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

Fundamentos de Sistemas Operativos

Relatório: Trabalho Prático 2 - Aula 4

Turma: 31D — Grupo: 4

Nome: João Ramos — Número: 50730

Nome: Miguel Alcobia — Número: 50746

Nome: Fábio Pestana — Número: 50756

Professor: Jorge Pais

Data: 5 de Novembro de 2023

Índice

Lista de figuras

Abreviaturas e símbolos

Lista de abreviaturas

GUI	Graphical User Interface
API	Application Programming Interface

Objetivos

Neste trabalho, os alunos terão como objetivo desenvolver um processo REI_SUBDITO, constituído por três tarefas: uma que implementa o comportamento REI; outra o comportamento SUBDITO, que conhece a API do robot, e a terceira tarefa é a GRAVAR que permite gravar alguns movimentos do robot num ficheiro.

A avaliação do trabalho está divida ao longo das seis aulas em que o trabalho deverá ser realizado. Na quarta aula, o objetivo era o desenho do diagrama de atividades do GRAVAR e a sua respetiva implementação.

Aula 4 - Desenho do diagrama de atividades do Gravar e respetiva implementação

1. 1 - Desenho do diagrama de atividades do Gravar

Seguindo as indicações do que é suposto implementar neste comportamento, chegou-se ao seguinte diagrama:

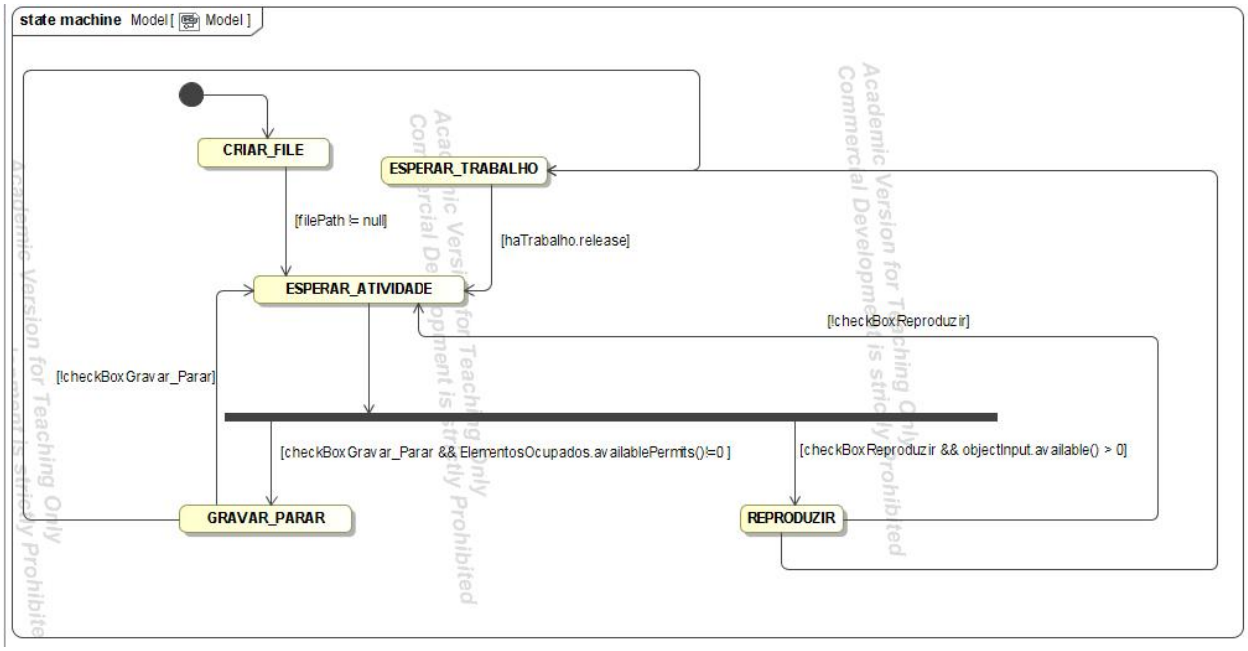

Figura 1 - Diagrama do autómato Comunicação do Gravar

O diagrama começa no estado `CRIAR_FILE`, onde criamos o ficheiro que será usado para a gravação dos comandos e para a sua leitura, para uma futura reprodução. Muda-se para o estado `ESPERAR_ATIVIDADE`, onde muda para o estado `GRAVAR_PARAR`, caso a *checkbox* "Gravar" esteja marcada e o buffer tenha já mensagens por ler. Também pode ir para o estado `REPRODUZIR`, caso a *checkbox* "Reproduzir" esteja marcada e o ficheiro ainda tenha informação por ler.

