ADEETC

Licenciatura em Engenharia

Informática e Multimédia

1º Semestre 16/17 SI

Fundamentos de Sistemas Operativos

Relatório do Trabalho nº 4

Eng. Carlos Gonçalves

Grupo 11

Trabalho realizado por:

Rui Santos nº39286

Diogo Fernandes nº39205

Hugo Safara nº40614

**Índice**

[**Introdução** 2](#_Toc475564458)

[**Desenvolvimento** 3](#_Toc475564459)

[**Diagrama Classes** 3](#_Toc475564460)

[**RobotPlayer** 5](#_Toc475564461)

[**MyRobot** 7](#_Toc475564462)

[**GUIplayer** 8](#_Toc475564463)

[**Robot Interface** 9](#_Toc475564464)

[**Conclusão** 11](#_Toc475564465)

[**Bibliografia** 11](#_Toc475564466)

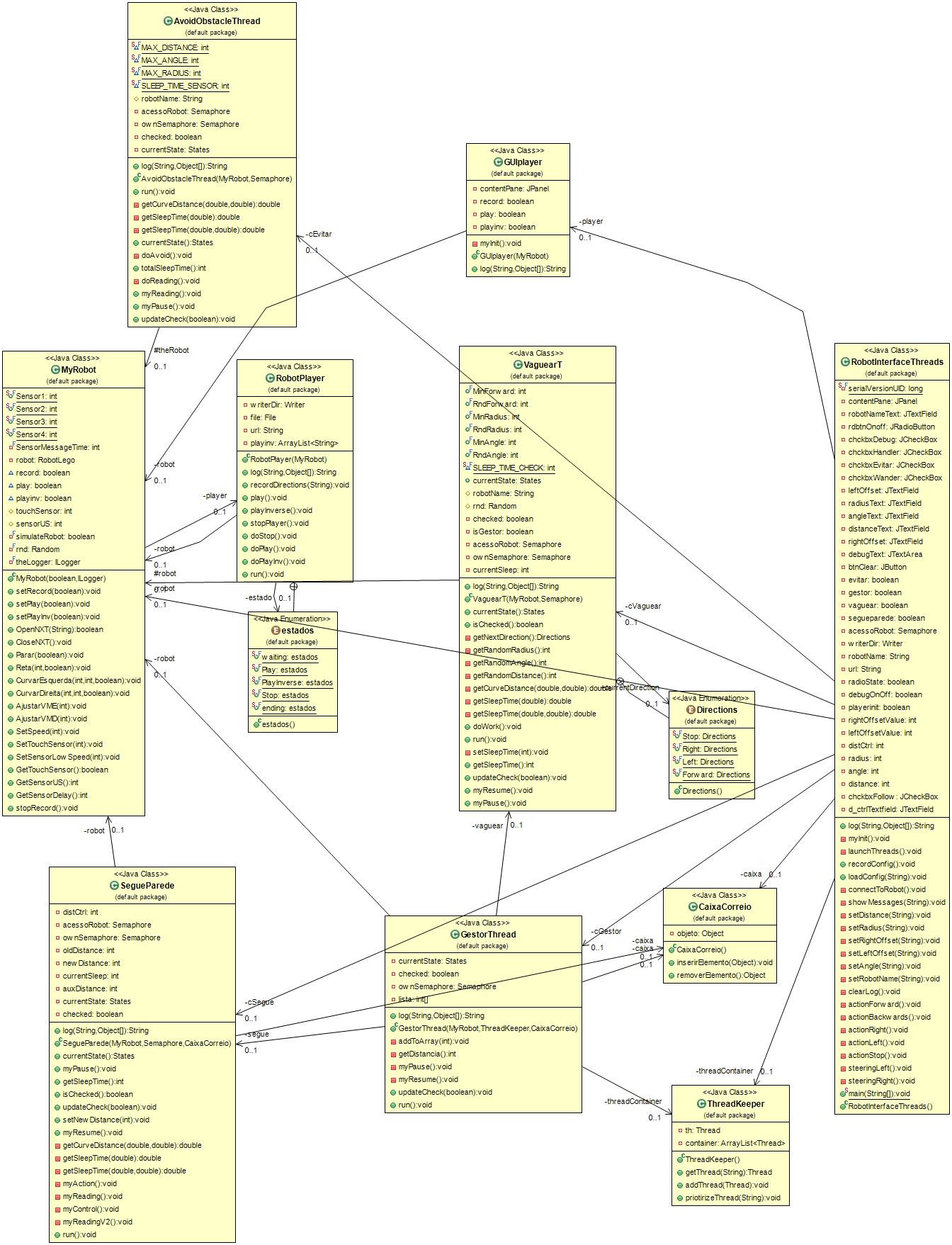
# **Introdução**

Este trabalho tem como objetivo, manipulação de ficheiros numa aplicação multiprocesso. Implementámos então numa GUI separada da GUI do Robot, um *player* com as funcionalidades de gravar a trajetória do robot, quer seja por comandos individuais ou estando o robot a *vaguear,* a reprodução regular do ficheiro que foi gravado, a reprodução inversa desse mesmo ficheiro e o *stop.*

Implementámos também, agora na GUI do robot, uma opção de gravar um ficheiro com as configurações do robot, que são carregadas automaticamente quando se escreve o nome do robot, caso já exista um ficheiro com o seu nome.

