e) {
1398                String nome = bd.getNome();  // Store the name in a
    variable
1399
1400                if (nome != null) {
1401                    // Check if the toggle button is selected (on)
1402                    boolean isOn = tglGravar.isSelected();
1403
1404                    // Set the 'gravar' boolean based on the toggle
    button state
1405                    try {
1406                        bd.setGravar(isOn);
1407                    } catch (InterruptedException e1) {
1408                        e1.printStackTrace();
1409                    }
1410
1411                    if (isOn) {
1412                        // Create an empty byte array
1413                        byte[] emptyBytes = new byte[0];
1414
1415                        // Open the file in write mode and overwrite
    it with an empty byte array
1416                        Path path = Paths.get(nome);  // Use the
    stored name
1417                        try {
1418                            Files.write(path, emptyBytes,
    StandardOpenOption.WRITE, StandardOpenOption.TRUNCATE_EXISTING);
1419                        } catch (IOException e1) {
1420                            e1.printStackTrace();
1421                        }
1422                    } else {
1423                        // Try-with-resources to automatically close
    resources (like FileWriter)
1424                        try (BufferedWriter writer = new
    BufferedWriter(new FileWriter(nome, true))) {
1425                            // Write data to the file
1426                            writer.write("5" + System.lineSeparator()
    );
1427                            System.out.println("Data has been
    successfully saved to the file.");
1428                        } catch (IOException e1) {
1429                            e1.printStackTrace();
1430                        }
1431                    }
1432                } else {
1433                    tglGravar.setSelected(false);
1434                    write("Selecione um ficheiro");
1435                }
1436            }
1437        });
1438
1439
1440        tglReproduzir.addActionListener(new ActionListener() {
1441            public void actionPerformed(ActionEvent e) {
1442                if (bdG.getNome() != null) {
1443                    // Check if the toggle button is selected (on)
1444                    boolean isOn = tglReproduzir.isSelected();
1445
1446                    // Set the 'gravar' boolean based on the toggle
    button state
```

```
1447                    bd.setReproduzir(isOn);
1448
1449                    // Append a message to the 'txtLog' text
    component
1450                    if (isOn)
1451                        write(" Começou a reproduzir \n");
1452                    else
1453                        write(" Parou de reproduzir \n");
1454                } else {
1455                    tglReproduzir.setSelected(false);
1456                }
1457            }
1458        });
1459
1460
1461
1462        setVisible(true);
1463      }
1464
1465    protected void start() {
1466        btnFile.setEnabled(true);
1467        if (bdG.getNome() != null) {
1468            tglReproduzir.setEnabled(true);
1469            tglGravar.setEnabled(true);
1470        } else {
1471            tglReproduzir.setEnabled(false);
1472            tglGravar.setEnabled(false);
1473        }
1474    }
1475
1476    protected void off() {
1477        btnFile.setEnabled(false);
1478        tglReproduzir.setEnabled(false);
1479        tglGravar.setEnabled(false);
1480    }
1481 }
1482
1483 Classe GUI_Rei_Subdito
1484
1485 package ptrabalho;
1486
1487 import java.awt.EventQueue;
1488
1489 import javax.swing.JFrame;
1490 import javax.swing.JPanel;
1491 import javax.swing.border.EmptyBorder;
1492 import javax.swing.JCheckBox;
1493 import java.awt.Font;
1494 import java.awt.event.ActionEvent;
1495 import java.awt.event.ActionListener;
1496 import java.awt.event.WindowAdapter;
1497 import java.awt.event.WindowEvent;
