Agente Limpeza em Labirinto

Descrição | Objetivo | Estrutura do Projeto | Requisitos | Como usar | Configuração | Licença |

Info

Professor: Dr. Aldo Henrique Dias Mendes

Participantes: Lucas silva, Luan medrado

Disciplina: Inteligencia artificial

Semestre: 2024/2

Descrição:

Este projeto implementa um sistema de agente autônomo que navega e limpa um labirinto gerado aleatoriamente. O agente percorre o labirinto identificando e removendo sujeiras representadas por células específicas do ambiente.

Objetivos

O agente localizar a sujeita A(agua) e P(po) e limpa dando prioridade para Agua

Estrutura do Projeto

O projeto está organizado da seguinte forma:

```
- Main.java: Classe principal que executa o programa, inicializa o labirinto e controla o fluxo de execução.
- Labirinto.java: Classe que representa o ambiente do labirinto, com a geração aleatória de células contendo sujeira e obstáculos
- AgenteLabirinto.java: Classe que implementa o agente que navega pelo labirinto, tomando decisões sobre o movimento e a limpeza.
- PosicaoXY.java: Classe que define a posição do agente e outros objetos dentro do labirinto.
```

▶PAES (Percepção, Ação, Estado, Solução)

- Percepção: O agente percebe o ambiente ao seu redor identificando a célula em que está localizado e verificando os tipos de sujeira ("A" para água e "P" para poeira) ao seu redor.
- Ação: As ações do agente consistem em mover-se para uma célula vizinha e limpar o tipo de sujeira encontrada. As direções possíveis são cima, baixo, esquerda e direita, dependendo da validade da célula.
- Estado: O estado do agente é representado pela sua posição atual no labirinto e o número de movimentos realizados para limpar o ambiente. O estado do labirinto é alterado quando o agente limpa uma célula.
- Solução: A solução final é atingida quando todas as células com sujeira ("A" ou "P") são limpas, momento em que o agente para de se mover. O sucesso do agente é determinado pela capacidade de encontrar e limpar todas as células sujas.

👷‍♂️🔧Como usar

1. Clone o repositório:
    - Bash `git clone https://github.com/Lucca7r/`
2. Instale as dependências:
    - none

⚙Exemplo de Execução

- Ao iniciar, o labirinto será exibido com as células representando água ("A") e poeira ("P"). O agente se moverá e limpará as células, mostrando visualmente seu progresso até completar a tarefa de limpeza.

©Licença:

PARTES DE CADA ALUNO:

LUAN RESPONSAVEL PELO LABIRINTO

LUCAS RESPONSAVEL PELO AGENTE BUSCA SUJEIRA

CODIGO FONTE:

-MAIN

```
--MAIN


import agente.AgenteLabirinto;
import ambiente.Labirinto;
import geral.PosicaoXY;

public class Main{

    public static void main(String[] args) throws
InterruptedException   {

        Labirinto labirinto = new Labirinto(4);
        labirinto.exibirLabirinto();

        AgenteLabirinto agente = new AgenteLabirinto(labirinto);
        agente.setPosicao(new PosicaoXY(2,2));
        labirinto.setAgente(agente);

        while(agente.isAindaLimpando()) {
            agente.zerarPilha();
            agente.movimentar();
            labirinto.exibirLabirinto();
            Thread.sleep(1500);
        }
        return;
    }

}
```

--LABIRINTO

```
package ambiente;

import agente.AgenteLabirinto;
import geral.PosicaoXY;
import java.util.Random;

public class Labirinto {

    private int tamanhoLabirinto;
    private String[][] labirinto;
    private AgenteLabirinto agente;
    private Random random;



    public Labirinto(int tamanhoLabirinto) {
        this.tamanhoLabirinto = tamanhoLabirinto;
        this.random = new Random();
        this.construirNovoLabirinto();
    }


    private void construirNovoLabirinto() {
        labirinto = new
String[this.tamanhoLabirinto][this.tamanhoLabirinto];
        for (int i = 0; i < this.tamanhoLabirinto; i++) {
            for (int j = 0; j < this.tamanhoLabirinto; j++) {
                if (random.nextBoolean()) {
                    this.labirinto[i][j] = "P";
                } else {
                    this.labirinto[i][j] = "A";
                }
            }
        }
    }

    public void exibirLabirinto() {
        atualizarPosicaoAgente();
        for (int i = 0; i < tamanhoLabirinto; i++) {
            for (int j = 0; j < tamanhoLabirinto; j++) {
                if (labirinto[i][j].equals("*A*")) {
                    System.out.print("|" + labirinto[i][j] + "|");
```

```
                } else {
                    System.out.print("| " + labirinto[i][j] + " |");
                }
            }
            System.out.println("");
        }
        System.out.println("");
    }

    private void atualizarPosicaoAgente() {
        if (this.agente != null) {
            PosicaoXY posAgente = this.agente.getPosicao();
            if
(!labirinto[posAgente.getPosX()][posAgente.getPosY()].equals("L")) {
                labirinto[posAgente.getPosX()][posAgente.getPosY()] =
"*A*";
            }
        }
    }

    public int getTamanhoLabirinto() {
        return this.tamanhoLabirinto;
    }

    public String retornarValorPosicaoLabirinto(PosicaoXY posicao) {
        return this.labirinto[posicao.getPosX()][posicao.getPosY()];
    }

    public void setAgente(AgenteLabirinto agente) {
        this.agente = agente;
    }

    public void limpar() {
        if (this.agente != null) {
            PosicaoXY posicao = this.agente.getPosicao();
            labirinto[posicao.getPosX()][posicao.getPosY()] = "L";
        } else {
            throw new IllegalStateException("Agente não configurado no
labirinto.");
        }
    }
}
```

