Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

Semestre de Inverno 2016/2017

Fundamentos de Sistemas Operativos

1º Trabalho Prático

Manual de funcionamento da Interface Gráfica do Robot Lego

Trabalho realizado pelo grupo 11:

Diogo Fernandes nº39205

Rui Santos nº39286

Hugo Silva nº40614

Objectivo

Desenvolvimento duma interface gráfica de utilizador em Java com Swing utilizando um editor de GUI. Contato com a API do robot didático Lego-Mindstorms.

Observações

Uma das bibliotecas utilizadas é a biblioteca RobotLego fornecida pela disciplina de FSO,

que fornece uma maneira simples de aceder ao hardware do robot.

Lista dos comandos mais importante:

```
robot.OpenNXT(String nomeDoRobot);
```

Abre a conexão com o Robot.

```
robot.CloseNXT();
```

Fecha a conexão com o Robot.

```
robot.Reta(int distancia);
```

Robot avança *distancia* cm para a frente ou para trás caso a *distancia* seja negativa.

```
robot.CurvarDireita(int raio, int angulo);
```

```
robot.CurvarEsquerda(int raio, int angulo);
```

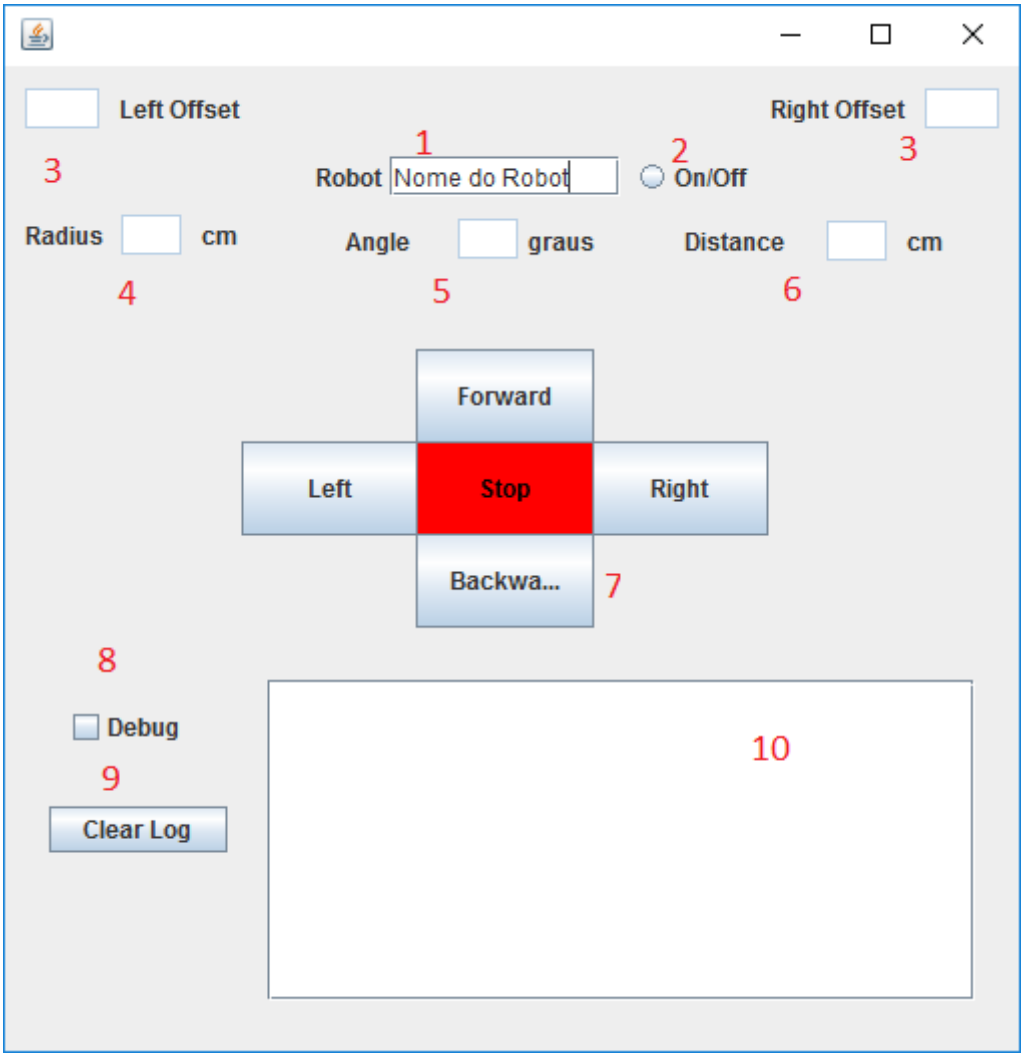
```
robot.Parar(boolean parar);
```

Neste caso o Parar tem duas funcionalidades muito importantes, que são definidas pelo valor do boolean, caso este esteja a falso o robot efetua a ação que estava a fazer até ao final e então sim o Robot para, caso este esteja a verdadeiro o Robot para de imediato cancelando todas as ações que tinha agendadas.

```
robot.AjustarVME(int leftOffsetValue);
```

```
robot.AjustarVMD(int leftOffsetValue);
```

Interface Gráfica do Comando do Robot Lego



1 – Nome do Robot: Introduzir o nome do Robot que pretende utilizar.

2 – On/Off: Radiobutton, que caso o nome do robot esteja correto conecta o comando com o Robot.

```
private void connectToRobot(){

        if(this.radioState==false){
                boolean auxEstado;
                auxEstado = this.robot.OpenNXT(this.robotName);
                if(auxEstado == false){
                        this.rdbtnOnoff.setSelected(false);
                        showMessages("Erro ao abrir o Robot: " + this.robotName);
                        this.radioState = false;
                        robotON(this.radioState);
                }else{
                        this.rdbtnOnoff.setSelected(true);
                        showMessages("Robot ligado : " + this.robotName);
                        this.radioState = true;