1498 import java.util.concurrent.Semaphore;
1499
1500 import javax.swing.SwingConstants;
1501
1502 public class GUI_Rei_Subdito extends GUI_Base {
1503
1504     private JPanel contentPane;
1505     private App_Rei appRei;
```

```
1506    private App_Subdito appSub;
1507    private App_Gravar appGravar;
1508    private GUI_Subdito gui_Subdito;
1509    private GUI_Gravar gui_Gravar;
1510
1511    private BD_Rei bdRei = new BD_Rei();
1512    private BD_Gravar bdGravar = new BD_Gravar();
1513    private Gravador gravador = new Gravador(bdGravar, gui_Gravar);
1514    private BD_Subdito bdSub = new BD_Subdito(gravador);
1515    private Semaphore subAvailable = new Semaphore(0);
1516    private Semaphore reiAvailable = new Semaphore(0);
1517    private Semaphore gravarAvailable = new Semaphore(0);
1518
1519    private BufferCircular bcG = new BufferCircular();
1520    private Semaphore haTrabalhoG = new Semaphore(1);
1521
1522
1523    public GUI_Rei_Subdito(BD_Base bd, BufferCircular bc, Semaphore
        ht)
1524  {
1525    super(bd);
1526    EventQueue.invokeLater(new Runnable()
1527    {
1528      public void run()
1529      {
1530        try
1531        {
1532          init_Rei_Subdito(bc, ht);
1533
1534        } catch (Exception e)
1535        {
1536          e.printStackTrace();
1537        }
1538      }
1539    });
1540  }
1541
1542
1543  public void init_Rei_Subdito(BufferCircular bc, Semaphore ht) {
1544
1545    setTitle("Trabalho 2 - Rei_Subdito");
1546    //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
1547    setBounds(0, 0, 754, 600);
1548
1549
1550
1551      //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
1552      getContentPane().setLayout(null);
1553      JCheckBox chckbxRei = new JCheckBox("REI");
1554      chckbxRei.setHorizontalAlignment(SwingConstants.LEFT);
1555      chckbxRei.setFont(new Font("Tahoma", Font.PLAIN, 15));
1556      chckbxRei.setBounds(16, 56, 97, 59);
1557      getContentPane().add(chckbxRei);
1558      JCheckBox chckbxSubdito = new JCheckBox("SUBDITO");
1559      chckbxSubdito.setHorizontalAlignment(SwingConstants.LEFT);
1560      chckbxSubdito.setFont(new Font("Tahoma", Font.PLAIN, 15));
1561      chckbxSubdito.setBounds(16, 100, 195, 59);
1562      getContentPane().add(chckbxSubdito);
1563      JCheckBox chckbxGravar = new JCheckBox("GRAVAR");
1564      chckbxGravar.setHorizontalAlignment(SwingConstants.LEFT);
```

```
1565        chckbxGravar.setFont(new Font("Tahoma", Font.PLAIN, 15));
1566        chckbxGravar.setBounds(16, 142, 108, 59);
1567        getContentPane().add(chckbxGravar);
1568        appRei = new App_Rei(bdRei, bc, ht, reiAvailable);
1569        Thread tRei = new Thread(appRei);
1570        tRei.start();
1571
1572        //appRei.run();
1573        //gui_Subdito = new GUI_Subdito(bdSub);
1574        appSub = new App_Subdito(bdSub, bc, ht, subAvailable, bcG,
    haTrabalhoG);
1575        Thread tSub = new Thread(appSub);
1576        tSub.start();
1577        appGravar = new App_Gravar(bdGravar, gravador,
    gravarAvailable, bcG, haTrabalhoG);
1578        Thread tGravar = new Thread(appGravar);
1579        tGravar.start();
1580
1581        chckbxRei.addActionListener(new ActionListener()
1582     {
1583        public void actionPerformed(ActionEvent e)
1584        {
1585          if(chckbxRei.isSelected()) {
1586            if (!appRei.gui.isVisible())
1587              appRei.gui = new GUI_Rei(bdRei);
1588            appRei.gui.start();
1589            System.out.println(bdRei.getMensagens().size());
1590            reiAvailable.release();
1591            reiAvailable.release();
1592            //appRei.run();
1593            //t.run();
1594            write("Ativei a GUI_REI \n");
1595          }
1596          else {
1597            appRei.gui.off();
1598            try {
1599              reiAvailable.acquire();
1600            } catch (InterruptedException e1) {
1601              // TODO Auto-generated catch block
1602              e1.printStackTrace();
1603            }