Se as *checkboxes* forem desmarcadas, volta-se para o estado `ESPERAR_ATIVIDADE`. Contudo, se continuarem ativas, entra-se no estado `ESPERAR_TRABALHO`, onde se espera a disponibilidade do Semáforo `haTrablho` para prosseguir .

1.2 - Implementação do comportamento Gravar

No `GRAVAR_PARAR` usa-se o `ObjectOutputStream.write()` para escrever as Mensagens no ficheiro; enquanto no `REPRODUZIR` usa-se o `ObjectInputStream.read()` para ler a

informação do ficheiro.

No `ESPERAR_TRABALHO` é feito o *acquire* do Semáfore `haTrabalho`, e o seu *release* é realizado após cada passagem pelo estado `GRAVAR_PARAR` e `REPRODUZIR`.

Observações

No decorrer das aulas, o professor sugeriu uma implementação que consiste na implementação de uma classe `MyRobotLegoEV3`, da qual o `Gravar_Spy` derivará. Desta forma, o `REI_SUBDITO` é quem altera o "robot" com que o `Subdito` está a comunicar. Sendo assim, o `Gravar` passa a agir como um "espião".

O grupo optou por esta implementação, mas no momento da elaboração deste relatório, ainda não se encontra completamente implementada; embora o diagrama acima esteja presente no autómato desenvolvido na classe GravarSpy.

Bibliografia

Consulta:

Mooddle:

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

Java ObjectInputStream (With Examples). (n.d.). Www.programiz.com. https://www.programiz.com/java-programming/objectinputstream

Código

Aula 4 - Desenho do diagrama de atividades do Gravar e respetiva implementação:

Código GUI Gravar:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP2 - Aula 4
//Coding UTF-08

// GUI Gravar

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GRAVAR extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JPanel panel_File;
    private JPanel panel_Gravar_Reproduzir;
    private JPanel log_panel;
    private JTextField textField_filepath;
    private JLabel label_file;
    private JButton searchButton;
    private JCheckBox checkBoxGravar_Parar;
    private JCheckBox checkBoxReproduzir;
    private JScrollPane scrollPane;
    private JLabel label_log;
    private JButton buttonCleanLog;
    private JTextArea LogtextArea;
    private boolean gravar_parar;
    private boolean reproduzir;
    private String filePath;

    public static void main(String[] args) {
        GRAVAR gravar = new GRAVAR();
    }

    public GRAVAR() {
        setTitle("Trabalho 2 - Gravar");

        this.gravar_parar = false;
        this.reproduzir = false;


        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 380);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

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

        panel_File = new JPanel();
        panel_File.setLayout(null);
        panel_File.setBorder(new TitledBorder(null, "Definir o nome do ficheiro",
```

```
TitledBorder.LEADING, TitledBorder.TOP, null, null));
        panel_File.setBounds(10, 0, 416, 69);


        textField_filepath = new JTextField();
        textField_filepath.setFont(new Font("Tahoma", Font.PLAIN, 12));
        textField_filepath.setColumns(10);
        textField_filepath.setBounds(67, 24, 276, 30);
        panel_File.add(textField_filepath);

        label_file = new JLabel("Ficheiro:");
        label_file.setFont(new Font("Tahoma", Font.PLAIN, 14));
        label_file.setBounds(10, 30, 57, 13);
        panel_File.add(label_file);

        searchButton = new JButton("...");
        searchButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                int result = fileChooser.showOpenDialog(null);

                if (result == JFileChooser.APPROVE_OPTION) {
                    setFilePath(fileChooser.getSelectedFile().getAbsolutePath());
                    textField_filepath.setText(filePath);
                }
            }
        });
        searchButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
        searchButton.setBounds(353, 24, 43, 30);
        panel_File.add(searchButton);

