
Licenciatura Engenharia Informática e Multimédia
Instituto Superior de Engenharia de Lisboa
Ano letivo 2023/2024

**Fundamentos de Sistemas Operativos**

Relatório: Trabalho Prático 1 - Aula 7

Turma: 31D — Grupo: 4

Nome: João Ramos — Número: 50730

Nome: Miguel Alcobia — Número: 50746

Nome: Fábio Pestana — Número: 50756

Professor: Jorge Pais

Data: 31 de Outubro de 2023

# Índice

# Lista de figuras

# Lista de tabelas

# Abreviaturas e símbolos

Lista de abreviaturas

GUI Graphical User Interface
API Application Programming Interface

# Objetivos

Neste trabalho, os alunos terão como objetivo desenvolver uma aplicação multiprocesso composta por dois processo em Java para o funcionamento do jogo "O Rei manda". Sendo que um dos processos implementa o comportamento do Rei, o outro implementa o do Súbdito que conhece a API do Robot.

A comunicação entre os processos do Rei e do Súbdito é efetuada através de memória partilhada suportada pela classe MappedByteBuffer.

A avaliação do trabalho está repartida ao longo das sete aulas em que o trabalho deverá ser realizado. Os objetivos aos longo das aulas foram os seguintes:

- Aula prática 1 – Instalação do WindowBuilder sobre o Eclipse. Desenho duma GUI com os componentes comuns aos processos Súbdito e Rei;
- Aula 2 – Instalação das bibliotecas do robot. Implementação da GUI do processo Súbdito e tratamento dos eventos para controlo do robot. Implementação da GUI do processo Rei.
- Aula 3 – Desenho do diagrama de classes utilizando o MagicDraw do processo Rei e do processo Súbdito incluindo as classes sugeridas para comunicação Mensagem e CanalDeComunicacao;
- Aula 4 – Desenvolvimento das classes Mensagem e CanalDeComunicacao e respetivo teste;
- Aula 5 – Desenho do diagrama de atividades, implementação e teste do processo Rei;
- Aula 6 – Desenho do diagrama de atividades, implementação e teste do processo Súbdito;
- Aula 7 – Integração final e validação de toda a aplicação multiprocesso.

*(retirado do enunciado do trabalho)*

# Aula 1 - Desenho duma GUI com os componentes comuns aos processos Súbdito e Rei

## 1. 1 - Elaboração da GUI Genérica

Nas interfaces sugeridas no enunciado (Figura 1 e 2), ao constatar-se que as GUI's do Rei e do Súbdito tinham alguns elementos em comum, optou-se por desenvolver-se uma GUI genérica da qual as outras duas derivariam.

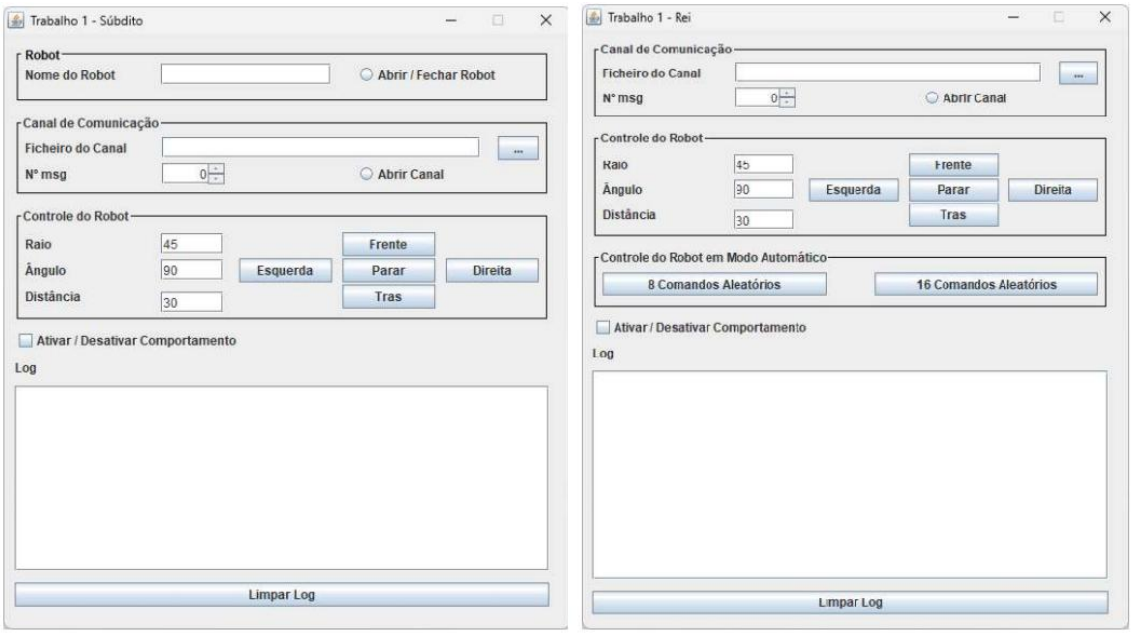


**Figura 1 - GUI sugerida para o processo do Súbdito e do Rei**

Com o auxílio do editor gráfico `WindowBuilder` no Eclipse, desenhou-se a seguinte janela:

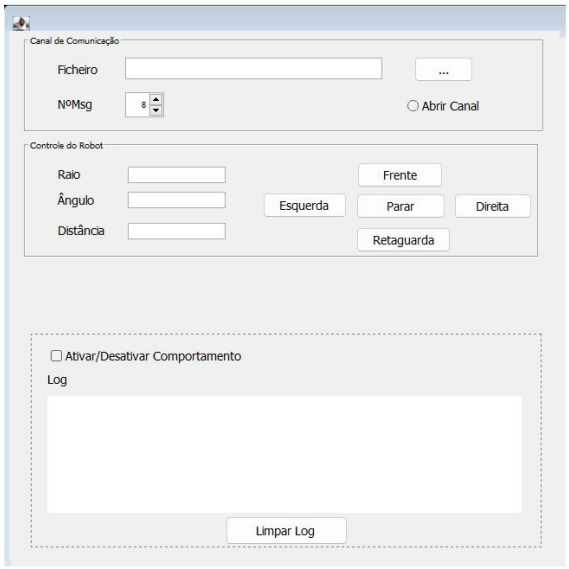


**Figura 2 - Resultado do desenho da GUI Genérica**

O espaço em branco da GUI Genérica destina-se aos elementos próprios de cada processo. No caso do Rei será acrescentado a secção do "Controle do Robot em Modo Automático" e, no caso do Súbdito, será acrescentado a secção "Robot".

## 1.2 - Explicação dos vários campos

### Canal de Comunicação:

Temos um `TextField` e um `Button` para escolher qual o ficheiro que servirá o *File Channel*. O `spinner` serve para assinalar o número de mensagens que serão estabelecidas entre o Rei e Súbdito para envio e leitura. O `RadioButton` serve, como está assinalado, para abrir ou fechar o Canal de Comunicação.

### Controle do Robot:

Temos um `TextField` onde estão os vários argumentos que serão necessários para os vários comandos. Cada `Button` representa uma direção ou um comando como é o caso de "Parar".

### Log:

Temos um `TextArea` para uma zona de *logging* onde fica registados os vários comandos e as mensagens que serão enviadas pelo Rei ou memorizadas pelo Súbdito. O `Button` "Limpar Log", serve para apagar toda a informação da zona de *logging.* A *checkbox* "Ativar/Desativar Comportamento" tem a função de ativar ou interromper o comportamento atribuído à janela onde se encontra.

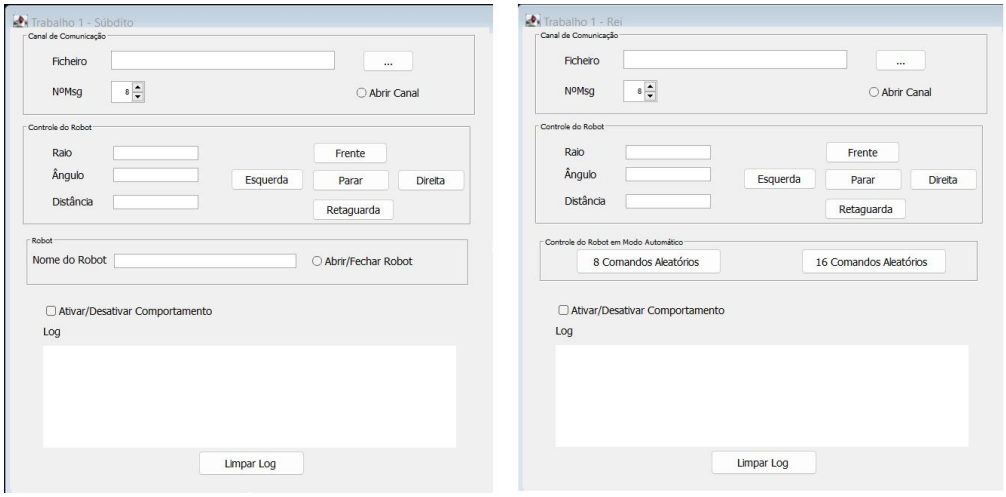


**Figura 3 - GUI dos processos Súbdito e Rei, resultantes da Aula 1**

# Aula 2 - A implementação da GUI do processo Rei e do Súbdito e tratamento dos eventos para controlo do robot

## 1. 1 - Implementação da GUI do processo Rei e do Súbdito

Continuando as interfaces realizadas na primeira aula, colocou-se os campos próprios de cada processo. O processo do **Rei** tem dois botões (tipo `Button`): um para enviar 8 comandos aleatórios e outro para enviar 16. Por sua vez, o do **Súbdito** tem um campo `TextField` para escrever o nome do Robot e um `RadioButton` para abrir e fechar o canal de comunicação com o robô.

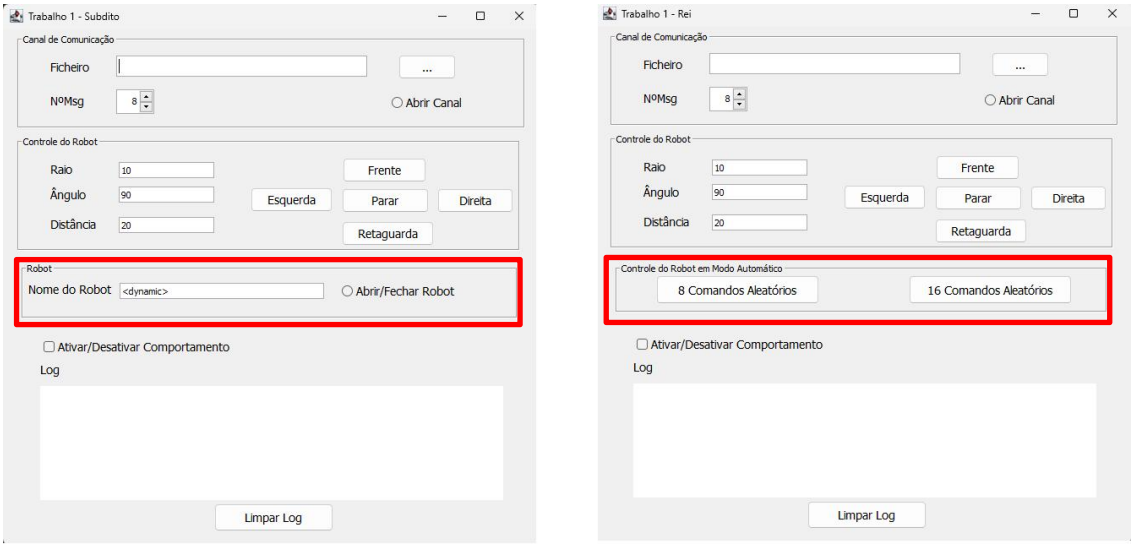


**Figura 4 - GUI's com os campos dos respetivos processos destacados**

Também foi criado um `actionPerformed` na área para atribuir um nome ao robot, que chama o método `setNomeRobot()` da classe `Dados`, como mostra o seguinte pedaço de código:

```
nameTextField = new JTextField(dados.getNomeRobot());
nameTextField.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                dados.setNomeRobot(nameTextField.getText());
        }
});
```