1604            write("Desativei a GUI_REI \n");
1605          }
1606
1607        }
1608     });
1609
1610        chckbxSubdito.addActionListener(new ActionListener()
1611     {
1612        public void actionPerformed(ActionEvent e)
1613        {
1614          if(chckbxSubdito.isSelected()) {
1615            if (!appSub.gui.isVisible())
1616              appSub.gui = new GUI_Subdito(bdSub);
1617            appSub.gui.start();
1618            System.out.println(bdRei.getMensagens().size());
1619            subAvailable.release();
1620            subAvailable.release();
1621            write("Ativei a GUI_SUBDITO \n");
1622          }
```

```
1623          else {
1624            appSub.gui.off();
1625            try {
1626              subAvailable.acquire();
1627            } catch (InterruptedException e1) {
1628              // TODO Auto-generated catch block
1629              e1.printStackTrace();
1630            }
1631            write("Desativei a GUI_SUBDITO \n");
1632          }
1633
1634
1635        }
1636      });
1637
1638        chckbxGravar.addActionListener(new ActionListener()
1639      {
1640        public void actionPerformed(ActionEvent e)
1641        {
1642          if(chckbxGravar.isSelected()) {
1643            if (!appGravar.gui.isVisible())
1644              appGravar.gui = new GUI_Gravar(bdGravar);
1645            //appGravar.gui.start();
1646            gravarAvailable.release();
1647            gravarAvailable.release();
1648            bdGravar.setGravarOff(true);
1649            write("Ativei a GUI_GRAVAR \n");
1650          }
1651          else {
1652            //appGravar.gui.off();
1653            try {
1654              bdGravar.setGravarOff(false);
1655              gravarAvailable.acquire();
1656            } catch (InterruptedException e1) {
1657              // TODO Auto-generated catch block
1658              e1.printStackTrace();
1659            }
1660            write("Desativei a GUI_GRAVAR \n");
1661          }
1662
1663
1664        }
1665      });
1666
1667        addWindowListener(new WindowAdapter(){
1668            public void windowClosing(WindowEvent e){
1669                if (appSub.getBD().isLigado())
1670                {
1671                  appSub.gui.write(" Desconectando o robot ... \n");
1672                  write(" Desconectando o robot ... \n");
1673                  appSub.getBD().getRobot().getRobot().CloseEV3();
1674                  appSub.getBD().setLigado(false);
1675                  System.exit(0);
1676                }
1677                else {
1678                    System.out.println("Closing program");
1679                    System.exit(0);
1680                }
1681            }
1682        });
```

```
1683
1684        setVisible(true);
1685    }
1686
1687
1688 }
1689
1690 Classe GUI_Rei
1691
1692 package ptrabalho;
1693
1694 import java.awt.Color;
1695 import java.awt.EventQueue;
1696 import java.awt.Font;
1697 import java.awt.event.ActionEvent;
1698 import java.awt.event.ActionListener;
1699 import java.util.Random;
1700
1701 import javax.swing.BorderFactory;
1702 import javax.swing.JButton;
1703 import javax.swing.JFrame;
1704 import javax.swing.JPanel;
1705 import javax.swing.WindowConstants;
1706 import javax.swing.border.Border;
1707 import javax.swing.border.EmptyBorder;
1708 import javax.swing.border.LineBorder;
1709 import javax.swing.border.TitledBorder;
1710
1711 public class GUI_Rei extends GUI_BaseRS
1712 {
1713   private int id = 0;
1714   protected JButton btn8com;
1715   protected JButton btn16com;
1716   protected JButton btn1com;
1717   Mensagem msg = null;
1718
1719
1720   public GUI_Rei(BD_Rei bd)
1721   {
1722     super(bd);
1723     EventQueue.invokeLater(new Runnable()
1724     {
1725       public void run()
1726       {
1727         try
1728         {
1729           init_Rei(bd);
1730
1731         } catch (Exception e)
1732         {
1733           e.printStackTrace();
1734         }
1735       }
1736     });
1737   }
1738
1739   /**
1740    * Create the frame.