# **Desenvolvimento**

## **Diagrama Classes**



## **RobotPlayer**

Neste processo, estão os métodos de gravar as direções, o reproduzir, o reproduzir ao contrário e o stop.

**public** **void** **recordDirections**(**String** data){

**try** {

writerDir.write(data + "\n");

} **catch** (**IOException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**public** **void** **play**() **throws** **FileNotFoundException**{

**Scanner** **sc**= **new** Scanner(file);

**String** **theString** = sc.nextLine();

**while** (sc.hasNextLine()) {

theString = theString + "\n" + sc.nextLine();

}

**String**[] **aux** = theString.split("\n");

**for**(**int** **i** = 0; i < aux.length; i++){

**if**(aux[i].contains("reta")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

**int** **y** = **Integer**.*parseInt*(x);

log("reta: "+y);

robot.Reta(y,**false**);

}

**if**(aux[i].contains("curvardireita")){

**String** **raio** = aux[i].substring(aux[i].indexOf(':') + 1, aux[i].indexOf(','));

**String** **angulo** = aux[i].substring(aux[i].indexOf(',') + 1);

**int** **r** = **Integer**.*parseInt*(raio);

**int** **a** = **Integer**.*parseInt*(angulo);

log("curvardireita: raio ="+r+" angulo ="+a);

robot.CurvarDireita(r, a,**false**);

}

**if**(aux[i].contains("curvaresquerda")){

**String** **raio** = aux[i].substring(aux[i].indexOf(':') + 1, aux[i].indexOf(','));

**String** **angulo** = aux[i].substring(aux[i].indexOf(',') + 1);

**int** **r** = **Integer**.*parseInt*(raio);

**int** **a** = **Integer**.*parseInt*(angulo);

log("curvaresquerda: raio ="+r+" angulo ="+a);

robot.CurvarEsquerda(r, a,**false**);

}

**if**(aux[i].contains("parar")){

**String** **y** = aux[i].substring(aux[i].indexOf(':') + 1);

**boolean** **x** = **Boolean**.*parseBoolean*(y);

log("STOP: "+x);

robot.Parar(x);

**try** {

**Thread**.*sleep*(1000);

} **catch** (**InterruptedException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

sc.close();

}

**public** **void** **playInverse**() **throws** **FileNotFoundException**{

**Scanner** **sc**= **new** Scanner(file);

**String** **theString** = sc.nextLine();

**while** (sc.hasNextLine()) {

theString = theString + "\n" + sc.nextLine();

}

**String**[] **aux** = theString.split("\n");

**for**(**int** **i** = aux.length-1; i > -1; i--){

**if**(aux[i].contains("reta")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

**int** **y** = **Integer**.*parseInt*(x);

log("reta: "+y);

robot.Reta(-y,**false**);

}

**if**(aux[i].contains("curvardireita")){

**String** **raio** = aux[i].substring(aux[i].indexOf(':') + 1, aux[i].indexOf(','));

**String** **angulo** = aux[i].substring(aux[i].indexOf(',') + 1);

**int** **r** = **Integer**.*parseInt*(raio);

**int** **a** = **Integer**.*parseInt*(angulo);

log("curvardireita: raio ="+r+" angulo ="+a);

robot.CurvarEsquerda(r, a,**false**);

}

**if**(aux[i].contains("curvaresquerda")){

**String** **raio** = aux[i].substring(aux[i].indexOf(':') + 1, aux[i].indexOf(','));

**String** **angulo** = aux[i].substring(aux[i].indexOf(',') + 1);

**int** **r** = **Integer**.*parseInt*(raio);

**int** **a** = **Integer**.*parseInt*(angulo);

log("curvaresquerda: raio ="+r+" angulo ="+a);

robot.CurvarDireita(r, a,**false**); }

**if**(aux[i].contains("parar")){

**String** **y** = aux[i].substring(aux[i].indexOf(':') + 1);

**boolean** **x** = **Boolean**.*parseBoolean*(y);

log("STOP: "+x);

robot.Parar(x);

