Nome: Alexandre Rodrigues Número: 75545

Nome: Miguel Cabrita Número: 75555

Nome: Afonso Rio Número: 76943

Grupo: PL3-G02 Parte 3

Pacote geometry:

Classe Point – Classe que guarda as coordenadas de um ponto no plano x0y.

Classe Segment – Classe que guarda dois pontos, representando um segmento de reta.

Classe Triangle – Classe que guarda três pontos, representando um triângulo.

Classe Rectangle – Classe que guarda quatro pontos, representando um retângulo.

Classe Polygon – Classe que guarda uma lista de pontos, representando um polígono.

Classe Circumference – Classe que guarda um ponto e um raio, representando o centro e o raio de uma circunferência.

Classe Obstacle – Classe abstrata que abstrai todos os obstáculos em apenas uma classe.

Classe Path – Classe que guarda uma lista de pontos, representando um caminho.

Classe Map – Classe que representa o mapa de obstáculos no playground.

Pacote facility:

Classe Robot – Classe que representa um robot, guardando toda a informação sobre um robot (Posição atual, nivel de bateria, se está disponível, a carregar ou em serviço, etc…)

Classe PathFinder – Classe que representa os cálculos que os robôs vão ter que realizar para realizar os trabalhos sem colidir com um obstáculo.

Classe RobotManager – Classe que representa o administrador dos robôs, guardando os pedidos para mais tarde dar a um certo robô, todos os robôs presentes na simulação e um mapa do “playground”.

Classe Job – Classe que representa o trabalho feito pelo robot, um ponto inicial e um ponto final.

Pacote simulation:

Classe MobileRoboticSimulation – Classe que representa a simulação de um robot. A simulação tem dois estados: um pré e um pós começo.

Classe MobileRoboticSimulationUI – Classe que representa a interface de usuário para a simulação de um robot.

Classe SimulationScanner – Classe que representa o scanner da simulação, que executa em paralelo à simulação, à espera de um input.

Interface IMobileRoboticSimulation – Interface que fornece uma blueprint para a simulação de um robot.

Pacote util:

Classe ArrayDequeAsQueue - Classe “opcional” com o propósito de filtrar a classe ArrayDeque, tendo acesso apenas aos métodos necessários para a implementação de uma queue. Também “muda” o nome dos métodos para que sejam mais intuitivos e descritivos de usar. Na realidade nada é implementado na classe, apenas chama métodos já implementados da classe ArrayDeque.

Classe UnweightedHashGraph – Classe que implementa um grafo sem peso, usando HashMaps com vértices e as suas respetivas conexões.

Interface IQueue – Interface que fornece uma blueprint para qualquer coleção que se comporte como uma queue.

Interface IUnweightedHashGraph – Interface que fornece uma blueprint para qualquer implementação de um grafo sem peso.

Pacote GUI:

Classe PreSimulationFrame - Classe que inclui um frame inicial em que o utilizador coloca a sua configuração desejada (tipo de obstáculos e pontos de carga) de uma simulação. Quando o utilizador termina a sua configuração, ao fechar o frame vai ser aberto um frame da simulação.

Classe FacilityPanel – Classe que contém um painel onde são desenhados os robots e os obstáculos.

Classe ExceptionFrame – Classe que quando acontece algum erro com a simulação, aparece um frame, mostrando uma mensagem de exceção e um botão, que quando pressionado fecha o frame.

Classe MobileRoboticSimulationGUI – Classe principal que é chamada pelo cliente para começar uma simulação.

Classe SimulationFrame – Classe onde a simulação ocorre. Quando o utilizador pressiona o botão para parar, o programa é fechado.

Classe Part2Client – Classe que contém o código da cliente da segunda parte.

Classe Part3Client – Classe que contém o código da cliente da terceira parte.

Decidimos utilizar os seguintes padrões de projeto:

Iterator – É um padrão que se baseia em conseguir iterar sobre uma coleção, e ter acesso aos seus elementos, sem ter visão do interior do objeto. Neste projeto usamos este padrão em classes como as classes auxiliares (util), Map e Path.