        panel_Gravar_Reproduzir = new JPanel();
        panel_Gravar_Reproduzir.setLayout(null);
        panel_Gravar_Reproduzir.setBorder(new TitledBorder(null, "Gravar & Reproduzir",
TitledBorder.LEADING, TitledBorder.TOP, null, null));
        panel_Gravar_Reproduzir.setBounds(10, 79, 416, 57);

        checkBoxGravar_Parar = new JCheckBox("Gravar/Parar");
        checkBoxGravar_Parar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setReproduzir(false);
                setGravar_parar(true);
                checkBoxReproduzir.setSelected(false);
            }
        });
        checkBoxGravar_Parar.setFont(new Font("Tahoma", Font.PLAIN, 14));
        checkBoxGravar_Parar.setBounds(32, 16, 103, 25);
        panel_Gravar_Reproduzir.add(checkBoxGravar_Parar);

        checkBoxReproduzir = new JCheckBox("Reproduzir");
        checkBoxReproduzir.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setGravar_parar(false);
                setReproduzir(true);
                checkBoxGravar_Parar.setSelected(false);
            }
        });
        checkBoxReproduzir.setFont(new Font("Tahoma", Font.PLAIN, 14));
        checkBoxReproduzir.setBounds(288, 16, 93, 25);
        panel_Gravar_Reproduzir.add(checkBoxReproduzir);

        log_panel = new JPanel();
        log_panel.setLayout(null);
        log_panel.setBounds(10, 140, 416, 193);

        scrollPane = new JScrollPane();
        scrollPane.setBounds(0, 54, 416, 129);
        log_panel.add(scrollPane);

        label_log = new JLabel("Log");
        label_log.setFont(new Font("Tahoma", Font.PLAIN, 14));
        label_log.setBounds(11, 13, 54, 18);
        log_panel.add(label_log);

        buttonCleanLog = new JButton("Limpar Log");
        buttonCleanLog.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                LogtextArea.setText(null);
            }
        });
```

```
        buttonCleanLog.setFont(new Font("Tahoma", Font.PLAIN, 13));
        buttonCleanLog.setBounds(294, 10, 110, 34);
        log_panel.add(buttonCleanLog);

        contentPane.add(panel_File);
        contentPane.add(panel_Gravar_Reproduzir);
        contentPane.add(log_panel);

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

        setVisible(true);
    }

    public boolean isGravar parar() {
        return gravar_parar;
    }

    public void setGravar_parar(boolean gravar_parar) {
        this.gravar_parar = gravar_parar;
    }

    public boolean isReproduzir() {
        return reproduzir;
    }

    public void setReproduzir(boolean reproduzir) {
        this.reproduzir = reproduzir;
    }

    public String getFilePath() {
        return filePath;
    }
    public void setFilePath(String filePath) {
        this.filePath = filePath;
    }

    public JTextArea getLogtextArea() {
        return LogtextArea;
    }

    public void setLogtextArea(JTextArea logtextArea) {
        LogtextArea = logtextArea;
    }