Para o funcionamento do `RadioButton` da GUI do Súbdito, verifica-se se a variável `onOff` (que informa se foi efetuada alguma ligação) está a `false`. Se estiver, verifica-se se o robot está conectado ao computador. Se a conexão existir, a `RadioButton` é selecionada (pintada) e a variável onOff passa a `true`, como está no demonstrado no excerto abaixo:

```
abrirFecharRobotRadBtn = new JRadioButton("Abrir/Fechar Robot");
abrirFecharRobotRadBtn.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
if(!dados.isOnOff()) {
if(dados.getRobot().OpenEV3(dados.getNomeRobot())) {
abrirFecharRobotRadBtn.setSelected(true);
dados.setOnOff(true);
}
else {
abrirFecharRobotRadBtn.setSelected(false);
}
}else {
abrirFecharRobotRadBtn.setSelected(false);
dados.getRobot().CloseEV3();
dados.setOnOff(false);
}
}
});
abrirFecharRobotRadBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
abrirFecharRobotRadBtn.setBounds(377, 23, 143, 21);
Robot_panel.add(abrirFecharRobotRadBtn);
```

## 1. 2 - Tratamento dos eventos para controlo do robot

A tarefa que se seguia era preparar os cinco botões de controle da GUI do Súbdito para responderem aos devidos comandos. Foi criado um `actionPerformed` para cada um deles e atribuído o código correspondente ao movimento desejado. A título de exemplo, o próximo pedaço de código mostra a mecânica do botão do movimento de ir para a frente:

```
FrenteButton = new JButton("Frente");
FrenteButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
        LogtextArea.append("Reta("+ dados.getDistancia()+") \nParar(false) \n");
        dados.getRobot().Reta(dados.getDistancia());
        dados.getRobot().Parar(false);
        }
});
FrenteButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
FrenteButton.setBounds(387, 26, 99, 29);
controlPanel.add(FrenteButton);
```

Ao clicarmos no botão “Frente”, surgirá na `LogtextArea` as linhas respetivas do comando dado: `Reta(`distância desejada`)` e `Parar(false)` para o robot parar depois de percorrer a distância determinada pelo utilizador. Durante esta ação as instruções são passadas ao robot pelo método `getRobot()` da classe `Dados`.

Para as `TextFields` do Raio, da Distância e do Ângulo procedeu-se sempre da mesma forma. Segue o exemplo da `TextField` da Distância:

```
DisttextField = new JTextField("" + dados.getDistancia());
DisttextField.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                try {
                        dados.setDistancia(Integer.parseInt(DisttextField.getText()));
                }catch(NumberFormatException nfe) {
                        DisttextField.setText("" + dados.getDistancia());
                }
        }
});
DisttextField.setColumns(10);
DisttextField.setBounds(122, 96, 113, 19);
controlPanel.add(DisttextField);
```

O número inserido pelo utilizador é assumido como novo valor para a distância. Caso haja algum erro, como enviar uma palavra em vez de um número, o valor assumido é o último valor válido enviado. Caso ocorra um erro logo no começo, o valor pré-definido é 20, no caso da distância.

# Aula 3 - Desenho do diagrama de classes utilizando o MagicDraw do processo Rei e do processo Súbdito

## 1. 1 - Desenho dos diagramas

Para a realização dos diagramas, elaboram-se mais classes do que aquelas já trabalhadas nas aulas anteriores. Criou-se uma classe **Rei** e outra classe **Súbdito** que terão os comportamentos que lhe são devidos, deixando as classes das respetivas GUIs limitadas à interface.

O resultado dos diagramas foi o seguinte:

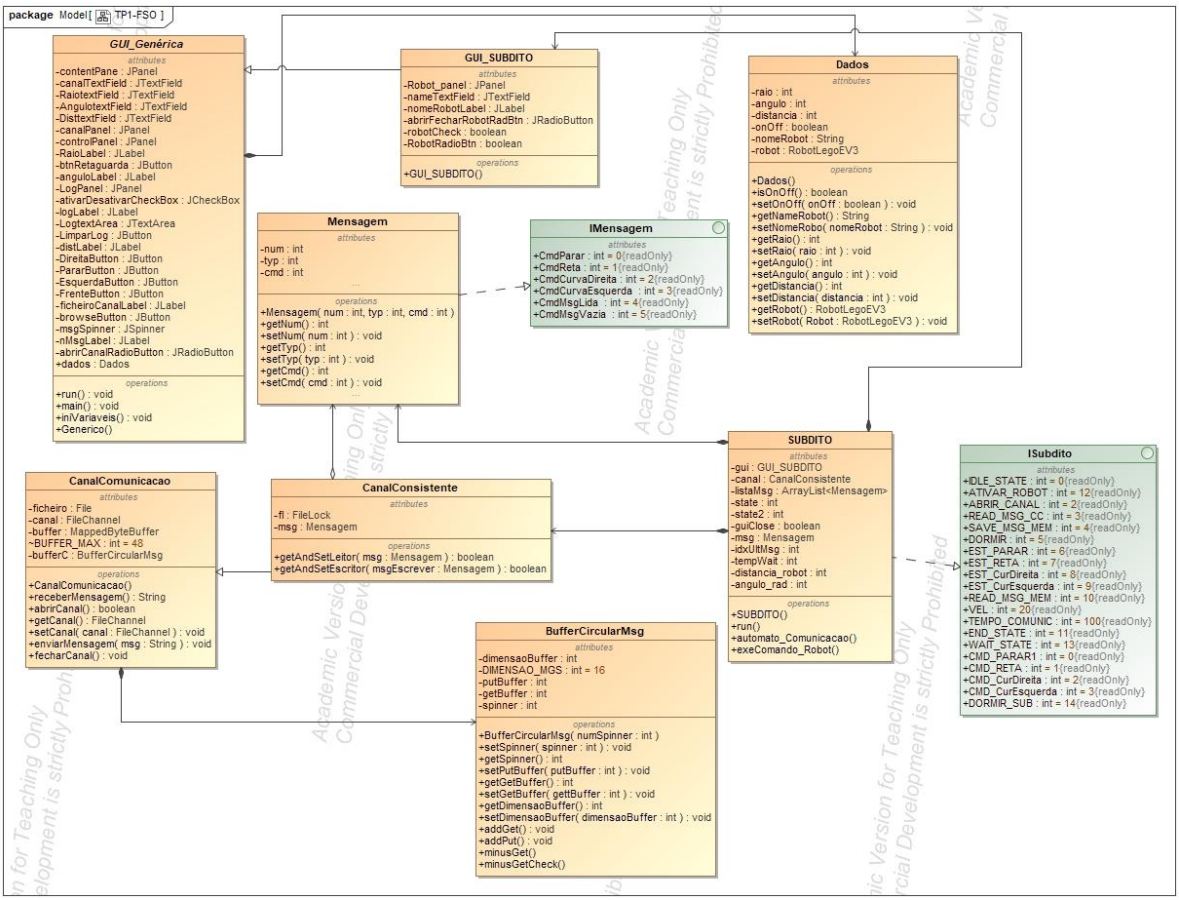


**Figura 5 - Diagrama de classes do Súbdito**

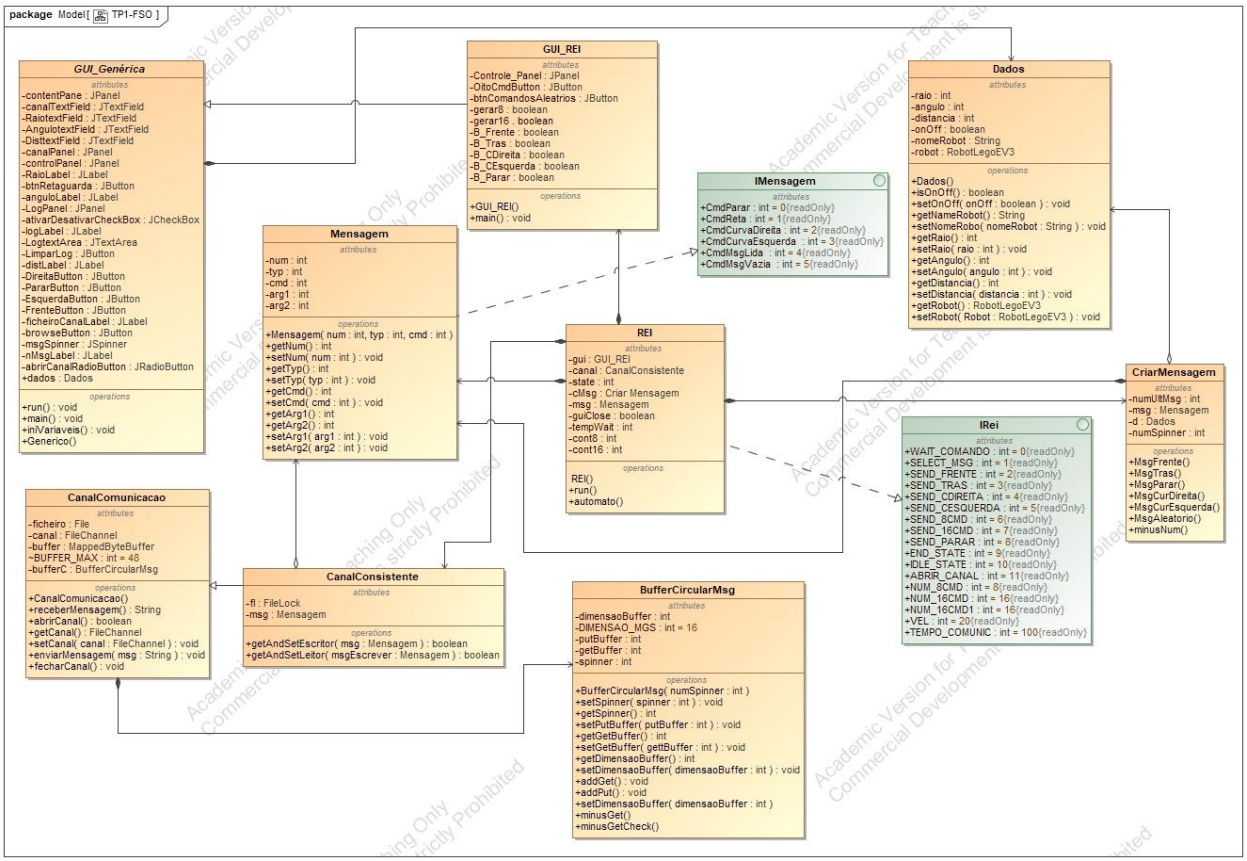


**Figura 6 - Diagrama de classes do Rei**

A `GUI_SUBDITO` é herdada da classe `GUI_Genérica` sendo que esta compõe a classe `Dados`. A classe `SUBDITO` compõe a classe da sua GUI, da `Mensagem` e do Canal de Comunicação Consistente (`CanalConsistente`).

A classe do Canal de Comunicação Consistente (`CanalConsistente`) é herdada da classe `CanalComunicacao`. O Canal Consistente pode ser composto por mensagens (tipo `Mensagem`).

Como referido anteriormente, o diagrama do **Rei** segue uma lógica parecida ao diagrama do **Súbdito**, com a diferença de que a classe CriarMensagem não existe no diagrama do Rei. Esta classe serve para facilitar a criação de mensagens dependendo do comando pretendido pelo Rei. CriarMensagem é uma classe que compõe Mensagens e pode ser composto por dados (tipo Dados).