**try** {

**Thread**.*sleep*(1000);

} **catch** (**InterruptedException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**this**.robot.Parar(**false**);

sc.close();

}

**public** **void** **stopPlayer**(){

**try** {

writerDir.close();

} **catch** (**IOException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

Maquina de estados do RobotPlayer:

**public** **void** **run**() {

**while**(estado != *estados*.***ending***){

**switch** (estado) {

**case** ***waiting***:

//log("Case: waiting");

**break**;

**case** ***Play***:

log("Case: Play");

**try** {

play();

} **catch** (**FileNotFoundException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

estado = *estados*.***waiting***;

**break**;

**case** ***PlayInverse***:

log("Case: PlayInverse");

**try** {

playInverse();

} **catch** (**FileNotFoundException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

estado = *estados*.***waiting***;

**break**;

**case** ***Stop***:

log("Case: Stop");

stopPlayer();

estado = *estados*.***waiting***;

**break**;

**default**:

**break**; }

}

}

## **MyRobot**

Como o gravar direções não pode gravar os comandos do evitar obstáculos, nos comandos do MyRobot, como o *reta(int distance), o curvarDireita(int radius, int angle) e o curvarDireita(int radius, int angle)* , fizemos uma alteração e acrescentámos um parâmetro *boolean* que é true caso seja comandos do evitar obstáculos.

**public** **void** **Reta**(**int** distancia, **boolean** chkavoid) {

**if** ( **this**.simulateRobot==**false**) {

**this**.robot.Reta( distancia );

**if** (!chkavoid && record) {

player.recordDirections("reta:" + distancia);

}

}

**else** {

**this**.theLogger.log( "Reta(distancia->%d)", distancia );

**if** (!chkavoid && record) {

player.recordDirections("reta:" + distancia);

}

}

}

**public** **void** **CurvarEsquerda**(**int** raio, **int** angulo, **boolean** chkavoid) {

**if** ( **this**.simulateRobot==**false**) {

**this**.robot.CurvarEsquerda(raio, angulo);

**if**(!chkavoid && record){

player.recordDirections("curvaresquerda:"+raio+','+angulo);

}

}

**else** {

**this**.theLogger.log( "CurvarEsquerda(raio->%d, angulo->%d)", raio, angulo );

**if**(!chkavoid && record){ player.recordDirections("curvaresquerda:"+raio+','+angulo);

}

}

}

**public** **void** **CurvarDireita**(**int** raio, **int** angulo, **boolean** chkavoid) {

**if** ( **this**.simulateRobot==**false**) {

**this**.robot.CurvarDireita(raio, angulo);

**if**(!chkavoid && record){

player.recordDirections("curvardireita:"+raio+','+angulo);

}

}

**else** {

**this**.theLogger.log( "CurvarDireita(raio->%d, angulo->%d)", raio, angulo );

**if**(!chkavoid && record){

player.recordDirections("curvardireita:"+raio+','+angulo);

}

}

}

## **GUIplayer**

É a interface do *player,* simples apenas com os 4 botões o *record, play, playInverse, stop.*

**public** **GUIplayer**(**MyRobot** robot) {

**this**.robot = robot;

setDefaultCloseOperation(**JFrame**.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 142, 221);

contentPane = **new** JPanel();

contentPane.setBorder(**new** EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(**null**);

**JButton** **btnRecord** = **new** JButton("RECORD");

btnRecord.addActionListener(**new** ActionListener() {

**public** **void** **actionPerformed**(**ActionEvent** arg0) {

**if**(!record){

record = !record;

robot.setRecord(record);

}

}

});

btnRecord.setBounds(20, 51, 89, 23);

contentPane.add(btnRecord);

**JButton** **btnPlay** = **new** JButton("PLAY");

btnPlay.addActionListener(**new** ActionListener() {

**public** **void** **actionPerformed**(**ActionEvent** e) {

**if**(!play){

play = !play;

robot.setPlay(play);

}

}

});

btnPlay.setBounds(20, 85, 89, 23);

contentPane.add(btnPlay);

**JButton** **btnPlayInv** = **new** JButton("PLAY INV");

btnPlayInv.addActionListener(**new** ActionListener() {

**public** **void** **actionPerformed**(**ActionEvent** e) {

**if**(!playinv){

playinv = !playinv;

robot.setPlayInv(playinv);

}

}

});

btnPlayInv.setBounds(20, 119, 89, 23);

contentPane.add(btnPlayInv);

**JButton** **btnStop** = **new** JButton("STOP");

btnStop.addActionListener(**new** ActionListener() {

**public** **void** **actionPerformed**(**ActionEvent** e) {

**if**(record){

record = !record;

robot.setRecord(record);

robot.stopRecord();

}

**if**(play){

play = !play;

robot.setPlay(play);

}

**if**(playinv){

playinv = !playinv;

robot.setPlayInv(playinv);

}

}

});

btnStop.setBounds(20, 148, 89, 23);

contentPane.add(btnStop);

**JLabel** **lblPlayer** = **new** JLabel("PLAYER");

lblPlayer.setBounds(50, 11, 46, 14);

contentPane.add(lblPlayer);

myInit();

}

## **Robot Interface**

À classe RobotInterface, acrescentámos os métodos de gravar e carregar as configurações do robot.