1741    */
1742   public void init_Rei(BD_Rei bd)
```

```
1743  {
1744
1745    setTitle("Trabalho 1 - GUI Rei");
1746    setDefaultCloseOperation(WindowConstants. DO_NOTHING_ON_CLOSE);
1747    setBounds(1000, 0, 760, 600);
1748
1749    Border cost = BorderFactory.createLineBorder(new Color(0,0,0)
      ,1);
1750    TitledBorder borda_rei = BorderFactory.createTitledBorder(cost,
      "Controle do Robot em Modo Automático");
1751    JPanel panel_1_1 = new JPanel();
1752    panel_1_1.setLayout(null);
1753    panel_1_1.setName("Controle do Robot em Modo Automático");
1754    panel_1_1.setBorder(new LineBorder(new Color(0, 0, 0)));
1755    panel_1_1.setBounds(10, 34, 719, 113);
1756    panel_1_1.setBorder(borda_rei);
1757    getContentPane().add(panel_1_1);
1758
1759    btn8com = new JButton("8 Comandos Aleatórios");
1760    btn8com.setEnabled(false);
1761    btn8com.setFont(new Font("Arial", Font.BOLD, 12));
1762    btn8com.setBounds(10, 78, 300, 25);
1763    panel_1_1.add(btn8com);
1764
1765    btn16com = new JButton("16 Comandos Aleatórios");
1766    btn16com.setEnabled(false);
1767    btn16com.setFont(new Font("Arial", Font.BOLD, 12));
1768    btn16com.setBounds(390, 78, 300, 25);
1769    panel_1_1.add(btn16com);
1770
1771    btn1com = new JButton("1 Comando Aleatório");
1772    btn1com.setFont(new Font("Arial", Font.BOLD, 12));
1773    btn1com.setEnabled(false);
1774    btn1com.setBounds(10, 25, 680, 25);
1775    panel_1_1.add(btn1com);
1776    contentPane = new JPanel();
1777    contentPane.setBorder(new EmptyBorder(100, 100,100, 100));
1778
1779    btn1com.addActionListener(new ActionListener()
1780    {
1781      public void actionPerformed(ActionEvent e)
1782      {
1783        msg = gerarRandomMensagem();
1784        bd.addMensagem(msg);
1785        write(" Criei a mensagem " + msg + "\n");
1786      }
1787    });
1788
1789
1790    btn8com.addActionListener(new ActionListener()
1791    {
1792      public void actionPerformed(ActionEvent e)
1793      {
1794        for (int i = 0; i<8; i++)
1795        {
1796          msg = gerarRandomMensagem();
1797          bd.addMensagem(msg);
1798          write(" Criei a mensagem " + msg + "\n");
1799
1800        }
```

```
1801        }
1802      });
1803
1804      btn16com.addActionListener(new ActionListener()
1805      {
1806        public void actionPerformed(ActionEvent e)
1807        {
1808          for (int i = 0; i<16; i++)
1809          {
1810            msg = gerarRandomMensagem();
1811            bd.addMensagem(msg);
1812            write(" Criei a mensagem " + msg + "\n");
1813          }
1814
1815        }
1816      });
1817
1818
1819      btnFrt.addActionListener(new ActionListener()
1820      {
1821        public void actionPerformed(ActionEvent e)
1822        {
1823          Mensagem mensagem = new Mensagem(id,1,bd.getDist(),0);
1824          bd.addMensagem(mensagem);
1825          write(" Criei a mensagem " + mensagem + "\n");
1826        }
1827      });
1828
1829      btnEsq.addActionListener(new ActionListener()
1830      {
1831        public void actionPerformed(ActionEvent e)
1832        {
1833          Mensagem mensagem = new Mensagem(id,3,bd.getRaio(),bd.
    getAng());
1834          bd.addMensagem(mensagem);
1835          write(" Criei a mensagem " + mensagem + "\n");
1836        }
1837      });
1838
1839      btnDir.addActionListener(new ActionListener()
1840      {
1841        public void actionPerformed(ActionEvent e)