                        robotON(this.radioState);
                }
        }else{
                this.robot.CloseNXT();
                this.radioState = false;
                this.rdbtnOnoff.setSelected(false);
                robotON(this.radioState);
```

```
                showMessages("Robot Desligado: " + this.robotName);
        }
```

3 – Steering Offset's: Ajusta a rotação dos motores do robot.

```
private void steeringLeft(){
        this.robot.AjustarVME(this.leftOffsetValue);
}

private void steeringRight(){
        this.robot.AjustarVMD(this.rightOffsetValue);
}
```

4 – Radius: Determina um distancia em centímetros para o raio de circunferência que o robot descreve ao efetuar uma curva.

5 – Angle: Determina a amplitude a que o Robot efetua a curva com base no angulo em graus.

6 – Distance: Determina a distancia que o Robot percorre ao efetuar um movimento.

7 – Arrow Keys: Botões que determinam as ações do Robot (frente, trás, esquerda, direita e stop).

```
private void actionForward() {
        // TODO Auto-generated method stub
        try {
                this.robot.Reta(this.distance);
                this.robot.Parar(false);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                showMessages("Robot nao disponivel: " + e.getMessage());
        }
}

private void actionBackwards() {
        try{
                this.robot.Reta(-this.distance);
                this.robot.Parar(false);
        }catch(Exception e){
                showMessages("Robot nao disponivel: " + e.getMessage());
        }
}

private void actionRight(){
        try{
                this.robot.CurvarDireita(this.radius, this.angle);
                this.robot.Parar(false);
        }catch(Exception e){
                showMessages("Robot nao disponivel: " + e.getMessage());
        }
}

private void actionLeft(){
        try {
                this.robot.CurvarEsquerda(this.radius, this.angle);
                this.robot.Parar(false);
        } catch (Exception e) {
                showMessages("Robot nao disponivel: " + e.getMessage());
        }
}

private void actionStop(){
        try {
                this.robot.Parar(true);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                showMessages("Robot nao disponivel: " + e.getMessage());
        }
}
```

8 – Debug: Radiobutton que ativa a consola do comando do Robot.