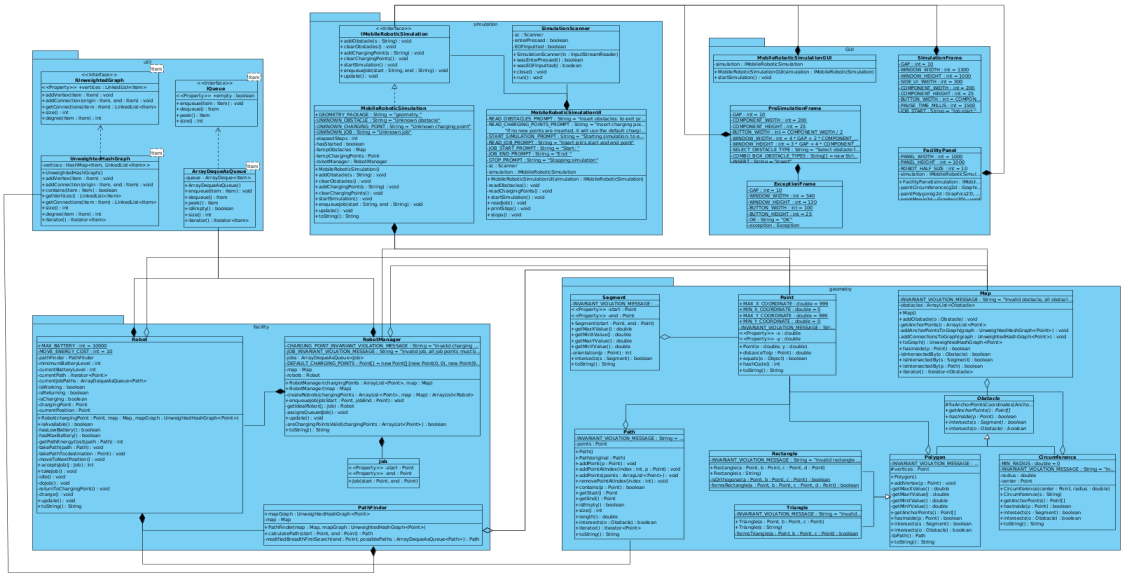
Facade – É um padrão que se baseia na abstração de um sistema complexo usando uma ou várias classes facades, neste projeto a classe MobileRoboticSimulation é uma classe Facade.

Composite – É um padrão que se baseia na divisão de um objeto complexo em objetos mais simples, este padrão é visível na classe Obstacle e todas as suas subclasses.

UML:

Como não é possível colocar a imagem do UML aqui, o mesmo está na pasta UML enviada com o resto dos ficheiros.

No entanto está aqui uma foto do mesmo de qualquer das formas:



Manual de utilização (utilizado na Parte 2):

Quando se inicia o programa, o utilizador é apresentado com a opção de adicionar obstáculos à simulação. Os obstáculos são colocados numa linha, cada um, com o seguinte formato:

<Tipo de obstáculo> <Propriedades>

Por exemplo:

Rectangle 0 0 1 0 1 1 0 1 (Retângulo com vértices (0,0);(1, 0);(1,1);(0,1))

Triangle 2 2 6 7 2 10 (Triângulo com vértices (2,2);(6, 7);(2,10))

Circumference 7 5 2 (Circunferência com centro (7,5) e raio 2)

Em caso de acontecer algum com erro com a inserção de um obstáculo, este não é introduzido e imprime uma mensagem com a descrição e causa do erro.

Uma vez que o utilizador se encontra satisfeito com os obstáculos, insere uma linha vazia para ir para a próxima fase.

Nesta fase, o utilizador é apresentado com a opção de adicionar os pontos de carga dos robots. Os pontos de carga são colocados numa linha, com o seguinte formato:

<coordenadaX> <coordenadaY>

O utilizador pode optar por escolher os pontos de carga default, por não inserir nenhum ponto de carga, ou seja, por inserir uma linha vazia de início.

Os pontos de carga default são: (0,0);(999,999);(999,0);(0,999)

Caso aconteça algum erro, é imprimida uma mensagem.

Uma vez que o utilizador se encontra satisfeito com os pontos de carga, insere uma linha vazia para começar a simulação. Caso aconteça algum erro, é imprimida uma mensagem com a causa do erro e este processo recomeça.

Se a simulação for inicializada com sucesso, o utilizador tem três opções:

1) Não inserir nada, e a cada segundo é impresso um passo da simulação.

2) Inserir o EOF, terminando a simulação e o programa.

3) Pressionar o enter, entrando no modo de submissão de trabalho. Neste caso, o utilizador é apresentado com o modo de submissão de trabalho, parando temporariamente os passos que os robots tomam, para inserir o ponto inicial e final do trabalho a ser executado. Em caso de erro, é impresso uma mensagem com a causa do erro e é apresentado novamente com o modo de submissão do trabalho.

Manual de utilização (utilizado na Parte 3):

Muito semelhante à parte 2, o user incialmente é apresentado com uma janela para inserir os obstáculos e os pontos de carga / robôs de uma simulação. O formato é exatamente o mesmo com a exceção de não precisar de escrever o tipo do obstaculo, já que o mesmo pode ser escolhido numa combo box. O user pressiona “Start” e começa a simulação.

Durante a simulação, o user poderá submeter um trabalho usando o mesmo formato na parte 2 num pequeno UI no lado esquerdo, pressionando “Submit job” o user poderá ver o caminho que os robos percorrem, ou não.

Pressionando o butao “Stop”, a simulção para e o programa chega ao fim

Melhoramentos feitos entre a parte 2 e a parte 3, correspondentes à parte 2:

Foi resolvido um bug que causava um extremo uso de memoria e tempo, potencialmente explodindo o programa, a calcular um caminho, este problema mostrava-se mais notável quando o caminho dado era impossivel de calcular.

A classe MobileRoboticSimulationUI foi alterada, de forma a conter tudo o que seja de respeito da mesma.