1842        {
1843          Mensagem mensagem = new Mensagem(id,2,bd.getRaio(),bd.
    getAng());
1844          bd.addMensagem(mensagem);
1845          write(" Criei a mensagem " + mensagem + "\n");
1846        }
1847      });
1848
1849      btnTras.addActionListener(new ActionListener()
1850      {
1851        public void actionPerformed(ActionEvent e)
1852        {
1853          Mensagem mensagem = new Mensagem(id,1,-bd.getDist(),0);
1854          bd.addMensagem(mensagem);
1855          write(" Criei a mensagem " + mensagem + "\n");
1856        }
1857      });
1858
```

```
1859      btnParar.addActionListener(new ActionListener()
1860      {
1861        public void actionPerformed(ActionEvent e)
1862        {
1863          Mensagem mensagem = new Mensagem(id,0,0,0);
1864          bd.addMensagem(mensagem);
1865          write(" Criei a mensagem " + mensagem + "\n");
1866        }
1867      });
1868
1869      setVisible(true);
1870    }
1871
1872    private Mensagem gerarRandomMensagem() {
1873      Mensagem m = new Mensagem();
1874      Random rn = new Random();
1875      int[] variaveis = new int[4]; // array de 4 variaveis
1876      int tipoMensagem = rn.nextInt(3); // random between 0-2
1877      if(id==8)
1878        id=0;
1879      variaveis = gerarVariaveis(tipoMensagem);
1880      m.setId(variaveis[0]);
1881      m.setTipo(variaveis[1]);
1882      m.setArg1(variaveis[2]);
1883      m.setArg2(variaveis[3]);
1884      return m;
1885    }
1886
1887    /*
1888     * Metodo auxiliar ao gerarRandomMensagem()
1889     *
1890     * @param tMsg - tipo de mensagem
1891     */
1892    private int[] gerarVariaveis(int tMsg) {
1893      Random rn = new Random();
1894      int[] variaveis = new int[4];
1895      int variavel;
1896      if (tMsg == 0) { // para tMsg 0 faz reta
1897        variaveis[0] = id;
1898        variaveis[1] = 1; // 1 equivale a reta na minha mensagem
1899
1900        variavel = rn.nextInt(45) + 5; // random between 5-50 cm reta
1901        int sinal = rn.nextInt(2);
1902        if (sinal == 1)
1903          variavel*=-1;
1904        variaveis[2] = variavel;
1905        variaveis[3] = 0;
1906      } else if (tMsg == 1) { // para tMsg 1 faz curva direita
1907        variaveis[0] = id;
1908        variaveis[1] = 2; // 2 equivale a reta na minha mensagem
1909        variavel = rn.nextInt(30); // random between 0-30 raio
1910        variaveis[2] = variavel;
1911        variavel = rn.nextInt(70) + 20;// random between 20-90 angulo
1912        variaveis[3] = variavel;
1913      } else { // para tMsg 0 faz curva esquerda
1914        variaveis[0] = id;
1915        variaveis[1] = 3; // 3 equivale a reta na minha mensagem
1916        variavel = rn.nextInt(30); // random between 0-30 raio
1917        variaveis[2] = variavel;
1918        variavel = rn.nextInt(70) + 20;// random between 20-90 angulo
```

```
      variaveis[3] = variavel;
    }
    return variaveis;
  }

  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 GUI_Subdito

```
package ptrabalho;

import java.awt.EventQueue;

import javax.swing.BorderFactory;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;

import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.SwingConstants;
import javax.swing.WindowConstants;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.JTextArea;

public class GUI_Subdito extends GUI_BaseRS
{

  private JPanel contentPane;
  private JTextField txtNome;


  /**
   * Launch the application.
```

```
1979     */
1980    public GUI_Subdito(BD_Subdito bd)
1981    {
1982      super(bd);
1983      EventQueue.invokeLater(new Runnable()
1984      {
1985        public void run()
1986        {
1987          try
1988          {
1989            init_Subdito(bd);
1990
1991          } catch (Exception e)
1992          {
1993            e.printStackTrace();
1994          }
1995        }
1996      });
1997    }
1998
1999    /**
2000     * Create the frame.