As classes `Mensagem`, `Rei` e `Subdito` implementam interfaces: `IMensagem`, `IRei` e `ISubdito`, respetivamente. Estas interfaces, são usadas para trabalhar com valores constantes.

# Aula 4 - Desenvolvimento das classes Mensagem e CanalComunicacao e respetivo teste

## 1. 1 - Desenvolvimento da classe Mensagem

A classe Mensagem só tem o seu construtor e os *gets* e *sets* dos seus atributos. Esta classe implementa a interface `IMensagem` que associa o número do comando ao seu significado, dando uma maior clareza ao código.

As mensagens que têm a seguinte estrutura:

| Número : num (int) | Comando: cmd (int) | Argumento 1: arg1(int) | Argumento 2: arg2(int) |
|---|---|---|---|

**Tabela 1 - Estrutura da Mensagem**

O número (num) identifica a cardinalidade da mensagem enviada e os argumentos (arg1 e arg2) são os argumentos para os comandos do robot.

O significado de cada comando (cmd) é apresentado na Tabela 2:

| Comandos (cmd) | Ordem |
|---|---|
| 0 | Parar |
| 1 | Reta |
| 2 | Curvar à direita |
| 3 | Curvar à esquerda |
| 4 | Aviso de mensagem lida |
| 5 | Aviso de canal vazio |

**Tabela 2 - Significado do valor comando**

## 1. 2 - Desenvolvimento da classe CanalComunicacao e CanalConsistente

Começou-se por elaborar a classe CanalComunicacao, com o seu construtor, *gets* e *sets* e os seguintes métodos:

- `abrirCanal()` - Tal como o nome indica serve para abrir o canal.
- `fecharCanal()` - Método para fechar o canal.
- `receberMensagem()` - Método do tipo Mensagem que vai buscar os vários bytes que compõe a Mensagem ao *buffer*.

- `enviarMensagem()` - Este método coloca no *buffer* os bytes que formam a mensagem.
- `print()` - Método para dar print dos valores passados na Mensagem.

(Os primeiros dois métodos acima foram fornecidos pelo professor em aula)

Ao abrir o canal é enviada uma mensagem com comando (cmd) com valor 5, para avisar que o canal está vazio.

Criou-se um `bufferC` no `CanalComunicacao`, um objeto de uma nova classe: `BufferCircularMsg`, onde além do construtor e dos *gets* e *sets* existem mais dois métodos:

- `addGet()` - Incrementa a posição de leitura do buffer circular. Depois deste método ser chamado, o buffer circular aponta para a próxima posição a ler.
- `addPut()` - Incrementa a posição de escrita do buffer circular. Depois deste método ser chamado, o buffer circular aponta para a posição onde a próxima mensagem será escrita.
- `minusGet()` - Decrementa a posição do índice "getBuffer" no buffer.
- `minusGetCheck()` - Decrementa a posição do índice "getBufferCheck" no buffer.

No `CanalConsistente` além do construtor e dos *gets* e *sets,* temos dois métodos que são os principais da classe, que servem para aceder ao canal de comunicação em exclusão mútua :

- `getAndSetLeitor()` - Este método recebe uma mensagem do tipo `Mensagem`, acede ao canal e lê a mensagem que lá está no caso do canal não estar vazio e se a mensagem recebida tiver um comando diferente do comando lido (cmd=4). Se isto se verificar altera-se os valores da estrutura da mensagem recebida no método e é enviada uma mensagem que assinala a realização e o sucesso da leitura. No final, desimpede-se o canal.
- `getAndSetEscritor()` - Este método também recebe uma mensagem do tipo `Mensagem`, acede ao canal e verifica se a mensagem que lá está assinala o sucesso de uma leitura ou se o canal está vazio. Se alguma destas situações se verificar, a mensagem recebida no método é enviada. No final, desimpede-se o canal.

Também houve alterações nas GUIs, colocando o botão da escolha do ficheiro do canal de comunicação a abrir uma `JFileChooser` para selecionar-se o ficheiro desejado.

# Aula 5 - Desenho do diagrama de atividades, implementação e teste do processo Rei

## 1. 1 - Desenho do diagrama de atividades

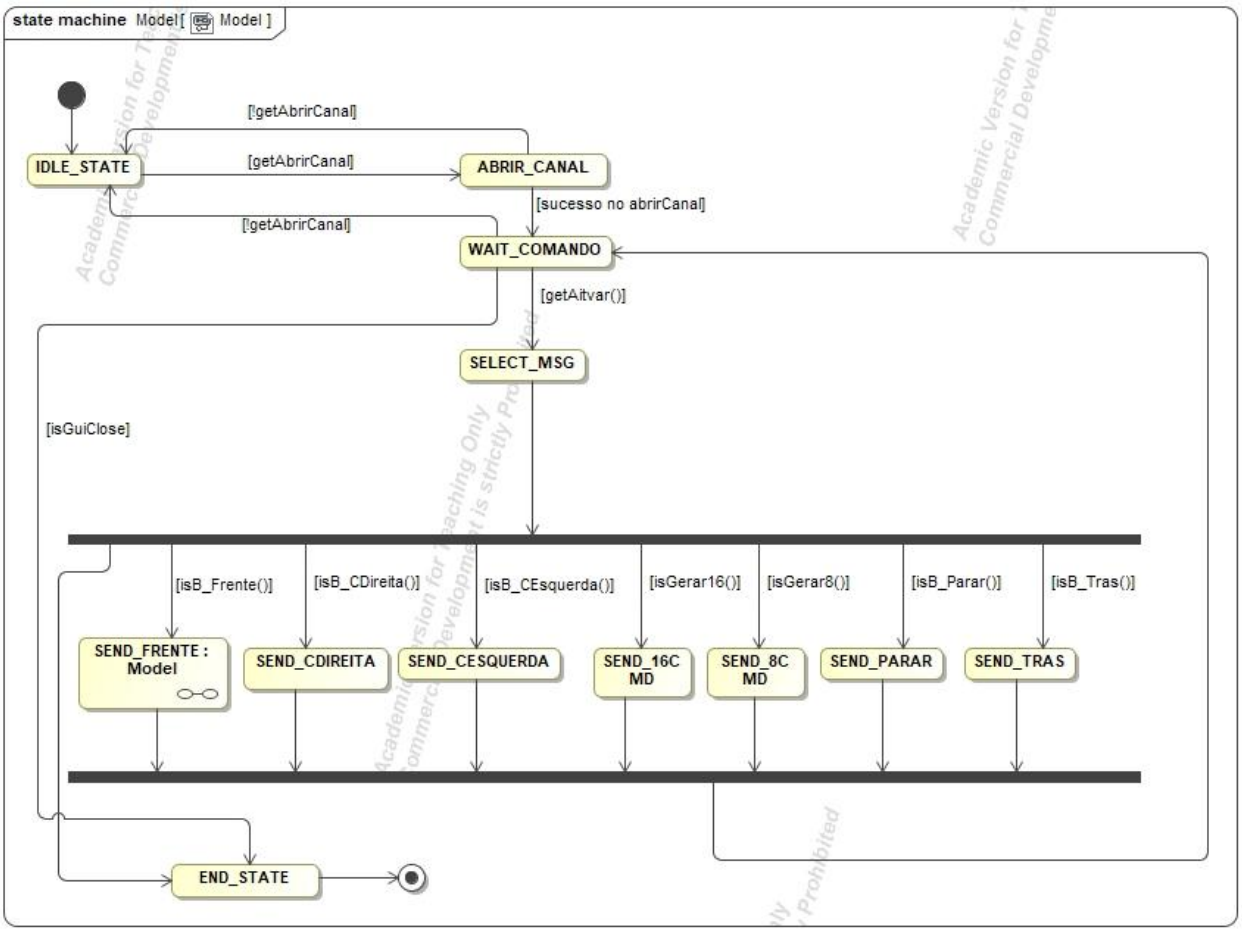


**Figura 7 - Diagrama de Atividades do Rei**

Passando à explicação do diagrama: Temos um estado inicial IDLE_STATE que serve para criar o canal consistente e um objeto do tipo CriarMensagem, uma nova classe que será explicada na parte da implementação. Assim que estes dois processos estejam concluídos, passamos para o estado ABRIR_CANAL, que abre um canal tendo por base um ficheiro que o utilizador cria onde quiser através do JFileChooser da GUI. Logo a seguir passamos para o WAIT_COMANDO que espera a checkbox de "Ativar/Desativar comportamento" para ir para o estado SELECT_MSG. Se a janela da GUI for fechada passamos para o estado END_STATE; por outro lado, se o canal for desativado passamos para o estado IDLE_STATE.

No SELECT_MSG consoante o botoão pressionado na GUI passamos para o respetivo estado. Por exemplo, se clicarmos no botão de "Frente" , passamos para o estado SEND_FRENTE.

No END_STATE ocorre o encerramento do canal através do método fecharCanal() da classe CanalComunicacao.

## 1. 2 - Implementação do processo Rei.

A máquina de estados representada na Figura 1, foi implementada através de um switch case, onde cada case correspondia a um dos estados do diagrama.

Criou-se uma nova classe chamada CriarMensagem, que além do seu construtor e dos *gets* e *sets* tem um método que corresponde a cada um dos comandos associados aos botões da GUI :

- `MsgFrente()`
- `MsgTras()`
- `MsgParar()`
- `MsgCurDireita()`
- `MsgCurEsquerda()`
- `MsgAleatorio()`

Por exemplo, se clicarmos no botão "Frente" da GUI, é invocado o método `MsgFrente()` para criar uma mensagem com o comando desejado.

No `MsgAletorio()` foi usado o método `Math.random()` para criar os números aleatórios nos intervalos desejados. Os estados SEND_16CMD e SEND_8CMD têm um contador para invocar o método `MsgAletorio()` 16 e 8 vezes, respetivamente (só para se o valor do contador atingir o valor 8 ou 16).