}
```

Código Classe MyRobotLegoEV3:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP2 - Aula 4
//Coding UTF-08

//Classe MyRobotLegoEV3

import robot.RobotLegoEV3;

public class MyRobotLegoEV3{
    private String nomeRobot;
    private boolean onOff;
    private RobotLegoEV3 robot;

    public MyRobotLegoEV3() {
        nomeRobot = "FT1"; //Pass 1234
        onOff = false;
        robot = new RobotLegoEV3();
    }

    public void frente(int distancia) {
        robot.Reta(distancia);
    }

    public boolean Open(String nome) {
        return robot.OpenEV3(nome);
    }

    public void Close() {
```

```
        robot.CloseEV3();
    }

    public void tras(int distancia) {
        robot.Reta(-1*distancia);
    }

    public void curvaDireita(int raio, int angulo) {
        robot.CurvarDireita(raio, angulo);
    }

    public void curvaEsquerda(int raio, int angulo) {
        robot.CurvarEsquerda(raio, angulo);
    }

    public void parar(boolean parar) {
        robot.Parar(parar);
    }

    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 RobotLegoEV3 getRobot() {
        return robot;
    }

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

Código Classe Gravar_Spy:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP2 - Aula 4
//Coding UTF-08

//Gravar

import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.concurrent.Semaphore;

import org.xml.sax.ext.LexicalHandler;

import robot.RobotLegoEV3;

public class Gravar_Spy extends MyRobotLegoEV3 implements IGravar_Spy, Runnable{

    private BufferCircular bCircular;
    private Dados dados;
    private GRAVAR gravarGUI;
    private boolean guiClose;
    private int state;
    private Mensagem msg;
    private Mensagem msgReadCheck;

    private int contIdx;
```

```
private Semaphore haTrabalho;

private File fileGravar;
private FileInputStream inputStream;
private FileOutputStream outputStream;
private ObjectInputStream objectInput;
private ObjectOutputStream objectOutput;

public Gravar Spy() {

    dados = new Dados();
    gravarGUI = new GRAVAR();
    state = CRIAR_FILE;
    msg = null;
    msgReadCheck = new Mensagem();

    contIdx = 0;
    haTrabalho = new Semaphore(0);

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

public Gravar_Spy(MyRobotLegoEV3 r) {

    setNomeRobot(r.getNomeRobot());
    setOnOff(r.isOnOff());
    setRobot(r.getRobot());

    dados = new Dados();
    gravarGUI = new GRAVAR();
    state = CRIAR_FILE;
    msg = null;
    msgReadCheck = new Mensagem();

    contIdx = 0;
    haTrabalho = new Semaphore(0);

    gravarGUI.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 CRIAR_FILE:
            System.out.println("CRIAR_FILE");
            if (gravarGUI.getFilePath() != null) {
                fileGravar = new File(gravarGUI.getFilePath());
                try {
                    if (fileGravar.createNewFile()) {
                        outputStream = new FileOutputStream(fileGravar, true);
                        objectOutput = new ObjectOutputStream(outputStream);

                        inputStream = new FileInputStream(fileGravar);
                        objectInput = new ObjectInputStream(inputStream);

                        state = ESPERAR_ATIVIDADE;
                        break;
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } else {
                try {
                    Thread.sleep(1);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
                }
                break;
            case ESPERAR_TRABALHO:
                System.out.println("ESPERAR_TRABALHO");
                try {
                    haTrabalho.acquire();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                state = ESPERAR_ATIVIDADE;
                break;
            case ESPERAR_ATIVIDADE:
                if (isGuiClose()) {
                    state = END_STATE;
                    break;
                }
                if (gravarGUI.isGravar_parar() ) {
                    state = GRAVAR_PARAR;
                }else if (gravarGUI.isReproduzir()) {
                    try {
                        if (objectInput.available() > 0) {
                            state = REPRODUZIR;
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }else {
                    state = ESPERAR_ATIVIDADE;
                }
                break;
            case GRAVAR_PARAR:
                System.out.println("GRAVAR_PARAR");
                try {
                    msg = bCircular.ler();

                    if (msg != null) {
                        objectOutput.write(msg.getNum());
                        objectOutput.write(msg.getCmd());
                        objectOutput.write(msg.getArg1());
                        objectOutput.write(msg.getArg2());

                        objectOutput.flush();

                        gravarGUI.getLogtextArea().append("G:"+ msg.toString() + "\n");