2001     */
2002    public void init_Subdito(BD_Subdito bd)
2003    {
2004      setTitle("Trabalho 2 - Subdito");
2005
2006      setDefaultCloseOperation(WindowConstants. DO_NOTHING_ON_CLOSE);
2007        setBounds(50, 400, 760, 600);
2008
2009
2010      JLabel lblNomeDoRobot = new JLabel("Nome do robot");
2011      lblNomeDoRobot.setHorizontalAlignment(SwingConstants.LEFT);
2012      lblNomeDoRobot.setFont(new Font("Arial", Font.BOLD, 12));
2013      lblNomeDoRobot.setBounds(30, 61, 103, 25);
2014      getContentPane().add(lblNomeDoRobot);
2015
2016      JRadioButton rdbtnAbrirFecharBlt = new JRadioButton("Abrir /
     Fechar Bluetooth");
2017      rdbtnAbrirFecharBlt.addActionListener(new ActionListener()
2018      {
2019        public void actionPerformed(ActionEvent e) {
2020          if (bd.isLigado())
2021              {
2022            System.out.println("Desligando...");
2023                  bd.getRobot().getRobot().CloseEV3();
2024                  bd.setLigado(false);
2025                  txtLog.append(" O robot foi desligado com sucesso \
     n");
2026
2027              } else
2028              {
2029                System.out.println("Ligando...");
2030                  bd.setLigado(bd.getRobot().getRobot().OpenEV3(bd.
     getNome()));
2031                  System.out.println(bd.isLigado());
2032                  if (!bd.isLigado()) {
2033                    rdbtnAbrirFecharBlt.setSelected(false);
2034                  }
2035                  else
```

```
2036                    txtLog.append(" O robot foi ligado com sucesso \n
    ");
2037
2038             }
2039       }
2040     });
2041
2042     txtNome = new JTextField();
2043     txtNome.addActionListener(new ActionListener()
2044     {
2045       public void actionPerformed(ActionEvent e)
2046       {
2047         bd.setNome(txtNome.getText());
2048         txtLog.append(" O nome do Robot é " + bd.getNome() + "\n");
2049       }
2050     });
2051
2052
2053     txtNome.setFont(new Font("Arial", Font.BOLD, 12));
2054     txtNome.setColumns(10);
2055     txtNome.setBounds(143, 61, 279, 25);
2056     getContentPane().add(txtNome);
2057
2058
2059     btnFrt.addActionListener(new ActionListener()
2060     {
2061       public void actionPerformed(ActionEvent e)
2062       {
2063         bd.getRobot().reta(new MensagemReta(0,1,bd.getDist()));
2064         bd.getRobot().parar(new MensagemParar(0,4,false));
2065             txtLog.append(" O robot avançou " + bd.getDist() + "\n"
    );
2066       }
2067     });
2068
2069     btnEsq.addActionListener(new ActionListener()
2070     {
2071       public void actionPerformed(ActionEvent e)
2072       {
2073         bd.getRobot().curvarEsquerda(new MensagemCurvar(0,3,bd.
    getRaio(),bd.getAng()));
2074         bd.getRobot().parar(new MensagemParar(0,4,false));
2075             txtLog.append(" O robot virou esquerda com raio = " +
    bd.getRaio() +" e angulo = " + bd.getAng() + "\n");
2076       }
2077     });
2078
2079     btnDir.addActionListener(new ActionListener()
2080     {
2081       public void actionPerformed(ActionEvent e)
2082       {
2083         bd.getRobot().curvarDireita(new MensagemCurvar(0,2,bd.
