

**I**nstituto **S**uperior de **E**ngenharia de **L**isboa

Licenciatura em Engenharia Informática e Multimédia

Semestre de Inverno 2018/2019

Fundamentos de Sistemas Operativos

**Trabalho Prático 2 – TP2**

Trabalho elaborado por:

Fábio Alexandre da Cruz Silva Dias, A42921

Jorge Miguel Coelho Silva, A44615

Docente: Carlos Gonçalves

2018/11/28

Índice

[Indices de Figuras 4](#_Toc531544327)

[1. Introdução 6](#_Toc531544328)

[1.1. O que são *Tarefas* e onde se enquadra neste projeto. 7](#_Toc531544329)

[1.2. Como são implementados estas *Tarefas*? 7](#_Toc531544330)

[1.3. Como se garante “prioridade”? 7](#_Toc531544331)

[1.4. O que é *Exclusão Mútua*? 7](#_Toc531544332)

[*1.5.* *Semáforos.* 7](#_Toc531544333)

[2. Diagrama de Atividades 8](#_Toc531544334)

# Indices de Figuras

[Figura 1 - Interface Gráfica (GUI) 6](#_Toc531544335)

[Figura 2 - Diagrama de Atividades (Geral) 8](#_Toc531544336)

# Introdução

Pretende-se o desenvolvimento de uma aplicação para controlo de um robot constituído por 3 **tarefas** **JAVA** que permitam aplicar comportamentos a um robot. Esses comportamentos são então o **Vaguear**, **Evitar Obstáculo** e o **Fugir**.

A comunicação entre estas **tarefas** é suportada através de objetos partilhados e a sincronização utilizando **semáforos** ou **monitores**.

Para a execução deste trabalho, é requisitado que seja modificada/criada uma interface gráfica (**GUI**) com opções, tipo *JCheckBox*[[1]](#footnote-1), para o **Vaguear**, **Evitar** e **Fugir**. Ao ativar esta *CheckBox*, as **tarefas** têm de ser *ativadas* ou *desativadas* de forma independente.

Na próxima figura *(Figura 1)* é apresentada as modificações visuais a nossa Interface Gráfica.

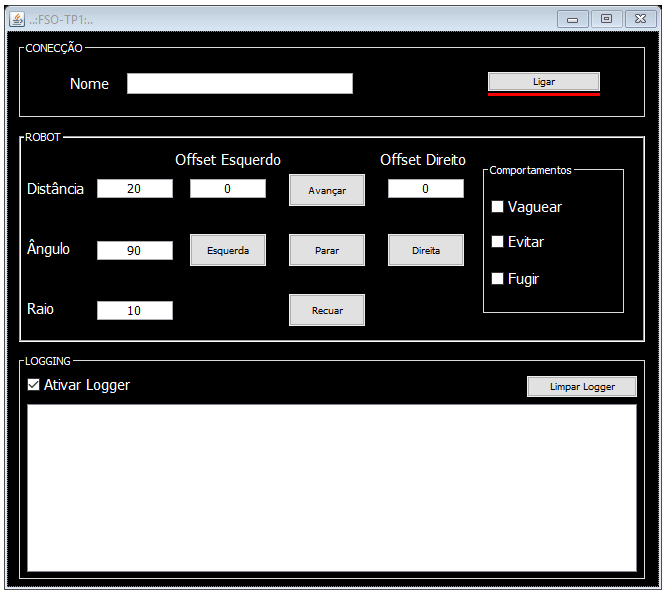


Figura 1 - Interface Gráfica (GUI)

## O que são *Tarefas* e onde se enquadra neste projeto.

*“Uma tarefa é uma atividade ou um trabalho que se têm de executar.”*

Neste projeto é pedido que todos os comportamentos sejam uma tarefa, assim todos os nosso comportamentos terão de seguir algumas “regras” para poderem executar, um dado algoritmo, sem que exista qualquer tipo de interferência com um outro qualquer comportamento.

## Como são implementados estas *Tarefas*?

Cada comportamento que tenha de ser modelado será uma extensão da classe *Thread*[[2]](#footnote-2), isto para, que exista uma certa “prioridade” para cada um dos comportamentos e que se garanta acesso exclusivo ao *recurso*.

## Como se garante “prioridade”?

Para garantir esta “prioridade” é criado um novo conceito de Sincronização entre Tarefas, isto é, cada tarefa vai ser executada e vai obter acesso ao recurso sem que mais nenhuma esteja lá, ou seja, *Exclusão Mútua.*

## O que é *Exclusão Mútua*?

Entende-se por *Exclusão Mútua*, o acesso exclusivo a um *recurso*. Para que isto seja possível, têm de se criar condições que garantam que cada *Tarefa* saiba se pode ou não pode aceder ao *recurso*. Este tipo de acesso pode ser feito com ***Semáforos e/ou Monitores***.

## *Semáforos.*

Um *Semáforo*[[3]](#footnote-3) controla o acesso a um *recurso* que seja partilhado por meio de um contador. Se esse contador for superior a 0, significa que existe acesso a esse recurso, caso não seja superior significa que não existe acesso a esse *recurso*. No caso do contador ser superior a 0, para se obter Exclusão desse semáforo é preciso fazer *semaforo.aquire()*, no qual vai decrementar uma unidade do contador. Quando já não for preciso ter Exclusão é só fazer *semaforo.release()*.

# Diagrama de Atividades

## *Diagrama de Atividades - Geral.*

No seguinte quadro está um Diagrama de Atividades geral do projeto.

Interface Gráfica

( GUI )

ROBOT

Comportamentos

*java.lang.Thread*

{Extende}

{Extende}

*Vaguear*

*Evitar*

*Fugir*

{Extende}

Figura - Diagrama de Atividades (Geral)

1. - *JCheckBox* – é um componente da biblioteca *java.swing* para interfaces gráficas. Estes componentes têm dois estados, *marcada* (verificada) e *desmarcada* (não verificada). [↑](#footnote-ref-1)
2. - A classe *Thread* é acedida através da biblioteca *java.lang.Thread*. [↑](#footnote-ref-2)
3. - ou *Semaphore* é acedida através da biblioteca *java.util.concurrent.Semaphore* [↑](#footnote-ref-3)