

**Projeto de Fundamentos de Programação**

Documentação técnica

Grupo Nº 19

| 109318 | Miguel Dinis |
| --- | --- |
|  | miguel.g.dinis@tecnico.ulisboa.pt |
| 107203 | Hugo Silva |
|  | hugo.costa.silva@tecnico.ulisboa.pt |

Licenciatura em Engenharia Mecânica

Licenciatura em Engenharia e Arquitetura Naval

Lisboa, Maio de 2024

**Índice**

[1.](#_3znysh7) Documentação técnica 3

[1.1.](#_2et92p0) Arquitetura do programa 3

[1.2.](#_tyjcwt) Projeto detalhado 4

1.2.1 Módulo Main 4

1.2.2 Módulo Tier 1 (Implementação 1) 5

1.2.3 Módulo Tier 2 (Implementação 2) 5

1.2.4 Módulo Waiter 5

1.2.5 Módulo Tables 6

[1.3.](#_3dy6vkm) Módulo Nome do Módulo 7

1.3.1 Entradas (aplicável para funções) 7

1.3.2 Saídas (aplicável para funções) 7

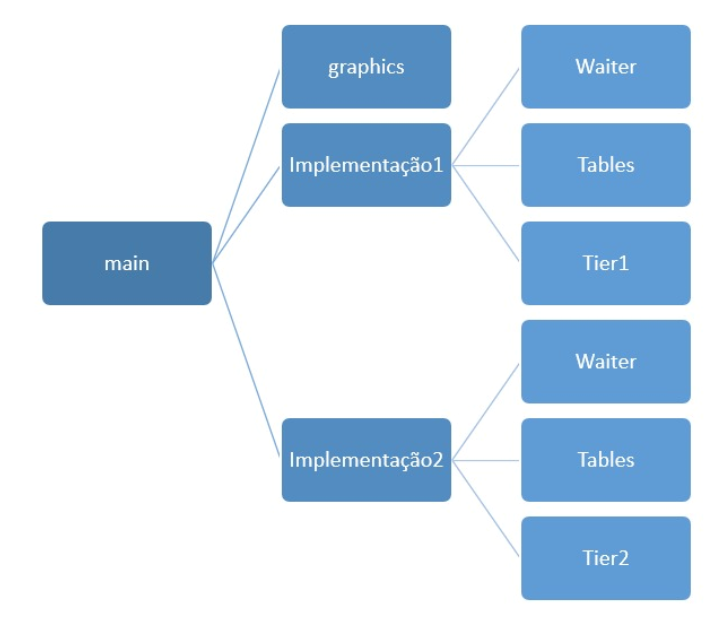
1.3.3 Parâmetros (aplicável para classes) 7

1.3.4 Métodos (aplicável para classes) 8

# Documentação técnica

## Arquitetura do programa

O programa que foi desenvolvido tem como propósito a simulação gráfica de robô que trabalha num restaurante, tendo como funções limpar o chão da sala, apanhar lixo e também servir às mesas. Para o desenvolvimento deste projeto foi utilizado a biblioteca “Graphics.py”, desenvolvida por John Zelle, e utilizamos os seguintes módulos de programação: main, Tier 1, Tier 2, waiter, tables. O módulo main é o ficheiro principal, neste está desenvolvido o menu, tendo sido definida a parte estética do menu que inclui as cores utilizadas, imagens, coordenadas e botões respectivos a cada implementação para que o menu tenha maior atratividade, mas mantendo o simples de entender. Neste módulo estão interligados os módulos Tier 1 e Tier 2, para que quando haja um click nos botões seja aberto uma nova janela executando a respetiva implementação. O módulo waiter possui a função de criar o waiter e uma outra função para criar as docks de carregamento do waiter, este módulo é necessário para ambas as implementações. O módulo Tables, define as funções que criam as mesas, sendo que uma função cria mesas circulares e outra cria mesas quadradas. No módulo Tier 1, é executada o movimento do robô (waiter) a volta da área da sala “limpando” o chão da área da sala desviando-se das mesas que encontra pelo seu caminho. Depois de iniciar o movimento na dock station e finalizar a limpeza da sala por completo este dirige-se a dockstation mais próxima. Na implementação 2, para além do robô limpar o chão este serve a mesa, porém ele tem bateria limitada e a seguir a servir duas mesas e dirigir-se ao balcão para buscar o prato, a sua bateria chega ao limite por isso este dirige à dockstation mais próxima para recarregar para além disto este tem a função de apanhar lixo que seja espalhado pela sala e retornando de seguida a dockstation mais próxima. No diagrama a seguir, é mostrada a interligação de todos os módulos (Classes, Arquivos, Bibliotecas):



## Projeto detalhado

1.2.1 **Módulo Main**

**Funções:**

**Main():** Esta função é responsável por executar o menu do programa. Nesta é gerada uma janela gráfica, que define a cor do fundo, apresenta também o nome do programa, uma pequena imagem e gera os botões para cada implementação. Na mesma há uma função que recebe as coordenadas e confirma se as mesmas encontram-se dentro dos botões que são gerados, se assim for efetua a função de cada um, senão simplesmente não acontecerá nada.