9 – Clear Log: Botão que limpa a consola do comando do Robot.

```
private void clearLog(){
        this.debugText.setText("");
}
```

10 – Consola: Textfield onde todas as ações do robot são registadas.

Anexos

Código Integral da interface do Robot:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
import RobotLego.RobotLego;

public class RobotInterface extends JFrame {
	private static final long serialVersionUID = 1L;
	private JPanel contentPane;
	private JTextField robotNameText;
	private JRadioButton rdbtnOnoff;
	private JCheckBox chckbxDebug;
	private JTextField leftOffset;
	private JTextField radiusText;
	private JTextField angleText;
	private JTextField distanceText;
	private JTextField rightOffset;
	private JTextArea debugText;
	private JButton btnClear;
	private RobotLego robot;
	private String robotName;
	private boolean radioState;
	private boolean debugOnOff;
	private int rightOffsetValue;
	private int leftOffsetValue;
	private int radius;
	private int angle;
```

```
private int distance;


private void myInit(){
          this.radioState = false;
          this.debugOnOff = false;
          this.radius = 0;
          this.angle = 0;
          this.distance = 0;
          this.rightOffsetValue = 0;
          this.leftOffsetValue = 0;
          this.robotName = "Nome do Robot";
          this.robot = new RobotLego();


          this.rdbtnOnoff.setSelected(this.radioState);
          this.chckbxDebug.setSelected(this.debugOnOff);
          this.rightOffset.setEnabled(false);
          this.leftOffset.setEnabled(false);
          this.distanceText.setEnabled(false);
          this.angleText.setEnabled(false);
          this.radiusText.setEnabled(false);
          this.robotNameText.setText(robotName);
}


private void robotON(boolean condition){
          if(condition){
                    this.rightOffset.setEnabled(true);
                    this.leftOffset.setEnabled(true);
                    this.distanceText.setEnabled(true);
                    this.angleText.setEnabled(true);
                    this.radiusText.setEnabled(true);
          }else{
                    this.rightOffset.setEnabled(false);
                    this.leftOffset.setEnabled(false);
                    this.distanceText.setEnabled(false);
                    this.angleText.setEnabled(false);
                    this.radiusText.setEnabled(false);
          }
}
private void connectToRobot(){
          if(this.radioState==false){
                    boolean auxEstado;
                    auxEstado = this.robot.OpenNXT(this.robotName);
```

```
                    if(auxEstado == false){
                              this.rdbtnOnoff.setSelected(false);
                              showMessages("Erro ao abrir o Robot: " + this.robotName);
                              this.radioState = false;
                              robotON(this.radioState);
                    }else{
                              this.rdbtnOnoff.setSelected(true);
                              showMessages("Robot ligado : " + this.robotName);
                              this.radioState = true;
                              robotON(this.radioState);
                    }
          }else{
                    this.robot.CloseNXT();
                    this.radioState = false;
                    this.rdbtnOnoff.setSelected(false);
                    robotON(this.radioState);
                    showMessages("Robot Desligado: " + this.robotName);
          }
}
private void showMessages(String message){
          if(this.debugOnOff){
                    this.debugText.append(message + "\n");
          }else{
                    this.debugText.append("");
          }
}
private void setDistance(String distance){
          try{
                    this.distance = Integer.parseInt(distance);
          }catch(Exception e){
                    showMessages("Erro seguinte: " + e.getMessage());
          }
}
private void setRadius(String radius){
          try{
                    this.radius = Integer.parseInt(radius);
          }catch(Exception e){
                    showMessages("Erro seguinte: " + e.getMessage());
          }
}