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
                haTrabalho.release();
                state = ESPERAR_TRABALHO;
                break;
            case REPRODUZIR:
                System.out.println("REPRODUZIR");
                try {
                    msgReadCheck.setNum(objectInput.read());
                    msgReadCheck.setCmd(objectInput.read());
                    msgReadCheck.setArg1(objectInput.read());
                    msgReadCheck.setArg2(objectInput.read());

                    gravarGUI.getLogtextArea().append("R:"+ msgReadCheck.toString() + "\n");
                } catch (Exception e) {
                    e.printStackTrace();
                }
                haTrabalho.release();
                state = ESPERAR_TRABALHO;
                break;
        }
    }

    public synchronized void frente(int distancia) {
        super.frente(distancia);
        if (gravarGUI.isGravar_parar() && bCircular.getElementosOcupados().availablePermits()!=0)
{
            try {
                objectOutput.write(contIdx);
                objectOutput.write(CMD_RETA);
                objectOutput.write(distancia);
                objectOutput.write(0);

                objectOutput.flush();
```

```
                gravarGUI.getLogtextArea().append("G:"+ msg.toString() + "\n");

            } catch (Exception e) {}

            contIdx++;
        }
    }

    public synchronized void tras(int distancia) {
        super.tras(distancia);
        if (gravarGUI.isGravar_parar() && bCircular.getElementosOcupados().availablePermits()!=0)
{
            try {
                objectOutput.write(contIdx);
                objectOutput.write(CMD_RETA);
                objectOutput.write(distancia);
                objectOutput.write(0);

                objectOutput.flush();

                gravarGUI.getLogtextArea().append("G:"+ msg.toString() + "\n");

            } catch (Exception e) {}

            contIdx++;
        }
    }

    public synchronized void curvaDireita(int raio, int angulo) {
        super.curvaDireita(raio, angulo);
        if (gravarGUI.isGravar_parar() && bCircular.getElementosOcupados().availablePermits()!=0)
{
            try {
                objectOutput.write(contIdx);
                objectOutput.write(CMD_CDIREITA);
                objectOutput.write(raio);
                objectOutput.write(angulo);
                objectOutput.write(0);

                objectOutput.flush();

                gravarGUI.getLogtextArea().append("G:"+ msg.toString() + "\n");

            } catch (Exception e) {}

            contIdx++;
        }
    }

    public synchronized void curvaEsquerda(int raio, int angulo) {
        super.curvaEsquerda(raio, angulo);
        if (gravarGUI.isGravar_parar() && bCircular.getElementosOcupados().availablePermits()!=0)
{
            try {
                objectOutput.write(contIdx);
                objectOutput.write(CMD_CESQUERDA);
                objectOutput.write(raio);
                objectOutput.write(angulo);
                objectOutput.write(0);

                objectOutput.flush();

                gravarGUI.getLogtextArea().append("G:"+ msg.toString() + "\n");

            } catch (Exception e) {}

            contIdx++;
        }
    }

    public BufferCircular getbCircular() {
        return bCircular;
    }

    public void setbCircular(BufferCircular bCircular) {
        this.bCircular = bCircular;
    }

    public Dados getDados() {
```

```
    return dados;
}

public void setDados(Dados dados) {
    this.dados = dados;
}

public GRAVAR getGravarGUI() {
    return gravarGUI;
}

public void setGravarGUI(GRAVAR gravarGUI) {
    this.gravarGUI = gravarGUI;
}

public boolean isGuiClose() {
    return guiClose;
}

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

public int getState1() {
    return state;
}

public void setState1(int state) {
    this.state = state;
}

public Semaphore getHaTrabalho() {
    return haTrabalho;
}

public void setHaTrabalho(Semaphore haTrabalho) {
    this.haTrabalho = haTrabalho;
}

public File getGravaFile() {
    return fileGravar;
}

public void setGravaFile(File gravaFile) {
    this.fileGravar = gravaFile;
}

public FileInputStream getInputStream() {
    return inputStream;
}

public void setInputStream(FileInputStream inputStream) {
    this.inputStream = inputStream;
}

public FileOutputStream getOutputStream() {
    return outputStream;
}

public void setOutputStream(FileOutputStream outputStream) {
    this.outputStream = outputStream;
}

public ObjectInputStream getObjectInput() {
    return objectInput;
}

public void setObjectInput(ObjectInputStream objectInput) {
    this.objectInput = objectInput;
}

public ObjectOutputStream getObjectOutput() {
    return objectOutput;
}

public void setObjectOutput(ObjectOutputStream objectOutput) {
    this.objectOutput = objectOutput;
}
```

```
}
```