    getRaio(),bd.getAng()));
2084         bd.getRobot().parar(new MensagemParar(0,4,false));
2085             txtLog.append(" O robot virou direita com raio = " + bd
    .getRaio() +" e angulo = " + bd.getAng() + "\n");
2086       }
2087     });
2088
2089     btnTras.addActionListener(new ActionListener()
```

```
2090        {
2091          public void actionPerformed(ActionEvent e)
2092          {
2093            bd.getRobot().reta(new MensagemReta(0,1,-bd.getDist()));
2094            bd.getRobot().parar(new MensagemParar(0,4,false));
2095                txtLog.append(" O robot recuou " + bd.getDist() + "\n")
    ;
2096          }
2097        });
2098
2099        btnParar.addActionListener(new ActionListener()
2100        {
2101          public void actionPerformed(ActionEvent e)
2102          {
2103            bd.getRobot().parar(new MensagemParar(0,4,false));
2104            //System.out.print(bd.getCanal().GetandSetReadLeitor());
2105            txtLog.append(" O robot parou \n");
2106            //txtLog.append("no limpa " + bd.getCanal().GetandSetRead()
    .toString() +"\n");
2107
2108          }
2109        });
2110
2111
2112        rdbtnAbrirFecharBlt.setFont(new Font("Arial", Font.BOLD, 12));
2113        rdbtnAbrirFecharBlt.setBounds(455, 61, 188, 25);
2114        getContentPane().add(rdbtnAbrirFecharBlt);
2115
2116        setVisible(true);
2117      }
2118
2119  }
2120
2121  Classe iMensagem
2122
2123  package ptrabalho;
2124
2125  public interface iMensagem {
2126
2127        int parar           = 0;
2128        int reta            = 1;
2129        int curvarDir         = 2;
2130        int curvarEsq         = 3;
2131        int vazia              = 4;
2132        int pedir             = 5;
2133        int sensor             = 6;
2134        int workflow         = 7;
2135        int endWorkflow     = 8;
2136  }
2137
2138  Classe Mensagem
2139
2140  package ptrabalho;
2141
2142  public class Mensagem implements iMensagem{
2143        int tipo, id, arg1, arg2;
2144
2145        public Mensagem(int id, int tipo, int arg1, int arg2) {
2146            this.tipo = tipo;
2147            this.id = id;
```

```
2148            this.arg1 = arg1;
2149            this.arg2 = arg2;
2150        }
2151
2152        public Mensagem() {
2153
2154        }
2155
2156
2157        @Override
2158        public String toString() {
2159            return "Mensagem{" +
2160                    " id= " + id +
2161                    " tipo=" + tipo +
2162                    " arg1=" + arg1 +
2163                    " arg2=" + arg2 +
2164                    '}';
2165        }
2166
2167        public int getTipo() {
2168            return tipo;
2169        }
2170
2171        public void setTipo(int tipo) {
2172            this.tipo = tipo;
2173        }
2174
2175        public int getId() {
2176            return id;
2177        }
2178
2179        public void setId(int id) {
2180            this.id = id;
2181        }
2182
2183        public int getArg1() {
2184            return arg1;
2185        }
2186
2187        public void setArg1(int arg) {
2188            this.arg1 = arg;
2189        }
2190
2191        public int getArg2() {
2192            return arg2;
2193        }
2194
2195        public void setArg2(int arg) {
2196            this.arg2 = arg;
2197        }
2198
2199        public boolean equals(Mensagem mensagem) {
2200
2201            if(mensagem==null) return false;
2202
2203            return (this.getId() == mensagem.getId());
2204        }
2205    }
2206
2207    Classe MensagemCurvar
```

```
2208
2209 package ptrabalho;
2210
2211 public class MensagemCurvar extends Mensagem{
2212   int raio, ang;
2213
2214   public MensagemCurvar(int id, int tipo, int raio, int ang) {
2215     super(id, tipo, raio, ang);
2216     this.raio = raio;
2217     this.ang  = ang;
2218   }
2219
2220   public int getRaio() {
2221     return raio;
2222   }
2223
2224   public void setRaio(int raio) {
2225     this.raio = raio;
2226   }
2227
2228   public int getAng() {
2229     return ang;
2230   }
2231
2232   public void setAng(int ang) {