**public** **void** **recordConfig**() **throws** **IOException**{

**File** **file** = **new** File(url+"config"+robotName+".txt");

**try** {

writerDir = **new** BufferedWriter(**new** OutputStreamWriter(**new** FileOutputStream(file), "utf-8"));

} **catch** (**IOException** **ex**) {

// report

}

writerDir.write("steeringleft:"+leftOffsetValue + "\n");

writerDir.write("steeringright:"+rightOffsetValue + "\n");

writerDir.write("distCtrl:"+distCtrl + "\n");

writerDir.write("debugbool:"+debugOnOff + "\n");

writerDir.write("distance:"+distance + "\n");

writerDir.write("angle:"+angle + "\n");

writerDir.write("radius:"+radius + "\n");

writerDir.close();

}

**public** **void** **loadConfig**(**String** robotName) **throws** **FileNotFoundException**{

**File** **file** = **new** File(url+"config"+robotName+".txt");

**if**(file.exists()){

**Scanner** **sc** = **new** Scanner(file);

**String** **theString** = sc.nextLine();

**while** (sc.hasNextLine()) {

theString = theString + "\n" + sc.nextLine();

}

**String**[] **aux** = theString.split("\n");

**for**(**int** **i** = 0; i < aux.length; i++){

**if**(aux[i].contains("steeringleft")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

setLeftOffset(x);

leftOffset.setText(x);

}

**if**(aux[i].contains("steeringright")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

setRightOffset(x);

rightOffset.setText(x);

}

**if**(aux[i].contains("distCtrl")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

setDistCtrl(x);

d\_ctrlTextfield.setText(x);

}

**if**(aux[i].contains("debugbool")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

**boolean** **y** = **Boolean**.*parseBoolean*(x);

debugOnOff= y;

chckbxDebug.setSelected(y);

}

**if**(aux[i].contains("distance")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

setDistance(x);

distanceText.setText(x);

}

**if**(aux[i].contains("angle")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

setAngle(x);

angleText.setText(x);

}

**if**(aux[i].contains("radius")){

**String** **x** = aux[i].substring(aux[i].indexOf(':') + 1);

setRadius(x);

radiusText.setText(x);

}

}

sc.close();

}

}

E o botão que lança o player e o botão de gravar as configurações.

**JButton** **btnInitPlayer** = **new** JButton("Init Player");

btnInitPlayer.addActionListener(**new** ActionListener() {

**public** **void** **actionPerformed**(**ActionEvent** arg0) {

playerinit = !playerinit;

**if**(playerinit){

player = **new** GUIplayer(robot);

player.setVisible(**true**);

}

}

});

btnInitPlayer.setBounds(10, 198, 134, 23);

contentPane.add(btnInitPlayer);

**JButton** **btnRecordConfig** = **new** JButton("Record Config");

btnRecordConfig.addActionListener(**new** ActionListener() {

**public** **void** **actionPerformed**(**ActionEvent** arg0) {

**try** {

recordConfig();

} **catch** (**IOException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

});

btnRecordConfig.setBounds(10, 164, 134, 23);

contentPane.add(btnRecordConfig);

# **Conclusão**

Tivemos alguns problemas na parte de escrever para o ficheiro, começámos por usar o *PrintWriter ,* mas depois concluímos que não era o mais apropriado e passámos a usar o *BufferedWriter.* Tomámos a decisão de usar a class MyRobot para gravar as direções do robot porque como todas as classes utilizam o MyRobot para darem as instruções ao robot, assim a implementação do gravar seria muito mais facilitada, porque independentemente das classes que acrescentarmos o gravar continua habilitado a gravar as instruções provenientes dessas classes. Conseguimos completar com êxito todos os pontos do trabalho.

# **Bibliografia**

Folhas da disciplina de FSO (Fundamentos de Sistemas Operativos):

<https://moodle.isel.pt/pluginfile.php?file=%2F692732%2Fmod_resource%2Fcontent%2F1%2FFundamentos%20de%20Sistemas%20Operativos.pdf>

Documentação java fornecida pela oracle:

<https://docs.oracle.com/javase/7/docs/api/overview-summary.html>