Código GUI REI_SUBDITO:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP2 - Aula 4
//Coding UTF-08

//GUI REI_SUBDITO

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JCheckBox;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextArea;
import javax.swing.JButton;
import javax.swing.JScrollPane;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GUI_REI_SUBDITO extends JFrame{

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JPanel panel;
    private JPanel panel_log;
    private JCheckBox checkBoxRei;
    private JCheckBox checkBoxSubdito;
    private JCheckBox checkBoxGravar;
    private JLabel labelLog;
    private JButton buttonCleanLog;
    private JScrollPane scrollPane;
    private JTextArea logTextArea;
    private boolean selectRei;
    private boolean selectSubdito;
    private boolean selectGravar;

    private BufferCircular bCircular;

    private Rei rei;
    private Subdito subdito;
    private Gravar_Spy gravar_Spy;


    public static void main(String[] args) {
        GUI_REI_SUBDITO frame = new GUI_REI_SUBDITO();
    }

    public GUI_REI_SUBDITO() {

        this.selectRei = false;
        this.selectSubdito = false;
        this.selectGravar = false;
        bCircular = new BufferCircular(12);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 380);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        this.setTitle("TP2 - REI_SUBDITO");

        panel = new JPanel();
        panel.setBorder(new TitledBorder(null, "Ativar/Desativar Tarefa", TitledBorder.LEADING,
TitledBorder.TOP, null, null));
        panel.setBounds(10, 10, 416, 65);
        panel.setLayout(null);

        checkBoxRei = new JCheckBox("REI");
        checkBoxRei.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
```

```
                if (checkBoxRei.isSelected()) {
                    rei = new Rei(bCircular);
                    rei.start();
                    rei.getHaTrabalho().release();
                }
            }
        });
        checkBoxRei.setFont(new Font("Tahoma", Font.PLAIN, 14));
        checkBoxRei.setBounds(6, 24, 118, 21);
        panel.add(checkBoxRei);

        checkBoxSubdito = new JCheckBox("SUBDITO");
        checkBoxSubdito.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                if (checkBoxSubdito.isSelected()) {
                    subdito = new Subdito(bCircular);
                    subdito.start();
                }
            }
        });
        checkBoxSubdito.setFont(new Font("Tahoma", Font.PLAIN, 14));
        checkBoxSubdito.setBounds(149, 24, 118, 21);
        panel.add(checkBoxSubdito);

        checkBoxGravar = new JCheckBox("GRAVAR");
        checkBoxGravar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                if (checkBoxGravar.isSelected()) {
                    gravar_Spy = new Gravar_Spy(subdito.getRobot());
                    new Thread(gravar_Spy).start();
                    subdito.setRobot(gravar_Spy);
                }
            }
        });
        checkBoxGravar.setFont(new Font("Tahoma", Font.PLAIN, 14));
        checkBoxGravar.setBounds(314, 24, 96, 21);
        panel.add(checkBoxGravar);

        panel_log = new JPanel();
        panel_log.setBounds(10, 96, 416, 237);
        contentPane.add(panel_log);
        panel_log.setLayout(null);


        labelLog = new JLabel("Log");
        labelLog.setFont(new Font("Tahoma", Font.PLAIN, 14));
        labelLog.setBounds(10, 10, 48, 27);
        panel_log.add(labelLog);

        buttonCleanLog = new JButton("Limpar Log");
        buttonCleanLog.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                logTextArea.setText(null);
            }
        });
        buttonCleanLog.setFont(new Font("Tahoma", Font.PLAIN, 14));
        buttonCleanLog.setBounds(303, 10, 103, 27);
        panel_log.add(buttonCleanLog);

        contentPane.setLayout(null);
        contentPane.add(panel);
        contentPane.add(panel_log);

        scrollPane = new JScrollPane();
        scrollPane.setBounds(0, 47, 416, 180);
        panel_log.add(scrollPane);

        logTextArea = new JTextArea();
        logTextArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        scrollPane.setViewportView(logTextArea);

        setVisible(true);
    }

}
```

Código Interface Gravar_Spy:

```
//ISEL- LEIM - Miguel Alcobia, Fábio Pestana, João Ramos
//Fundamentos de Sistemas Operativos - TP2 - Aula 4
//Coding UTF-08

//Interface IGravar_Spy

public interface IGravar_Spy {
    final int CRIAR_FILE = 0;
    final int INATIVO = 0;
    final int ESPERAR_TRABALHO = 1;
    final int ESPERAR_ATIVIDADE = 2;
    final int GRAVAR_PARAR = 3;
    final int REPRODUZIR = 4;
    final int END_STATE = 5;

    final int CMD_RETA = 1;
    final int CMD_CDIREITA = 2;
    final int CMD_CESQUERDA = 3;
}
```