2233     this.ang = ang;
2234   }
2235 }
2236
2237 Classe MensagemParar
2238
2239 package ptrabalho;
2240
2241 public class MensagemParar extends Mensagem{
2242
2243   boolean sincrono;
2244
2245   public MensagemParar(int id, int tipo, boolean sincrono) {
2246     super(id, tipo, 0, 0);
2247     this.sincrono = sincrono;
2248   }
2249
2250   @Override
2251   public String toString() {
2252     return super.toString() +" " +
2253         "sincrono=" + sincrono +
2254         '}';
2255   }
2256
2257   public boolean isSincrono() {
2258     return sincrono;
2259   }
2260
2261   public void setSincrono(boolean sincrono) {
2262     this.sincrono = sincrono;
2263   }
2264
2265 }
2266
2267 Classe MensagemReta
```

```
2268
2269 package ptrabalho;
2270
2271 public class MensagemReta extends Mensagem{
2272   int dist;
2273
2274   public MensagemReta(int id, int tipo, int dist) {
2275     super(id, tipo, dist, 0);
2276     this.dist = dist;
2277   }
2278
2279   public MensagemReta(int id, int tipo, int dist, int arg2) {
2280     super(id, tipo, dist, 0);
2281     this.dist = dist;
2282   }
2283
2284   public int getDist() {
2285     return dist;
2286   }
2287
2288   public void setDist(int dist) {
2289     this.dist = dist;
2290   }
2291
2292   @Override
2293   public String toString() {
2294     return super.toString() + " distancia= " + dist + "}";
2295   }
2296 }
2297
2298 Classe MensagemVazia
2299
2300 package ptrabalho;
2301
2302 public class MensagemVazia extends Mensagem{
2303     public MensagemVazia(int id, int tipo) {
2304         super(id, tipo, 0, 0);
2305     }
2306 }
2307
2308 Classe myRobotLegoEV3
2309
2310 package ptrabalho;
2311
2312 import robot.RobotLegoEV3;
2313
2314 public class myRobotLegoEV3 extends RobotEV3{
2315
2316   private RobotLegoEV3 robot;
2317
2318   public myRobotLegoEV3(Gravador g) {
2319     super(g);
2320     robot = new RobotLegoEV3();
2321   }
2322
2323
2324   public RobotLegoEV3 getRobot() {
2325     return robot;
2326   }
2327
```

```
2328
2329    synchronized void reta(Mensagem msg)
2330    {
2331      super.reta(msg);
2332      robot.Reta(msg.getArg1());
2333    }
2334
2335    synchronized void curvarDireita(Mensagem msg)
2336    {
2337      super.curvarDireita(msg);
2338      robot.CurvarDireita(msg.getArg1(),msg.getArg2());
2339    }
2340
2341    synchronized void curvarEsquerda(Mensagem msg)
2342    {
2343      super.curvarEsquerda(msg);
2344      robot.CurvarEsquerda(msg.getArg1(),msg.getArg2());
2345
2346    }
2347
2348    synchronized void parar(Mensagem msg)
2349    {
2350      super.parar(msg);
2351      robot.Parar(false);
2352    }
2353  }
2354
2355  Classe RobotEV3
2356
2357  package ptrabalho;
2358
2359  public class RobotEV3 {
2360
2361    protected Gravador gravador;
2362
2363    public RobotEV3(Gravador g) {
2364      gravador = g;
2365    }
2366
2367
2368    synchronized void reta(Mensagem msg) {
2369      //System.out.println(gravador.getGravar());
2370      if (gravador!=null && gravador.getGravar() && gravador.bdGravar
       .getGravarOff())
2371        gravador.reta(msg);
2372      //System.currentTimeMillis();
2373    }
2374
2375    synchronized void curvarEsquerda(Mensagem msg) {
2376      //System.out.println(gravador.getGravar());
2377      if (gravador!=null && gravador.getGravar() && gravador.bdGravar
       .getGravarOff())
2378        gravador.curvarEsquerda(msg);
2379      //System.currentTimeMillis();
2380    }
2381
2382    synchronized void curvarDireita(Mensagem msg) {
2383      //System.out.println(gravador.getGravar());
2384      if (gravador!=null && gravador.getGravar() && gravador.bdGravar
       .getGravarOff())
```

```
2385        gravador.curvarDireita(msg);
2386      //System.currentTimeMillis();
2387    }
2388
2389    synchronized void parar(Mensagem msg) {
2390      //System.out.println(gravador.getGravar());
2391      if (gravador!=null && gravador.getGravar() && gravador.bdGravar
          .getGravarOff())
2392        gravador.parar(msg);
2393      //System.currentTimeMillis();
2394    }
2395
2396  }
```