# Aula 6 - Desenho do diagrama de atividades, implementação e teste do processo Súbdito

## 1. 1 - Desenho do diagrama de atividades

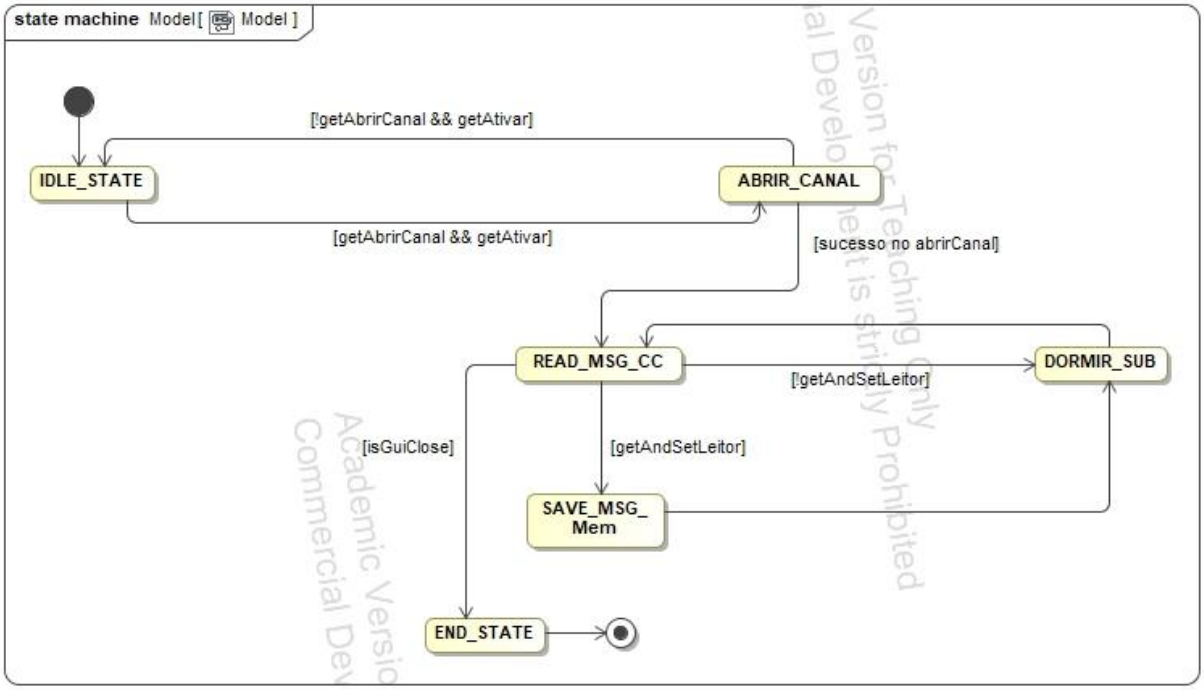


**Figura 8 - Diagrama de Atividades do 1º Autómato do Súbdito**

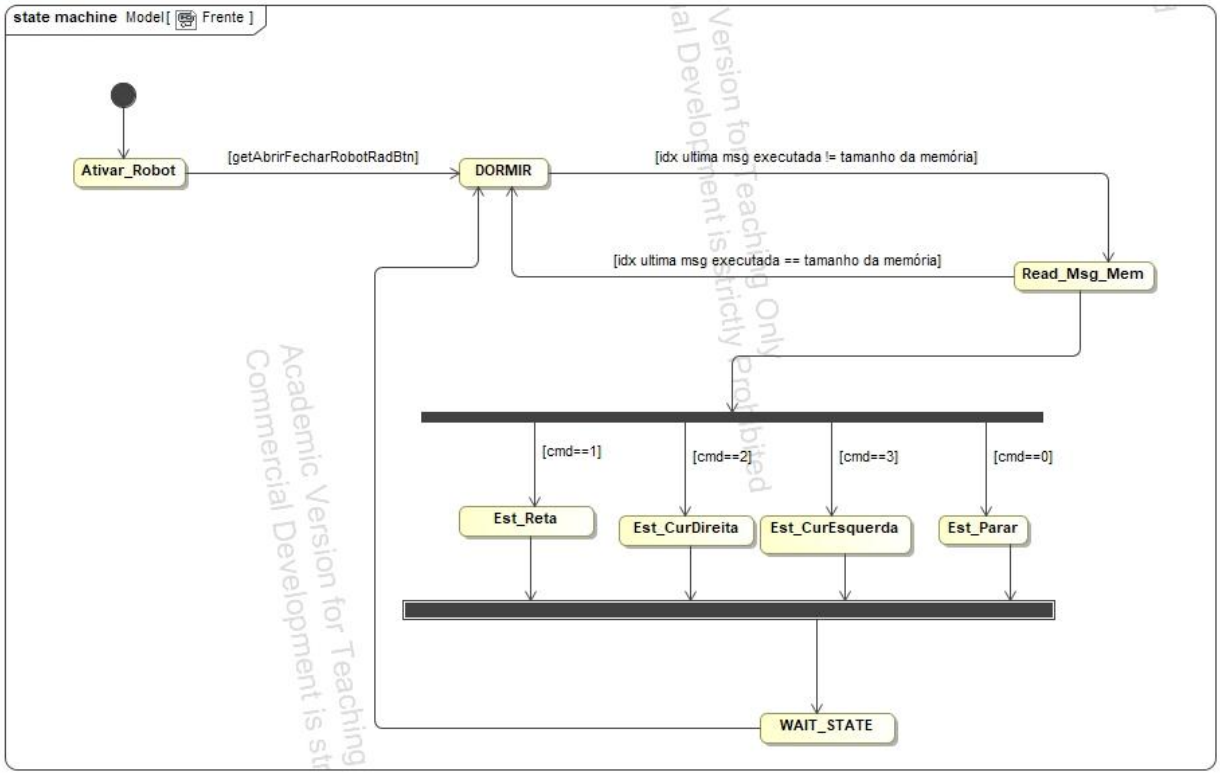


**Figura 9 - Diagrama de Atividades do 2º Autómato do Súbdito**

Passando à explicação do primeiro diagrama: Temos um estado inicial `IDLE_STATE` que serve para criar o canal consistente. Assim que este processo esteja concluído, passamos para o estado `ABRIR_CANAL`, que abre um canal tendo por base um ficheiro que o utilizador cria onde quiser através do `JFileChooser` da GUI. Logo a seguir passamos para o `READ_MSG_CC` que espera a leitura de uma mensagem ser realizada para ir para o estado `SAVE_MSG_MEM`, caso contrário passa para o estado `DORMIR_SUB` para ser submetido a um tempo de espera. Se a janela da GUI for fechada, passamos para o estado `END_STATE`.

No `SAVE_MSG_MEM` a mensagem é gravada na memória (`ArrayList`) e passamos para o estado `DORMIR_SUB`.

No `END_STATE` ocorre o encerramento do canal através do método `fecharCanal()` da classe `CanalComunicacao`.

No segundo diagrama, começamos no estado `ATIVAR_ROBOT`, que verifica se o `RadioButton RobotRadioBtn`, que tem implementado a parte de efetuar a ligação com o robot, foi selecionado. Passamos para o estado `DORMIR` que só muda de estado caso o índice da última mensagem e o tamanho da memória sejam diferentes, mudando para o estado `READ_MSG_MEM`.

No `READ_MSG_MEM`, segue-se para o estado `DORMIR`, caso o índice da última mensagem e o tamanho da memória sejam iguais. Se não, é verificado o digito referente ao comando da mensagem (`cmd`) para assim o estado ser atualizado para o estado correspondente. Por exemplo, se o comando for o "Parar" (`cmd = 0`), o estado é atualizado para o estado `EST_PARAR`.

Em cada estado `EST_` *`(COMANDO)`* é realizada a execução do comando recebido, enviando-a ao robot e exibindo-a na área de Log. Nestes estados também atualizamos o índice da última mensagem e as variáveis que são necessárias para calcular os tempos de espera desejados e, por fim, o estado é atualizado para o estado `WAIT_STATE`.

No estado `WAIT_STATE`, é calculado o tempo de espera desejado e espera-se esse tempo antes de voltarmos para o estado `DORMIR`.

### 1. 2 - Implementação do processo Súbdito.

As máquinas de estados representadas nas Figuras 1 e 2, foram implementadas através de um *switch case* (um para cada autómato), onde cada *case* corresponde a um dos estados dos respetivos diagramas.

O primeiro autómato foi nomeado `automato_Comunicacao()` e o segundo ficou com o nome `exeComando_Robot()`.

Para os tempos de espera foi usado o método `Thread.sleep()`.

# Aula 7 - Integração final

## 1. 1 - Integração final (Problemas e Soluções)

Na última aula testou-se a aplicação e corrigimos os problemas que foram surgindo, assim como se efetuaram melhorias sugeridas pelo professor. Por exemplo, os números das mensagens na área de Log estavam consoante a sua posição no *buffer*, voltando para 1 quando atingiam o máximo. O professor sugeriu os números das mensagens incrementar, independentemente da posição do *buffer*. Por exemplo. Dois cliques nos "16 comandos aleatórios", faria com as mensagens fossem de 1 a 32.

Os ciclos *for* existentes no envio das mensagens aleatórias também foram alterados, devido a serem bloqueantes e não permitirem a interrupção das 8 ou 16 mensagens, sendo substituídos por um contador.

Todos os procedimentos foram cumpridos com sucesso, exceto o teste de controlar dois robots, pois tínhamos uma instância do tipo Robot a ser criado de forma errada, criando Robots não desejados.

# Conclusões

Neste trabalho, os alunos conseguir desenvolver a aplicação multiprocesso composta por dois processo em Java para o funcionamento do jogo "O Rei manda". Sendo que devido a problemas com o bluetooth nos computadores, não conseguiram mostrar a aplicação a funcionar com dois *robots*, durante o tempo de aula.

Durante as aulas práticas percebeu-se a importância da elaboração de um bom diagrama de classes e de um diagrama de atividades, pois ajudam no desenvolvimento e na compreensão do problema quando a complexidade aumenta.

# Bibliografia

Consulta:

Mooddle:

Pais, Jorge. (2023 - 2024). *Fundamentos de Sistemas Operativos*

# Código

## Aula 7 - Integração final:

### Código GUI Genérica:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

// GUI Genérica

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.border.TitledBorder;
import javax.swing.JSpinner;
import javax.swing.JRadioButton;
import javax.swing.SpinnerNumberModel;
import javax.swing.border.EtchedBorder;
import java.awt.Color;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JScrollBar;

public class Generico extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField canalTextField;
    private JTextField RaiotextField;
    private JTextField AngulotextField;//
    private JTextField DisttextField;
    private JPanel canalPanel;
    private JPanel controlPanel;
    private JLabel RaioLabel;
    private JButton btnRetaguarda;
    private JLabel anguloLabel;
    private JPanel LogPanel;
    private JCheckBox ativarDesativarCheckBox;
    private JLabel logLabel;
    private JTextArea LogtextArea;
    private JButton LimparLog;
    private JLabel distLabel;
    private JButton DireitaButton;
    private JButton PararButton;
    private JButton EsquerdaButton;
    private JButton FrenteButton;
    private JLabel ficheiroCanalLabel;
    private JButton browseButton;
    private JSpinner msgSpinner;
    private JLabel nMsgLabel;
    private JRadioButton abrirCanalRadioButton;

    //minhas variavéis

    Dados dados;
    private boolean radioB;
    private String filePath;
    private JScrollPane scrollPane;

    public void iniVariaveis() {
```

```
        dados = new Dados();
        radioB = false;
        filePath = null;
    }

    public String getFilePath() {
        return filePath;
    }
    public void setFilePath(String filePath) {
        this.filePath = filePath;
    }
    public boolean isRadioB() {
        return radioB;
    }
    public void setRadioB(boolean radioB) {
        this.radioB = radioB;
    }

    public boolean getAtivar() {
        return ativarDesativarCheckBox.isSelected();
    }

    public boolean getAbrirCanal() {
        return abrirCanalRadioButton.isSelected();
    }

    public int spinnerValue() {
        return (Integer) msgSpinner.getValue();
    }


    public Generico() {

        iniVariaveis();

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 650, 649);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        canalPanel = new JPanel();
        canalPanel.setBounds(15, 5, 606, 115);
        canalPanel.setBorder(new TitledBorder(null, "Canal de Comunica\u00E7\u00E3o",
TitledBorder.LEADING, TitledBorder.TOP, null, null));
        controlPanel = new JPanel();
        controlPanel.setBounds(15, 126, 600, 139);
        controlPanel.setLayout(null);
        controlPanel.setBorder(new TitledBorder(new EtchedBorder(EtchedBorder.LOWERED, new
Color(255, 255, 255), new Color(160, 160, 160)), "Controle do Robot", TitledBorder.LEADING,
TitledBorder.TOP, null, new Color(0, 0, 0)));

        RaioLabel = new JLabel("Raio");
        RaioLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        RaioLabel.setBounds(41, 29, 54, 20);
        controlPanel.add(RaioLabel);

        RaiotextField = new JTextField("" + dados.getRaio());
        RaiotextField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                try {
                    dados.setRaio(Integer.parseInt(RaiotextField.getText()));
                }catch(NumberFormatException nfe) {
                    RaiotextField.setText("" + dados.getRaio());
                }
            }
        });
        RaiotextField.setColumns(10);
        RaiotextField.setBounds(122, 31, 113, 19);
        controlPanel.add(RaiotextField);

        btnRetaguarda = new JButton("Retaguarda");
        btnRetaguarda.setFont(new Font("Tahoma", Font.PLAIN, 14));
        btnRetaguarda.setBounds(386, 101, 109, 29);
        controlPanel.add(btnRetaguarda);

        anguloLabel = new JLabel("Ângulo");
        anguloLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        anguloLabel.setBounds(41, 59, 54, 20);
```