**Botão:** Esta classe recebe as coordenadas ao qual os botões serão gerados para a quantidade de implementações desejadas (3), nomeando cada um, definindo a cor dos mesmos.

**Exit\_botão:** Esta classe representa o botão de sair da janela representado por uma imagem “Exit”. Esta recebe as coordenadas, a largura e a altura do botão desejado. A classe cria um retângulo vermelho com o pdf “exit” lá projetado.

1.2.2 **Módulo Tier 1 (Implementação 1)**

**Funções:**

**Implementação 1():** Esta função é responsável por executar a primeira implementação do programa. Esta gera uma janela gráfica na qual é definida a cor de fundo e cria os elementos gráficos necessários, como as dockstations, o robô e as mesas. Imediatamente de seguida inicia um loop onde o robô move-se ao longo da janela para limpar a mesma evitando diferentes obstáculos (mesas), usando variadas funções definidas que caracterizam as colisões com constantes cálculos das distâncias entre o robô e as mesas para evitar erros. Assim sendo, este contorna as mesas e acaba por limpar a sala chegando ao final do seu objetivo e para na dockstation mais próxima.

1.2.3 **Módulo Tier 2 (Implementação 2)**

**Funções:**

**Implementação 2():** Esta função é responsável por executar a segunda implementação do programa. Esta gera uma janela gráfica na qual é definida a cor do fundo e cria os elementos gráficos necessários, como as dockstation, o robô e as mesas. Com um click na mesa o robô dirige-se à mesma recolhendo assim o pedido e posteriormente dirige-se à cozinha. Assim sendo retorna à mesa e de seguida à dockstation.

1.2.4 **Módulo Waiter**

**Funções:**

**Waiter:** Esta classe representa o robô na janela gráfica. Ela recebe as coordenadas e o raio do robô. A classe cria um círculo vermelho que representa o corpo do robô.

**Pratos:** Esta classe representa os pratos na janela gráfica. Ela recebe as coordenadas das mesas e ajusta gerando pratos nas mesma por uma questão de estética.

**Docks:** Esta classe representa as estações de acoplamento na janela. Ela recebe as coordenadas das mesma e o tamanho gerando assim como desejado.

1.2.5 **Módulo Tables**

**Funções:**

**TableR:** Esta classe representa as mesas retangulares que serão um dos obstáculos do robô na janela gráfica. Esta recebe as coordenadas do ponto superior esquerdo, não só mas também, a distância que pretende do outro ponto, gerando dessa forma a mesa desejada.

**TableC:** Esta classe representa as mesas circulares que são outros dos obstáculos do robô na janela gráfica. Assim sendo esta recebe as coordenadas do centro da mesa e o raio da mesma, gerando a mesa desejada. Esta inclui também a função do cálculo da distância entre o robô e as mesas que será usada nas colisões.

## Módulo Nome do Módulo

* + 1. **Entradas (aplicável para funções)**

**Arquivo: Waiter.py**

Classe Waiter : x, y, radius

Classe Pratos: x, y, raio

Classe Dock: x, y, tamanho

**Arquivo: Tables.py**

Classe TableR: x, y, color, width, height

Classe TableC: x, y, color, raiomesac

**Arquivo: main.py**

Classe Botão: x, y, color, label, width, height, radius

Classe Exit\_Botao: x, y, width, height

* + 1. **Saídas (aplicável para funções)**

**Arquivo: main.py**

Classe Botão: x, y, color, label, width, height, radius

Classe Exit\_Botao: x, y, width, height

* + 1. **Parâmetros (aplicável para classes)**

**Arquivo: Waiter.py**

Classe Waiter : x, y, radius

Classe Pratos: x, y, raio

Classe Dock: x, y, tamanho

**Arquivo: Tables.py**

Classe TableR: x, y, color, width, height

Classe TableC: x, y, color, raiomesac

**Arquivo: main.py**

Classe Botão: x, y, color, label, width, height, radius

Classe Exit\_Botao: x, y, width, height

* + 1. **Métodos (aplicável para classes)**

**Arquivo: Waiter.py**

Classe Waiter:: getCenter(), move(dx,dy), getP1(), getP2() draw(win)

**Arquivo: Tables.py**

Classe TableR: getCenter(), getP1(), draw(win)

Classe TableC: getCenter(), getP1(), draw(win)

**Arquivo: main.py**

Classe Botão: draw(win)

Classe Exit\_Botao: draw(win)