private void setRightOffset(String offset){
```

```
		try{
			this.rightOffsetValue = Integer.parseInt(offset);
		}catch(Exception e){
			showMessages("Erro seguinte: " + e.getMessage());
		}
	}

	private void setLeftOffset(String offset){
		try{
			this.leftOffsetValue = Integer.parseInt(offset);
		}catch(Exception e){
			showMessages("Erro seguinte: " + e.getMessage());
		}
	}

	private void setAngle(String angle){
		try{
			this.angle = Integer.parseInt(angle);
		}catch(Exception e){
			showMessages("Erro seguinte: " + e.getMessage());
		}
	}

	private void setRobotName(String name){
		try{
			this.robotName = name;
		}catch(Exception e){
			showMessages("Erro seguinte: " + e.getMessage());
		}
	}
	private void clearLog(){
		this.debugText.setText("");
	}
	private void actionForward() {
		try {
			this.robot.Reta(this.distance);
			this.robot.Parar(false);
		} catch (Exception e) {
			showMessages("Robot nao disponivel: " + e.getMessage());
		}
	}
	private void actionBackwards() {
```

```
		try{
			this.robot.Reta(-this.distance);
			this.robot.Parar(false);
		}catch(Exception e){
			showMessages("Robot nao disponivel: " + e.getMessage());
		}
	}
	private void actionRight(){
		try{
			this.robot.CurvarDireita(this.radius, this.angle);
			this.robot.Parar(false);
		}catch(Exception e){
			showMessages("Robot nao disponivel: " + e.getMessage());
		}
	}
	private void actionLeft(){
		try {
			this.robot.CurvarEsquerda(this.radius, this.angle);
			this.robot.Parar(false);
		} catch (Exception e) {
			showMessages("Robot nao disponivel: " + e.getMessage());
		}
	}
	private void actionStop(){
		try {
			this.robot.Parar(true);
		} catch (Exception e) {
			showMessages("Robot nao disponivel: " + e.getMessage());
		}
	}

	private void steeringLeft(){
		this.robot.AjustarVME(this.leftOffsetValue);
	}

	private void steeringRight(){
		this.robot.AjustarVMD(this.rightOffsetValue);
	}



	public static void main(String[] args) {
```

```
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					new RobotInterface();

				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	public RobotInterface() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 522, 530);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		rightOffset = new JTextField();
		rightOffset.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				setRightOffset(rightOffset.getText());
				showMessages("rightOffset -> " + rightOffsetValue);
			}
		});
		rightOffset.setBounds(459, 11, 37, 20);
		contentPane.add(rightOffset);
		rightOffset.setColumns(10);

		leftOffset = new JTextField();
		leftOffset.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				setLeftOffset(leftOffset.getText());
				showMessages("Left offset -> " + leftOffsetValue);
			}
		});
		leftOffset.setColumns(10);
		leftOffset.setBounds(10, 11, 37, 20);
		contentPane.add(leftOffset);

		JLabel lblLeftOffset = new JLabel("Left Offset");
```

```
lblLeftOffset.setBounds(57, 14, 70, 14);
contentPane.add(lblLeftOffset);

JLabel lblRightOffset = new JLabel("Right Offset");
lblRightOffset.setBounds(382, 14, 70, 14);
contentPane.add(lblRightOffset);

robotNameText = new JTextField();
robotNameText.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		setRobotName(robotNameText.getText());
		showMessages("Nome do robot -> " + robotName);
	}
});
robotNameText.setBounds(192, 45, 115, 20);
contentPane.add(robotNameText);
robotNameText.setColumns(10);

JLabel lblRobot = new JLabel("Robot");
lblRobot.setBounds(155, 48, 46, 14);
contentPane.add(lblRobot);

rdbtnOnoff = new JRadioButton("On/Off");
rdbtnOnoff.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		connectToRobot();
	}
});
rdbtnOnoff.setBounds(313, 44, 109, 23);
contentPane.add(rdbtnOnoff);