```
controlPanel.add(anguloLabel);
LogPanel = new JPanel();
LogPanel.setBounds(25, 350, 590, 247);
LogPanel.setLayout(null);

ativarDesativarCheckBox = new JCheckBox("Ativar/Desativar Comportamento");
ativarDesativarCheckBox.setFont(new Font("Tahoma", Font.PLAIN, 14));
ativarDesativarCheckBox.setBounds(19, 14, 251, 22);

LogPanel.add(ativarDesativarCheckBox);
logLabel = new JLabel("Log");
logLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
logLabel.setBounds(19, 42, 54, 20);
LogPanel.add(logLabel);

LimparLog = new JButton("Limpar Log");
LimparLog.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        LogtextArea.setText(null);
    }
});
LimparLog.setFont(new Font("Tahoma", Font.PLAIN, 14));
LimparLog.setBounds(225, 211, 141, 33);
LogPanel.add(LimparLog);

AngulotextField = new JTextField("" + dados.getAngulo());
AngulotextField.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            dados.setAngulo(Integer.parseInt(AngulotextField.getText()));
        }catch(NumberFormatException nfe) {
            AngulotextField.setText("" + dados.getAngulo());
        }
    }
});
AngulotextField.setColumns(10);
AngulotextField.setBounds(122, 60, 113, 19);
controlPanel.add(AngulotextField);

DisttextField = new JTextField("" + dados.getDistancia());
DisttextField.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            dados.setDistancia(Integer.parseInt(DisttextField.getText()));
        }catch(NumberFormatException nfe) {
            DisttextField.setText("" + dados.getDistancia());
        }
    }
});
DisttextField.setColumns(10);
DisttextField.setBounds(122, 96, 113, 19);
controlPanel.add(DisttextField);

distLabel = new JLabel("Distância");
distLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
distLabel.setBounds(41, 94, 80, 20);
controlPanel.add(distLabel);

DireitaButton = new JButton("Direita");
DireitaButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
DireitaButton.setBounds(499, 62, 90, 29);
controlPanel.add(DireitaButton);

PararButton = new JButton("Parar");
PararButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
PararButton.setBounds(386, 62, 103, 29);
controlPanel.add(PararButton);

EsquerdaButton = new JButton("Esquerda");
EsquerdaButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
EsquerdaButton.setBounds(278, 61, 97, 29);
controlPanel.add(EsquerdaButton);

FrenteButton = new JButton("Frente");
FrenteButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
FrenteButton.setBounds(387, 26, 99, 29);
controlPanel.add(FrenteButton);

canalPanel.setLayout(null);
ficheiroCanalLabel = new JLabel("Ficheiro");
```

```
        ficheiroCanalLabel.setBounds(41, 29, 54, 20);
        canalPanel.add(ficheiroCanalLabel);

        ficheiroCanalLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        canalTextField = new JTextField();
        canalTextField.setFont(new Font("Tahoma", Font.PLAIN, 14));
        canalTextField.setBounds(119, 26, 297, 25);
        canalPanel.add(canalTextField);
        canalTextField.setColumns(10);

        browseButton = new JButton("...");
        browseButton.setBounds(453, 25, 68, 29);
        canalPanel.add(browseButton);
        browseButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                JFileChooser fileChooser = new JFileChooser();
                int result = fileChooser.showOpenDialog(null);

                if (result == JFileChooser.APPROVE_OPTION) {
                    setFilePath(fileChooser.getSelectedFile().getAbsolutePath());
                    canalTextField.setText(filePath);
                }
            }
        });
        browseButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

        msgSpinner = new JSpinner();
        msgSpinner.setModel(new SpinnerNumberModel(8, 8, 12, 1));
        msgSpinner.setBounds(119, 66, 45, 29);
        canalPanel.add(msgSpinner);

        nMsgLabel = new JLabel("N°Msg");
        nMsgLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        nMsgLabel.setBounds(41, 69, 54, 20);
        canalPanel.add(nMsgLabel);

        abrirCanalRadioButton = new JRadioButton("Abrir Canal");
        abrirCanalRadioButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
        abrirCanalRadioButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setRadioB(true);
            }
        });
        abrirCanalRadioButton.setBounds(441, 60, 128, 43);
        canalPanel.add(abrirCanalRadioButton);

        contentPane.setLayout(null);
        contentPane.add(controlPanel);
        contentPane.add(canalPanel);
        contentPane.add(LogPanel);

        scrollPane = new JScrollPane();
        scrollPane.setBounds(19, 72, 561, 127);
        LogPanel.add(scrollPane);

        LogtextArea = new JTextArea();
        scrollPane.setViewportView(LogtextArea);

        setVisible(true);
    }

    public JButton getEsquerdaButton() {
        return EsquerdaButton;
    }
    public void setEsquerdaButton(JButton esquerdaButton) {
        EsquerdaButton = esquerdaButton;
    }
    public JButton getPararButton() {
        return PararButton;
    }
    public void setPararButton(JButton pararButton) {
        PararButton = pararButton;
    }
    public JButton getBtnRetaguarda() {
        return btnRetaguarda;
    }
    public void setBtnRetaguarda(JButton btnRetaguarda) {
        this.btnRetaguarda = btnRetaguarda;
    }
    public JButton getFrenteButton() {
```

```
        return FrenteButton;
    }
    public void setFrenteButton(JButton frenteButton) {
        FrenteButton = frenteButton;
    }

    public JButton getDireitaButton() {
        return DireitaButton;
    }
    public void setDireitaButton(JButton direitaButton) {
        DireitaButton = direitaButton;
    }
    public JTextArea getLogtextArea() {
        return LogtextArea;
    }

    public void setLogtextArea(JTextArea logtextArea) {
        LogtextArea = logtextArea;
    }
    public Dados getDados() {
        return dados;
    }
    public void setDados(Dados dados) {
        this.dados = dados;
    }


}
```

## Código GUI Rei:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

// GUI REI

import java.awt.EventQueue;
import javax.swing.JPanel;
import javax.swing.border.TitledBorder;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import java.awt.Font;
import javax.swing.JScrollBar;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GUI_REI extends Generico{

    private JPanel Controle_Panel;
    private JButton OitoCmdButton;
    private JButton btnComandosAleatorios;
    private boolean gerar8;
    private boolean gerar16;
    private boolean B_Frente;
    private boolean B_Tras;
    private boolean B_CDireita;
    private boolean B_CEsquerda;
    private boolean B_Parar;

    public boolean isGerar8() {
        return gerar8;
    }


    public void setGerar8(boolean gerar8) {
        this.gerar8 = gerar8;
    }


    public boolean isGerar16() {
        return gerar16;
    }


    public void setGerar16(boolean gerar16) {
        this.gerar16 = gerar16;
```

```
    }

    public boolean botoesGerarCmd() {
        return isGerar8() || isGerar16();
    }

    public void OffBotoes() {
        setGerar8(false);
        setGerar16(false);
        setB_CDireita(false);
        setB_CEsquerda(false);
        setB_Parar(false);
        setB_Frente(false);
    }

    public GUI_REI() {
        super();

        B_Frente=false;
        B_Tras=false;
        B_CDireita=false;
        B_CEsquerda=false;
        B_Parar=false;
        gerar8=false;
        gerar16=false;

        this.setTitle("Trabalho 1 - Rei");

        Controle_Panel = new JPanel();
        Controle_Panel.setBorder(new TitledBorder(null, "Controle do Robot em Modo
Autom\u00E1tico", TitledBorder.LEADING, TitledBorder.TOP, null, null));
        Controle_Panel.setBounds(21, 279, 587, 62);

        OitoCmdButton = new JButton("8 Comandos Aleatórios");
        OitoCmdButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                OffBotoes();
                setGerar8(true);
            }
        });
        OitoCmdButton.setBounds(50, 15, 193, 34);
        OitoCmdButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

        btnComandosAleatorios = new JButton("16 Comandos Aleatórios");
        btnComandosAleatorios.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                OffBotoes();
                setGerar16(true);
            }
        });
        btnComandosAleatorios.setBounds(349, 15, 193, 34);
        btnComandosAleatorios.setFont(new Font("Tahoma", Font.PLAIN, 14));

        Controle_Panel.setLayout(null);
        Controle_Panel.add(OitoCmdButton);
        Controle_Panel.add(btnComandosAleatorios);

        getContentPane().add(Controle_Panel);

        getFrenteButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                setB_Frente(true);
            }
        });

        getBtnRetaguarda().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                setB_Tras(true);
            }
        });

        getDireitaButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                setB_CDireita(true);
            }
        });

        getEsquerdaButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                setB_CEsquerda(true);
```

```
            }
        });

        getPararButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                setB_Parar(true);
            }
        });
    }

    public boolean isB_Parar() {
        return B_Parar;
    }

    public void setB_Parar(boolean b_Parar) {
        B_Parar = b_Parar;
    }

    public boolean isB_CDireita() {
        return B_CDireita;
    }

    public void setB_CDireita(boolean b_CDireita) {
        B_CDireita = b_CDireita;
    }

    public boolean isB_CEsquerda() {
        return B_CEsquerda;
    }

    public void setB_CEsquerda(boolean b_CEsquerda) {
        B_CEsquerda = b_CEsquerda;
    }

    public boolean isB_Tras() {
        return B_Tras;
    }

    public void setB_Tras(boolean b_Tras) {
        B_Tras = b_Tras;
    }

    public boolean isB_Frente() {
        return B_Frente;
    }

    public void setB_Frente(boolean b_Frente) {
        B_Frente = b_Frente;
    }

}
```

## Código GUI Súbdito:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//GUI Súbdito

import javax.swing.JPanel;
import javax.swing.border.TitledBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public class GUI_SUBDITO extends Generico {

    private JPanel Robot_panel;
    private JTextField nameTextField;
    private JLabel nomeRobotLabel;
    private JRadioButton abrirFecharRobotRadBtn;
    private boolean robotCheck;
    private boolean RobotRadioBtn;

    public GUI_SUBDITO() {
```

```
        super();

        RobotRadioBtn = false;

        this.setTitle("Trabalho 1 - Subdito");

        Robot_panel = new JPanel();
        Robot_panel.setBorder(new TitledBorder(null, "Robot", TitledBorder.LEADING,
TitledBorder.TOP, null, null));
        Robot_panel.setBounds(20, 276, 592, 69);
        Robot_panel.setLayout(null);

        nomeRobotLabel = new JLabel("Nome do Robot");
        nomeRobotLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        nomeRobotLabel.setBounds(10, 25, 111, 13);
        Robot_panel.add(nomeRobotLabel);

        nameTextField = new JTextField(dados.getNomeRobot());
        nameTextField = new JTextField();
        nameTextField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                dados.setNomeRobot(nameTextField.getText());
            }
        });
        nameTextField.setBounds(118, 24, 242, 19);
        Robot_panel.add(nameTextField);
        nameTextField.setColumns(10);

        abrirFecharRobotRadBtn = new JRadioButton("Abrir/Fechar Robot");
        abrirFecharRobotRadBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                setRobotRadioBtn(true);
                if(!dados.isOnOff()) {
                    if(dados.getRobot().OpenEV3(dados.getNomeRobot())) {
                        abrirFecharRobotRadBtn.setSelected(true);
                        dados.setOnOff(true);
                        getLogtextArea().append("Ligado ao RobotLegoEV3:" +
dados.getNomeRobot()+"\n");
                    }
                    else {
                        abrirFecharRobotRadBtn.setSelected(false);
                        getLogtextArea().append("Insucesso ao ligar o RobotLegoEV3 \n");
                    }
                }else {
                    abrirFecharRobotRadBtn.setSelected(false);
                    dados.getRobot().CloseEV3();
                    dados.setOnOff(false);
                    getLogtextArea().append("Desligado com sucesso \n");
                }
            }
        });
        abrirFecharRobotRadBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
        abrirFecharRobotRadBtn.setBounds(377, 23, 143, 21);
        Robot_panel.add(abrirFecharRobotRadBtn);

        getContentPane().add(Robot_panel);

        getFrenteButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                getDados().getRobot().Reta(dados.getDistancia());
                getDados().getRobot().Parar(false);
                getLogtextArea().append("Reta (Sucesso) " + dados.getDistancia() + "\n");
            }
        });

        getBtnRetaguarda().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                getDados().getRobot().Reta(-1*dados.getDistancia());
                getDados().getRobot().Parar(false);
                getLogtextArea().append("Tras (Sucesso) " + -1*dados.getDistancia() + "\n");
            }
        });

        getDireitaButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                getDados().getRobot().CurvarDireita(dados.getRaio(), dados.getAngulo());
                getDados().getRobot().Parar(false);
                getLogtextArea().append("CDireira (Sucesso) " + dados.getRaio() + " " +
dados.getAngulo() + "\n");
            }
```