--AGENTELABIRINTO

```java
package agente;

import ambiente.Labirinto;
import geral.PosicaoXY;

public class AgenteLabirinto {

    private Labirinto labirinto;
    private PosicaoXY posXY;
    private MovimentosAgenteLabirinto movimento;
    private int pilhaMovimentos;

    public AgenteLabirinto(Labirinto labirinto) {
        this.labirinto = labirinto;
        this.posXY = new PosicaoXY(0, 0);
        this.movimento = MovimentosAgenteLabirinto.CIMA;
        this.pilhaMovimentos = 0;
    }

    public void setPosicao(PosicaoXY posicao) {
        this.posXY = posicao;
    }

    public void movimentar() {
        boolean existeAgua = existeAguaNoLabirinto();

        PosicaoXY proximaAgua = buscarSujeiraProxima("A");

        if (existeAgua && proximaAgua != null) {
            this.posXY = proximaAgua;
            this.labirinto.limpar();
            this.pilhaMovimentos++;

        }

        if (!existeAgua) {
            PosicaoXY proximoPo = buscarSujeiraProxima("P");
            if (proximoPo != null) {
                this.posXY = proximoPo;
                this.labirinto.limpar();
                this.pilhaMovimentos++;
            }
        }
    }
```

```
    public PosicaoXY buscarSujeiraProxima(String tipoSujeira) {
        for (int i = 0; i < labirinto.getTamanhoLabirinto(); i++) {
            for (int j = 0; j < labirinto.getTamanhoLabirinto(); j++) {
                if (labirinto.retornarValorPosicaoLabirinto(new
PosicaoXY(i, j)).equals(tipoSujeira)) {
                    return new PosicaoXY(i, j);
                }
            }
        }


        return null;
    }

    public boolean existeAguaNoLabirinto() {
        for (int i = 0; i < labirinto.getTamanhoLabirinto(); i++) {
            for (int j = 0; j < labirinto.getTamanhoLabirinto(); j++) {
                if (labirinto.retornarValorPosicaoLabirinto(new
PosicaoXY(i, j)).equals("A")) {
                    return true;
                }
            }
        }
        return false;
    }

    public String verificarTipoSujeira(PosicaoXY posicao) {
        return this.labirinto.retornarValorPosicaoLabirinto(posicao);
    }

    public boolean isPosicaoValida(int x, int y) {
        String valor = this.labirinto.retornarValorPosicaoLabirinto(new
PosicaoXY(x, y));
        return !valor.equals("P");
    }

    public PosicaoXY retornarMovimento() {
        int retornoPosX = this.posXY.getPosX();
        int retornoPosY = this.posXY.getPosY();

        switch(movimento) {
            case CIMA:
                if (retornoPosX > 0 && isPosicaoValida(retornoPosX - 1,
retornoPosY)) {
                    retornoPosX -= 1;
                } else {
                    movimento = MovimentosAgenteLabirinto.BAIXO;
                }
```

```
                break;
            case BAIXO:
                if (retornoPosX < this.labirinto.getTamanhoLabirinto() -
1 && isPosicaoValida(retornoPosX + 1, retornoPosY)) {
                    retornoPosX += 1;
                } else {
                    movimento = MovimentosAgenteLabirinto.ESQUERDA;
                }
                break;
            case ESQUERDA:
                if (retornoPosY > 0 && isPosicaoValida(retornoPosX,
retornoPosY - 1)) {
                    retornoPosY -= 1;
                } else {
                    movimento = MovimentosAgenteLabirinto.DIREITA;
                }
                break;
            case DIREITA:
                if (retornoPosY < this.labirinto.getTamanhoLabirinto() -
1 && isPosicaoValida(retornoPosX, retornoPosY + 1)) {
                    retornoPosY += 1;
                } else {
                    movimento = MovimentosAgenteLabirinto.CIMA;
                }
                break;
        }
        return new PosicaoXY(retornoPosX, retornoPosY);
    }

    public PosicaoXY getPosicao() {
        return this.posXY;
    }

    public boolean isAindaLimpando() {
        return pilhaMovimentos < 4;
    }

    public void zerarPilha() {
        this.pilhaMovimentos = 0;
    }
}
```