JLabel lblRaio = new JLabel("Radius");
lblRaio.setBounds(10, 77, 46, 14);
contentPane.add(lblRaio);

radiusText = new JTextField();
radiusText.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		setRadius(radiusText.getText());
		showMessages("Radius -> " + radius);
	}
});
```

```
radiusText.setBounds(58, 74, 30, 20);
contentPane.add(radiusText);
radiusText.setColumns(10);

JLabel lblCm = new JLabel("cm");
lblCm.setBounds(98, 77, 46, 14);
contentPane.add(lblCm);

JLabel lblAngulo = new JLabel("Angle");
lblAngulo.setBounds(170, 80, 46, 14);
contentPane.add(lblAngulo);

angleText = new JTextField();
angleText.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		setAngle(angleText.getText());
		showMessages("Angle -> " + angle);
	}
});
angleText.setColumns(10);
angleText.setBounds(226, 76, 30, 20);
contentPane.add(angleText);

JLabel lblGraus = new JLabel("graus");
lblGraus.setBounds(261, 80, 46, 14);
contentPane.add(lblGraus);

JLabel lblDistancia = new JLabel("Distance");
lblDistancia.setBounds(339, 80, 56, 14);
contentPane.add(lblDistancia);

distanceText = new JTextField();
distanceText.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		setDistance(distanceText.getText());
		showMessages("Distance -> " + distance);
	}
});
distanceText.setColumns(10);
distanceText.setBounds(410, 77, 30, 20);
contentPane.add(distanceText);
```

```
JLabel lblCm_1 = new JLabel("cm");
lblCm_1.setBounds(450, 80, 46, 14);
contentPane.add(lblCm_1);

JButton btnNewButton = new JButton("Forward");
btnNewButton.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		steeringLeft();
		steeringRight();
		actionForward();
		showMessages("Forward -> " + distance + "cm.");
	}

});
btnNewButton.setBounds(205, 141, 89, 47);
contentPane.add(btnNewButton);

JButton btnStop = new JButton("Stop");
btnStop.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		actionStop();
		showMessages("STOP!");
	}
});
btnStop.setBackground(Color.RED);
btnStop.setForeground(Color.BLACK);
btnStop.setBounds(205, 187, 89, 47);
contentPane.add(btnStop);

JButton btnRight = new JButton("Right");
btnRight.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		steeringLeft();
		steeringRight();
		actionRight();
		showMessages("Right -> " + distance + "cm.");
	}
});
btnRight.setBounds(292, 187, 89, 47);
contentPane.add(btnRight);

JButton btnLeft = new JButton("Left");
```

```
btnLeft.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		steeringLeft();
		steeringRight();
		actionLeft();
		showMessages("Left -> " + distance + "cm.");
	}
});
btnLeft.setBounds(118, 187, 89, 47);
contentPane.add(btnLeft);

JButton btnBackwards = new JButton("Backwards");
btnBackwards.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		steeringLeft();
		steeringRight();
		actionBackwards();
		showMessages("Backwards -> " + distance + "cm." );
	}
});
btnBackwards.setBounds(205, 233, 89, 47);
contentPane.add(btnBackwards);

chckbxDebug = new JCheckBox("Debug");
chckbxDebug.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		debugOnOff = !debugOnOff;
		if(debugOnOff){
			showMessages("Debug Ativo!");
		}
	}
});
chckbxDebug.setBounds(30, 318, 81, 23);
contentPane.add(chckbxDebug);

JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(131, 306, 353, 160);
contentPane.add(scrollPane);

debugText = new JTextArea();
debugText.setForeground(Color.BLACK);
scrollPane.setViewportView(debugText);
```

```
                btnClear = new JButton("Clear Log");
                btnClear.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                clearLog();
                        }
                });
                btnClear.setBounds(22, 369, 89, 23);
                contentPane.add(btnClear);


                this.setVisible(true);


                myInit();
        }
}
```