```
        });

        getEsquerdaButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                getDados().getRobot().CurvarEsquerda(dados.getRaio(), dados.getAngulo());
                getDados().getRobot().Parar(false);
                getLogtextArea().append("CEsquerda (Sucesso) " + dados.getRaio() + " " +
dados.getAngulo() + "\n");
            }
        });

        getPararButton().addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                getDados().getRobot().Parar(true);
                getLogtextArea().append("Parar (Sucesso) \n");
            }
        });
    }

    public JPanel getRobot_panel() {
        return Robot_panel;
    }

    public void setRobot_panel(JPanel robot_panel) {
        Robot_panel = robot_panel;
    }

    public JTextField getNameTextField() {
        return nameTextField;
    }

    public void setNameTextField(JTextField nameTextField) {
        this.nameTextField = nameTextField;
    }

    public JLabel getNomeRobotLabel() {
        return nomeRobotLabel;
    }

    public void setNomeRobotLabel(JLabel nomeRobotLabel) {
        this.nomeRobotLabel = nomeRobotLabel;
    }

    public boolean getAbrirFecharRobotRadBtn() {
        return abrirFecharRobotRadBtn.isSelected();
    }

    public void setAbrirFecharRobotRadBtn(Boolean valor) {
        abrirFecharRobotRadBtn.setSelected(valor);
    }

    public boolean isRobotRadioBtn() {
        return RobotRadioBtn;
    }

    public void setRobotRadioBtn(boolean RobotRadioBtn) {
        this.RobotRadioBtn = RobotRadioBtn;
    }

}
```

## Código Classe Dados:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//Classe Dados

import robot.RobotLegoEV3;

public class Dados {

    private int raio, angulo, distancia;
    private String nomeRobot;
    private boolean onOff;
    private RobotLegoEV3 robot;

    //Construtor por defeito
```

```
    public Dados(){
        raio = 10;
        angulo = 90;
        distancia = 20;
        nomeRobot = "FT2"; //Pass 1234
        onOff = false;
        robot = new RobotLegoEV3();
    }

    public boolean isOnOff() {
        return onOff;
    }

    public void setOnOff(boolean onOff) {
        this.onOff = onOff;
    }

    public String getNomeRobot() {
        return nomeRobot;
    }

    public void setNomeRobot(String nomeRobot) {
        this.nomeRobot = nomeRobot;
    }

    public int getRaio() {
        return raio;
    }

    public void setRaio(int raio) {
        this.raio = raio;
    }

    public int getAngulo() {
        return angulo;
    }

    public void setAngulo(int angulo) {
        this.angulo = angulo;
    }

    public int getDistancia() {
        return distancia;
    }

    public void setDistancia(int distancia) {
        this.distancia = distancia;
    }

    public RobotLegoEV3 getRobot() {
        return robot;
    }

    public void setRobot(RobotLegoEV3 robot) {
        this.robot = robot;
    }
}
```

## Código Classe BufferCircularMsg:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//Classe Buffer Circular

public class BufferCircularMsg {

    private int dimensaoBuffer;

    final int DIMENSAO_MGS = 16;

    private int putBuffer, getBuffer;
    private int putBufferCheck, getBufferCheck;

    public int getPutBufferCheck() {
        return putBufferCheck;
    }
```

```
    public void setPutBufferCheck(int putBufferCheck) {
        this.putBufferCheck = putBufferCheck;
    }

    public int getGetBufferCheck() {
        return getBufferCheck;
    }

    public void setGetBufferCheck(int getBufferCheck) {
        this.getBufferCheck = getBufferCheck;
    }

    private int spinner;

    public BufferCircularMsg(int numSpinner) //msgSpinner
    {
        this.spinner = numSpinner;
        dimensaoBuffer = DIMENSAO_MGS*spinner;
        putBuffer = 0;
        getBuffer = 0;
        putBufferCheck=0;
        getBufferCheck=0;
    }

    public int getSpinner() {
        return spinner;
    }

    public void setSpinner(int spinner) {
        this.spinner = spinner;
    }

    public int getPutBuffer() {
        return putBuffer;
    }

    public void setPutBuffer(int putBuffer) {
        this.putBuffer = putBuffer;
    }

    public int getGetBuffer() {
        return getBuffer;
    }

    public void setGetBuffer(int getBuffer) {
        this.getBuffer = getBuffer;
    }

    public int getDimensaoBuffer() {
        return dimensaoBuffer;
    }

    public void setDimensaoBuffer(int dimensaoBuffer) {
        this.dimensaoBuffer = dimensaoBuffer;
    }

    public void minusGet() {
        this.getBuffer -= this.DIMENSAO_MGS;
    }

    public void minusGetCheck() {
        this.getBufferCheck -= this.DIMENSAO_MGS;
    }
}
```

## Código Classe CanalComunicacao:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//CanalComunicacao

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.BufferUnderflowException;
import java.nio.MappedByteBuffer;
```

```
import java.nio.channels.FileChannel;

public class CanalComunicacao {

    // ficheiro
    private File ficheiro;
    // canal que liga o conteúdo do ficheiro ao Buffer
    private FileChannel canal;
    // buffer
    private MappedByteBuffer buffer;

    private Mensagem msgS;

    private BufferCircularMsg bufferC;

    public Mensagem getMsgS() {
        return msgS;
    }

    public void setMsgS(Mensagem msgS) {
        this.msgS = msgS;
    }

    public CanalComunicacao(int numSpinner){
        this.ficheiro=null;
        this.canal= null;
        this.buffer= null;
        this.msgS= new Mensagem();
        this.bufferC = new BufferCircularMsg(numSpinner);
    }

    public boolean abrirCanal(String filePath) {
        // cria um ficheiro com o caminho recebido
        ficheiro = new File(filePath);
        //cria um canal de comunicação de leitura e escrita
        try {
            canal = new RandomAccessFile(ficheiro, "rw").getChannel();

        } catch (FileNotFoundException e) {return false;}
        // mapeia para memória o conteúdo do ficheiro
        try {
            buffer = canal.map(FileChannel.MapMode.READ_WRITE, 0, bufferC.getDimensaoBuffer());
            //Mensagem msgS avisa que o canal está vazio (Cmd 5)
            for(int i = 1; i <= bufferC.getSpinner();i++)
            {
                msgS.setCmd(msgS.CmdMsgVazia);
                enviarMensagem(msgS);
            }
        } catch (IOException e) { return false;}
        return true;
    }

    // recebe uma mensagem do buffer convertendo-a numa Mensagem
    public Mensagem receberMensagem() {

        if(bufferC.getGetBuffer() >= bufferC.getDimensaoBuffer()){
            bufferC.setGetBuffer(0);
        }

        getBuffer().position(bufferC.getGetBuffer());

        int num = getBuffer().getInt();
        int cmd = getBuffer().getInt();
        int arg1= getBuffer().getInt();
        int arg2= getBuffer().getInt();

        bufferC.setGetBuffer(16 + bufferC.getGetBuffer()); //16 posições do ringBuffer

        Mensagem mensagemRecebe = new Mensagem();
        mensagemRecebe.setNum(num);
        mensagemRecebe.setCmd(cmd);
        mensagemRecebe.setArg1(arg1);
        mensagemRecebe.setArg2(arg2);

        return mensagemRecebe;
    }

    // envia uma mensagem
    public void enviarMensagem(Mensagem msg) {
```

```
        if(bufferC.getPutBuffer() >= bufferC.getDimensaoBuffer()){
            bufferC.setPutBuffer(0);
        }

        getBuffer().position(bufferC.getPutBuffer());
        getBuffer().putInt(msg.getNum());
        getBuffer().putInt(msg.getCmd());
        getBuffer().putInt(msg.getArg1());
        getBuffer().putInt(msg.getArg2());

        bufferC.setPutBuffer(16 + bufferC.getPutBuffer());
    }

    public Mensagem receberMensagemCheck() {

        if(bufferC.getGetBufferCheck() >= bufferC.getDimensaoBuffer()){
            bufferC.setGetBufferCheck(0);
        }

        getBuffer().position(bufferC.getGetBufferCheck());

        int num = getBuffer().getInt();
        int cmd = getBuffer().getInt();
        int arg1= getBuffer().getInt();
        int arg2= getBuffer().getInt();

        bufferC.setGetBufferCheck(16 + bufferC.getGetBufferCheck());

        Mensagem mensagemRecebe = new Mensagem();
        mensagemRecebe.setNum(num);
        mensagemRecebe.setCmd(cmd);
        mensagemRecebe.setArg1(arg1);
        mensagemRecebe.setArg2(arg2);

        return mensagemRecebe;
    }

    public void enviarMensagemCheck(Mensagem msg) {

        if(bufferC.getPutBufferCheck() >= bufferC.getDimensaoBuffer()){
            bufferC.setPutBufferCheck(0);
        }

        getBuffer().position(bufferC.getPutBufferCheck());
        getBuffer().putInt(msg.getNum());
        getBuffer().putInt(msg.getCmd());
        getBuffer().putInt(msg.getArg1());
        getBuffer().putInt(msg.getArg2());

        bufferC.setPutBufferCheck(16 + bufferC.getPutBufferCheck()); //16 posições do ringBuffer
    }

    public BufferCircularMsg getBufferC() {
        return bufferC;
    }

    public void setBufferC(BufferCircularMsg bufferC) {
        this.bufferC = bufferC;
    }

    // fecha o canal entre o buffer e o ficheiro
    public void fecharCanal() {
        if (canal!=null)
            try {
                canal.close();
            } catch (IOException e) { canal= null; }
    }

    public File getFicheiro() {
        return ficheiro;
    }

    public void setFicheiro(File ficheiro) {
        this.ficheiro = ficheiro;
    }

    public FileChannel getCanal() {
        return canal;
    }
```

```
    public void setCanal(FileChannel canal) {
        this.canal = canal;
    }

    public MappedByteBuffer getBuffer() {
        return buffer;
    }

    public void setBuffer(MappedByteBuffer buffer) {
        this.buffer = buffer;
    }

}
```

## Código Classe CanalConsistente:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 4
//Coding UTF-08

//CanalConsistente

import java.io.IOException;
import java.nio.BufferUnderflowException;
import java.nio.channels.FileLock;import javax.swing.text.StyledEditorKit.ItalicAction;

public class CanalConsistente extends CanalComunicacao {

    FileLock fl;

    private Mensagem msg;

    public CanalConsistente(int numSpinner) {
        super(numSpinner);
        this.msg = new Mensagem();
        this.fl = null;
    }

    public boolean getAndSetLeitor(Mensagem msg) {
        fl = null;
        try {
            fl = getCanal().lock();
            Mensagem msgRecebida = receberMensagem();

            if(msgRecebida.getCmd()!= msg.CmdMsgVazia && msgRecebida.getCmd()!= msg.CmdMsgLida) {
                msg.setNum(msgRecebida.getNum());
                msg.setCmd(msgRecebida.getCmd());
                msg.setArg1(msgRecebida.getArg1());
                msg.setArg2(msgRecebida.getArg2());

                //Mensagem de Check avisa que já foi lida (Cmd 4)
                Mensagem msgCheckMensagem = new Mensagem();
                msgCheckMensagem.setCmd(msg.CmdMsgLida);
                enviarMensagemCheck(msgCheckMensagem);

                fl.release();
                return true;
            }else {
                getBufferC().minusGet();
            }
            fl.release();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (fl != null) {
                try {
                    fl.release();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return false;
    }

    public boolean getAndSetEscritor(Mensagem msgEscrever) {
        fl=null;
        try {
```

```
            fl = getCanal().lock();
            Mensagem msgRecebida = receberMensagemCheck();
            if (msgRecebida.getCmd() == msgEscrever.CmdMsgLida || msgRecebida.getCmd()==
msgEscrever.CmdMsgVazia) {
                enviarMensagem(msgEscrever);
                fl.release();
                return true;
            }else getBufferC().minusGetCheck();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            if(fl != null) {
                try {
                    fl.release();
                }catch(IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return false;
    }

    public Mensagem getMsg() {
        return msg;
    }

    public void setMsg(Mensagem msg) {
        this.msg = msg;
    }

    // usado para testes
    public void printPutandGet() {
        System.out.println("Ind de receber msg: " + getBufferC().getGetBuffer());
        System.out.println("Ind de enviar msg: " + getBufferC().getPutBuffer());
        System.out.println("Ind de receber check msg: " + getBufferC().getGetBufferCheck());
        System.out.println("Ind de enviar check msg: " + getBufferC().getPutBufferCheck() + "\n");
    }
}
```

## Código Classe Mensagem:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//Mensagem

public class Mensagem implements IMensagem{

    private int num;
    private int cmd;
    private int arg1;
    private int arg2;

    //comando 4 - já leio
    //comando 5 -canal vazio

    // Construtor
    public Mensagem() {
        this.num = 0;
        this.cmd = 9;
        this.arg1 = 0;
        this.arg2 = 0;
    }

    //CONSTRUTOR PARA COLOCAR AS MENSAGEM
    //NO ARRAYLIST (MEMÓRIA) DA FORMA PRETENDIDA
    public Mensagem(Mensagem arrayMensagem) {
        this.num = arrayMensagem.num;
        this.cmd = arrayMensagem.cmd;
        this.arg1 = arrayMensagem.arg1;
        this.arg2 = arrayMensagem.arg2;
    }

    public int getArg1() {
```

```
        return arg1;
    }

    public void setArg1(int arg1) {
        this.arg1 = arg1;
    }

    public int getArg2() {
        return arg2;
    }

    public void setArg2(int arg2) {
        this.arg2 = arg2;
    }

    public int getNum() {
        return num;
    }

    public void setNum(int num) {
        this.num = num;
    }

    public int getCmd() {
        return cmd;
    }

    public void setCmd(int cmd) {
        this.cmd = cmd;
    }

    public void print() {
        System.out.println(this.num +" "+ this.cmd + " " + this.arg1 + " " + this.arg2);
    }

    public String toString() {
        return "Msg: num: " + this.num +" cmd: "+ this.cmd + " arg1: " + this.arg1 + " arg2: " +
this.arg2;
    }

}
```

## Código Interface IMensagem:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//IMensagem

public interface IMensagem {
    final int CmdParar = 0;
    final int CmdReta = 1;
    final int CmdCurvaDireita = 2;
    final int CmdCurvaEsquerda = 3;
    final int CmdMsgLida = 4;
    final int CmdMsgVazia = 5;
}
```

## Código Classe CriarMensagem:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//CriarMensagem

import java.util.Iterator;
import java.util.Random;

public class CriarMensagem {

    private Dados d;
    private Mensagem msg;
    private int numUltMsg;
    private int numSpinner;
```

```
public CriarMensagem(Dados dados, int numSpinnerInput) {
    d = dados;
    msg = null;
    this.numUltMsg = 0;
    numSpinner = numSpinnerInput;
}

public Mensagem MsgFrente() {
    msg = new Mensagem();
    this.numUltMsg++;
    msg.setNum(this.numUltMsg);
    msg.setCmd(msg.CmdReta);
    msg.setArg1(d.getDistancia());
    msg.setArg2(0);
    return msg;
}

public Mensagem MsgTras() {
    msg = new Mensagem();
    this.numUltMsg++;
    msg.setNum(this.numUltMsg);
    msg.setCmd(msg.CmdReta);
    msg.setArg1(-1*d.getDistancia());
    msg.setArg2(0);
    return msg;
}

public Mensagem MsgParar() {
    msg = new Mensagem();
    this.numUltMsg++;
    msg.setNum(this.numUltMsg);
    msg.setCmd(msg.CmdParar);
    msg.setArg1(1);
    msg.setArg2(0);
    return msg;
}

public Mensagem MsgCurDireita() {
    msg = new Mensagem();
    this.numUltMsg++;
    msg.setNum(this.numUltMsg);
    msg.setCmd(msg.CmdCurvaDireita);
    msg.setArg1(d.getRaio());
    msg.setArg2(d.getAngulo());
    return msg;
}

public Mensagem MsgCurEsquerda() {
    msg = new Mensagem();
    this.numUltMsg++;
    msg.setNum(this.numUltMsg);
    msg.setCmd(msg.CmdCurvaEsquerda);
    msg.setArg1(d.getRaio());
    msg.setArg2(d.getAngulo());
    return msg;
}

public Mensagem MsgAleatorio() {
    msg = new Mensagem();
    this.numUltMsg++;
    msg.setNum(this.numUltMsg);
    msg.setCmd((int) (Math.random() * 3) + 1);
    if (msg.getCmd()==msg.CmdReta) {
        msg.setArg1((int) (Math.random()* 40) + 10 );
        msg.setArg2(0);
    }else {
        msg.setArg1((int) (Math.random() * 30) + 1);
        msg.setArg2((int) (Math.random() * 360));

    }

    return msg;
}


public int getNumSpinner() {
    return numSpinner;
}
```

```
    public void setNumSpinner(int numSpinner) {
        this.numSpinner = numSpinner;
    }


    public Dados getD() {
        return d;
    }

    public void setD(Dados d) {
        this.d = d;
    }

    public Mensagem getMsg() {
        return msg;
    }

    public void setMsg(Mensagem msg) {
        this.msg = msg;
    }

    public int getNumUltMsg() {
        return numUltMsg;
    }

    public void setNumUltMsg(int numUltMsg) {
        this.numUltMsg = numUltMsg;
    }

    public void minusNum() {
        this.numUltMsg = this.numUltMsg-1;
    }
}
```

## Código Classe Rei:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//Rei

import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Rei implements IRei {
    private CanalConsistente canal;
    private GUI_REI gui_REI;
    private int state;
    private CriarMensagem cMsg;
    private Mensagem mensagem;
    private boolean guiClose;
    private int tempWait;
    private int cont16;
    private int cont8;

    public Rei() {
        gui_REI = new GUI_REI();
        canal = null;
        state = IDLE_STATE;
        cMsg = null;
        mensagem = null;
        guiClose = false;
        tempWait = 0;
        cont16 = 0;
        cont8=0;
        gui_REI.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                setGuiClose(true);// A janela foi fechada
            }
        });
    }

    public void run() {
        while(state!=END_STATE) {
            automato();
        }
```

```
}

public void automato() {

    switch (state) {

        case IDLE_STATE:
            if (gui_REI.getAbrirCanal()) {
                canal = new CanalConsistente(gui REI.spinnerValue());
                cMsg = new CriarMensagem(gui_REI.dados, gui_REI.spinnerValue());
                state = ABRIR_CANAL;
                break;
            } else {
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    System.err.println("O Canal não foi fechado");
                    e.printStackTrace();
                }
            }
            break;
        case ABRIR_CANAL:
            canal.abrirCanal(gui_REI.getFilePath());
            state=WAIT_COMANDO;
            break;
        case WAIT_COMANDO:
            if (!gui_REI.getAbrirCanal()) {
                state = IDLE_STATE;
                break;
            }
            if (gui_REI.getAtivar()) {
                state = SELECT_MSG;
                break;
            }
            if (isGuiClose()) {
                System.out.println("END_STATE");
                state = END_STATE;
                break;
            }
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                System.err.println("A espera foi interrumpida");
                e.printStackTrace();
            }
            break;
        case SELECT_MSG:
            if (gui_REI.isB_Frente()) {
                state=SEND_FRENTE;
                break;
            }

            if (gui_REI.isB_Tras()) {
                state=SEND_TRAS;
                break;
            }

            if (gui_REI.isB_CDireita()) {
                state=SEND_CDIREITA;
                break;
            }

            if (gui_REI.isB_CEsquerda()) {
                state=SEND_CESQUERDA;
                break;
            }

            if (gui_REI.isB_Parar()) {
                state=SEND_PARAR;
                break;
            }

            if (gui_REI.isGerar8()) {
                state=SEND_8CMD;
                break;
            }

            if (gui_REI.isGerar16()) {
                state=SEND_16CMD;
                break;
```

```
        }
        if (isGuiClose()) {
            System.out.println("END_STATE");
            state = END_STATE;
            break;
        }
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.err.println("A espera foi interrumpida");
            e.printStackTrace();
        }
        state = WAIT_COMANDO;
        break;
    case SEND_FRENTE:

        mensagem = cMsg.MsgFrente();
        if (canal.getAndSetEscritor(mensagem)) {
            gui_REI.getLogtextArea().append(mensagem.toString() +"\n");
        }else {
            gui_REI.getLogtextArea().append("Erro em envio \n");
            cMsg.minusNum();
        }
        gui_REI.setB_Frente(false);
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.err.println("A espera foi interrumpida");
            e.printStackTrace();
        }
        state = WAIT_COMANDO;
        break;

    case SEND_TRAS:

        mensagem = cMsg.MsgTras();
        if (canal.getAndSetEscritor(mensagem)) {
            gui_REI.getLogtextArea().append(mensagem.toString() +"\n");
        }else {
            gui_REI.getLogtextArea().append("Erro em envio \n");
            cMsg.minusNum();
        }
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.err.println("A espera foi interrumpida");
            e.printStackTrace();
        }
        gui_REI.setB_Tras(false);
        state = WAIT_COMANDO;
        break;

    case SEND_CDIREITA:

        mensagem = cMsg.MsgCurDireita();
        if (canal.getAndSetEscritor(mensagem)) {
            gui_REI.getLogtextArea().append(mensagem.toString() +"\n");
        }else {
            gui_REI.getLogtextArea().append("Erro em envio \n");
            cMsg.minusNum();
        }
        gui_REI.setB_CDireita(false);
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.err.println("A espera foi interrumpida");
            e.printStackTrace();
        }
        state = WAIT_COMANDO;
        break;

    case SEND_CESQUERDA:

        mensagem = cMsg.MsgCurEsquerda();
        if (canal.getAndSetEscritor(mensagem)) {
            gui_REI.getLogtextArea().append(mensagem.toString() +"\n");
        }else {
            gui_REI.getLogtextArea().append("Erro em envio \n");
            cMsg.minusNum();
        }
```

```
        gui_REI.setB_CEsquerda(false);
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.err.println("A espera foi interrumpida");
            e.printStackTrace();
        }
        state = WAIT_COMANDO;
        break;

    case SEND_PARAR:

        mensagem = cMsg.MsgParar();
        if (canal.getAndSetEscritor(mensagem)) {
            gui REI.getLogtextArea().append(mensagem.toString() +"\n");
        }else {
            gui_REI.getLogtextArea().append("Erro em envio \n");
            cMsg.minusNum();
        }
        gui_REI.setB_Parar(false);
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.err.println("A espera foi interrumpida");
            e.printStackTrace();
        }
        state = WAIT_COMANDO;
        break;

    case SEND_8CMD:
        if(cont8!=NUM_8CMD) {
            mensagem = cMsg.MsgAleatorio();
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.err.println("A espera foi interrumpida");
                e.printStackTrace();
            }
            if (canal.getAndSetEscritor(mensagem)){
                gui_REI.getLogtextArea().append(mensagem.toString() +"\n");
                cont8++;
            }else {
                gui_REI.getLogtextArea().append("Erro em envio \n");
                cMsg.minusNum();
                state = WAIT_COMANDO;
                break;
            }
        }else {
            cont8=0;
            gui_REI.OffBotoes();
            gui_REI.setGerar8(false);
        }
        state = WAIT_COMANDO;
        break;
    case SEND_16CMD:
        if(cont16!=NUM_16CMD) {
            mensagem = cMsg.MsgAleatorio();
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.err.println("A espera foi interrumpida");
                e.printStackTrace();
            }
            if (canal.getAndSetEscritor(mensagem)){
                gui_REI.getLogtextArea().append(mensagem.toString() +"\n");
                cont16++;
            }else {
                gui_REI.getLogtextArea().append("Erro em envio \n");
                cMsg.minusNum();
                state = WAIT_COMANDO;
                break;
            }
        }else {
            cont16=0;
            gui_REI.OffBotoes();
            gui_REI.setGerar16(false);
        }
        state = WAIT_COMANDO;
        break;
    case END_STATE:
```

```
                canal.fecharCanal();

                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    System.err.println("A espera foi interrumpida");
                    e.printStackTrace();
                }
                break;
            }
        }

        public boolean isGuiClose() {
            return guiClose;
        }

        public void setGuiClose(boolean guiClose) {
            this.guiClose = guiClose;
        }

        public CanalConsistente getCanal() {
            return canal;
        }

        public void setCanal(CanalConsistente canal) {
            this.canal = canal;
        }

        public GUI_REI getGui_REI() {
            return gui_REI;
        }

        public void setGui_REI(GUI_REI gui_REI) {
            this.gui_REI = gui_REI;
        }

        public int getState() {
            return state;
        }

        public void setState(int state) {
            this.state = state;
        }


        public static void main(String[] args) {
            Rei r = new Rei();
            r.run();
        }
}
```

## Código Interface IRei:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//IRei

public interface IRei {
    final int WAIT_COMANDO = 0;
    final int SELECT_MSG = 1;
    final int SEND_FRENTE = 2;
    final int SEND_TRAS = 3;
    final int SEND_CDIREITA = 4;
    final int SEND_CESQUERDA = 5;
    final int SEND_8CMD = 6;
    final int SEND_16CMD = 7;
    final int SEND_PARAR = 8;
    final int END_STATE = 9;
    final int IDLE_STATE = 10;
    final int ABRIR_CANAL = 11;
    final int NUM_8CMD = 8;
    final int NUM_16CMD= 16;
    final int VEL = 20;
    final int TEMPO_COMUNIC = 100;
}
```

## Código Classe Subdito:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//Subdito


import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;
import java.util.Iterator;
import java.lang.Math;
import javax.accessibility.AccessibleTableModelChange;

public class Subdito implements ISubdito {
    private CanalConsistente canal;
    private GUI_SUBDITO gui_SUB;
    private int state;
    private int state2;
    private boolean guiClose;
    private Mensagem msg;
    private int idxUltMsg;
    private int tempWait;
    private int distancia_robot;
    private double angulo_rad;

    private ArrayList<Mensagem> memoria;

    public Subdito() {
        gui_SUB = new GUI_SUBDITO();
        canal = null;
        state = IDLE_STATE;
        state2 = ATIVAR_ROBOT;
        msg = new Mensagem();
        guiClose = false;
        idxUltMsg = 0;
        tempWait = 0;
        distancia_robot=0;
        angulo_rad=0;
        memoria = new ArrayList<Mensagem>();


        gui_SUB.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                setGuiClose(true);// A janela foi fechada
            }
        });
    }

    public void run() {
        while(state!=END_STATE) {
            automato_Comunicacao();
            exeComando_Robot();
        }
    }

    public void automato_Comunicacao() {
        switch (state) {
            case IDLE_STATE:
                if (gui_SUB.getAbrirCanal() && gui_SUB.getAtivar()) {
                    canal = new CanalConsistente(gui_SUB.spinnerValue());
                    state = ABRIR_CANAL;
                    break;
                } else {
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                        System.err.println("O Canal não foi fechado");
                        e.printStackTrace();
                    }
                }
                break;
            case ABRIR_CANAL:
                canal.abrirCanal(gui_SUB.getFilePath());
                state=READ_MSG_CC;
                break;
```

```
            case READ_MSG_CC:
                if (isGuiClose()) {
                    state = END_STATE;
                    break;
                }
                if (canal.getAndSetLeitor(msg)){
                    gui_SUB.getFrenteButton().setEnabled(false);
                    gui_SUB.getBtnRetaguarda().setEnabled(false);
                    gui_SUB.getDireitaButton().setEnabled(false);
                    gui_SUB.getEsquerdaButton().setEnabled(false);
                    gui_SUB.getPararButton().setEnabled(false);
                    state=SAVE_MSG_MEM;
                    break;
                }else {
                    gui_SUB.getFrenteButton().setEnabled(true);
                    gui_SUB.getBtnRetaguarda().setEnabled(true);
                    gui_SUB.getDireitaButton().setEnabled(true);
                    gui_SUB.getEsquerdaButton().setEnabled(true);
                    gui_SUB.getPararButton().setEnabled(true);
                    state=DORMIR_SUB;
                    break;
                }
            case SAVE_MSG_MEM:
                memoria.add(new Mensagem(msg));
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    System.err.println("Erro no sleep");
                    e.printStackTrace();
                }
                state=DORMIR_SUB;
                break;
            case DORMIR_SUB:
                try {
                    Thread.sleep(500);
                } catch (Exception e) {
                    System.out.println("Erro no sleep");
                }
                state = READ_MSG_CC;
                break;
        }
    }

    public void exeComando_Robot() {
        switch (state2) {
            case ATIVAR_ROBOT:
                if (gui_SUB.isRobotRadioBtn()) {
                    state2 = DORMIR;
                    break;
                } else {
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                        System.err.println("Erro no sleep");
                        e.printStackTrace();
                    }
                }
            case DORMIR:
                if (idxUltMsg != memoria.size()) {
                    state2 = READ_MSG_MEM;
                }
                break;
            case READ_MSG_MEM:
                if (idxUltMsg == memoria.size()) {
                    state2 = DORMIR;
                    break;
                }
                if (memoria.get(idxUltMsg).getCmd()==CMD_PARAR) {
                    state2=EST_PARAR;
                    break;
                }
                if (memoria.get(idxUltMsg).getCmd()==CMD_RETA) {
                    state2=EST_RETA;
                    break;
                }
                if (memoria.get(idxUltMsg).getCmd()==CMD_CurDireita) {
                    state2=EST_CurDireita;
                    break;
                }
                if (memoria.get(idxUltMsg).getCmd()==CMD_CurEsquerda) {
```

```
                    state2=EST_CurEsquerda;
                    break;
                }
            case EST_RETA:
                gui_SUB.getDados().getRobot().Reta(memoria.get(idxUltMsg).getArg1());
                gui_SUB.getDados().getRobot().Parar(false);

                gui_SUB.getLogtextArea().append(memoria.get(idxUltMsg).toString()+" \n");

                distancia_robot = memoria.get(idxUltMsg).getArg1();

                state2 = WAIT_STATE;
                break;
            case EST_CurDireita:
                gui_SUB.getDados().getRobot().CurvarDireita(memoria.get(idxUltMsg).getArg1(),
memoria.get(idxUltMsg).getArg2());
                gui_SUB.getDados().getRobot().Parar(false);
                gui_SUB.getLogtextArea().append(memoria.get(idxUltMsg).toString()+" \n");

                angulo_rad = memoria.get(idxUltMsg).getArg2() * (2*(int)Math.PI)/360;
                distancia_robot = (int)angulo_rad * memoria.get(idxUltMsg).getArg1();

                state2 = WAIT_STATE;
                break;
            case EST_CurEsquerda:
                gui_SUB.getDados().getRobot().CurvarEsquerda(memoria.get(idxUltMsg).getArg1(),
memoria.get(idxUltMsg).getArg2());
                gui_SUB.getDados().getRobot().Parar(false);
                gui_SUB.getLogtextArea().append(memoria.get(idxUltMsg).toString()+" \n");
                angulo_rad = memoria.get(idxUltMsg).getArg2() * (2*(int)Math.PI)/360;
                distancia_robot = (int)angulo_rad * memoria.get(idxUltMsg).getArg1();

                state2 = WAIT_STATE;
                break;
            case EST_PARAR:
                gui_SUB.getDados().getRobot().Parar(true);
                distancia_robot=0;
                gui_SUB.getLogtextArea().append(memoria.get(idxUltMsg).toString()+" \n");
                state2 = WAIT_STATE;
                break;
            case WAIT_STATE:
                if (distancia_robot!=0) {
                    tempWait = distancia_robot/VEL;
                    try {
                        Thread.sleep(tempWait*1000 + TEMPO_COMUNIC);
                    } catch (Exception e) {
                        System.out.println("Erro no sleep");
                    }
                }else {
                    try {
                        Thread.sleep(TEMPO_COMUNIC);
                    } catch (Exception e) {
                        System.out.println("Erro no sleep");
                    }
                }
                idxUltMsg++;
                state2 = DORMIR;
                break;
        }
    }

    public static void main(String[] args) {
        Subdito s = new Subdito();
        s.run();
    }

    public CanalConsistente getCanal() {
        return canal;
    }

    public void setCanal(CanalConsistente canal) {
        this.canal = canal;
    }

    public GUI_SUBDITO getGui_SUB() {
        return gui_SUB;
    }

    public void setGui_SUB(GUI_SUBDITO gui_SUB) {
        this.gui_SUB = gui_SUB;
```

```
    }

    public int getState() {
        return state;
    }

    public void setState(int state) {
        this.state = state;
    }

    public boolean isGuiClose() {
        return guiClose;
    }

    public void setGuiClose(boolean guiClose) {
        this.guiClose = guiClose;
    }

    public Mensagem getMensagem() {
        return msg;
    }

    public void setMensagem(Mensagem msg) {
        this.msg = msg;
    }

    public ArrayList<Mensagem> getMemoria() {
        return memoria;
    }

    public void setMemoria(ArrayList<Mensagem> memoria) {
        this.memoria = memoria;
    }
}
```

## Código Interface ISubdito:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP1 - Aula 7
//Coding UTF-08

//ISubdito

public interface ISubdito {
    final int IDLE_STATE = 0;
    final int ATIVAR_ROBOT = 12;
    final int ABRIR_CANAL = 2;
    final int READ_MSG_CC = 3;
    final int SAVE_MSG_MEM = 4;
    final int DORMIR = 5;
    final int EST_PARAR = 6;
    final int EST_RETA = 7;
    final int EST_CurDireita = 8;
    final int EST_CurEsquerda = 9;
    final int READ_MSG_MEM = 10;
    final int VEL = 20;
    final int TEMPO_COMUNIC = 100;
    final int END_STATE = 11;
    final int WAIT_STATE = 13;
    final int CMD_PARAR = 0;
    final int CMD_RETA = 1;
    final int CMD_CurDireita = 2;
    final int CMD_CurEsquerda = 3;
    final int DORMIR_SUB=14;
}
```