

```

package Consola;

import java.util.ArrayList;

public class AdvancedHemissonConsole {

    private ArrayList<HemissonRobot> robots;
    private HemissonRobot robot;
    private int createdRobots;

    private static final int esquerda = 5;
    private static final int direita = 3;

    public AdvancedHemissonConsole(String[] args) {
        this.robots = new ArrayList<HemissonRobot>();
        this.robot = null;
        this.createdRobots = 0;
    }

    public void runConsole() throws Exception {
        Scanner sc = new Scanner(System.in);
        String command;
        while(true) {
            System.out.print("Cmd: ");
            command = sc.next();
            if(command.equalsIgnoreCase("create")) {
                createRobot(sc);
            } else if(command.equalsIgnoreCase("setActive")) {
                changeActiveRobot(sc);
            } else if(command.equalsIgnoreCase("delete")) {
                deleteRobot(sc);
            } else if(command.equalsIgnoreCase("help")) {
                printHelp();
            } else if(command.equalsIgnoreCase("infmove")) {
                makeMove(sc, false);
            } else if(command.equalsIgnoreCase("move")) {
                makeMove(sc, true);
            } else if(command.equalsIgnoreCase("read")) {
                makeRead();
            } else if(command.equalsIgnoreCase("readl")) {
                makeReadLight();
            } else if(command.equalsIgnoreCase("rt")) {
                command = sc.next();
                Direccao d = null;
                if(command.equalsIgnoreCase("direita"))
                    d = new Direccao(Direccao.direita);
                else if(command.equalsIgnoreCase("esquerda"))
                    d = new Direccao(Direccao.esquerda);
                else
                    d = new Direccao(Direccao.tras);

                d.executarMovimento(robot);
            } else if(command.equalsIgnoreCase("stop")) {
                robot.setSpeed(0,0);
            } else if(command.equalsIgnoreCase("execute")) {
                command = sc.next();
                TarefaRobot th = null;
                if(command.equalsIgnoreCase("passear"))
                    th = new PassearFugir(robot, null);
                else if(command.equalsIgnoreCase("linha")) {
                    AvaliadorAmbiente a = new AvaliadorAmbiente(robot);
                    a.init();
                    a.calcularLimiar();
                    th = new SegueLinha(robot, a);
                    System.out.println("Carregue enter para iniciar");
                    sc.next();
                }
            }
        }
    }
}

```

```

    } else if(command.equals("linhaFast")) {
        AvaliadorAmbiente a = new AvaliadorAmbiente(robot);
        a.init();
        a.calcularLimiar();
        System.out.println("Carregue enter para iniciar");
        sc.next();
        th = new SegueLinhaFast(robot,a);
    } else if(command.equals("passearFast"))
        th = new PassearFugirAgressivo(robot,null);
    else if(command.equals("passearTurnos"))
        th = new PassearFugirAgressivoTurnos(robot,null);
    else if(command.equals("passearDetect"))
        th = new PassearFugirDetect(robot,null);
    else if(command.equalsIgnoreCase("passearDetectBeacon"))
        th = new PassearFugirDetectContinuo(robot,null);
    else if(command.equals("detect"))
        th = new DetectarRobot(robot,null);
    if(th != null) {
        th.start();
        System.out.println("Escreva stop para terminar a tarefa");
        command = sc.next();
        if(command.equalsIgnoreCase("stop")) {
            th.interrupt();
            robot.setSpeed(0,0);
            System.out.println("Tarefa terminada");
        } else {

        }

    } else
        System.out.println("Função desconhecida");
    } else if(command.equalsIgnoreCase("quit")) {
        break;
    } else {
        System.out.println("Comando Desconhecido");
    }
    robot.beeper(1);
}

private void deleteRobot(Scanner sc) {
    // TODO Auto-generated method stub

}

private void changeActiveRobot(Scanner sc) {
    // TODO Auto-generated method stub

}

private void createRobot(Scanner sc) {
    String nome, comm;

    System.out.println("");
    System.out.println("Criar um novo robot");
    System.out.println("Indique o nome do robot: ");
    nome = sc.next();
    System.out.println("Indique a porta COM que deverá ser usada (número apenas:");
    ");
    comm = sc.next();
    try {
        int p = Integer.parseInt(comm);
        if (p <= 0)
            throw new Exception("invalid number");
        comm = "COM" + p;
    } catch (Exception e) {

```

```

        System.out.println("COM inválida");
        return;
    }
    robot = new HemissonRobot(comm, nome);
    robots.add(robot);
    this.createdRobots++;
}

private void printHelp() {
    System.out.println("Consola para controle de robos Hemisson");
    System.out.println("Comandos disponiveis:");
    System.out.println("Help: mostra esta mensagem");
    System.out.println("infmove Esq Dir: Efectua o movimento com os valores Esq e Dir nos motores esquerdo e direito, e desactiva a consola");
    System.out.println("move Esq Dir: Efectua o movimento com os valores Esq e Dir nos motores durante 1 segundo");
    System.out.println("read: Mostra os valores obtidos pelos sensores de distancia (Sonar)");
    System.out.println("readl: Mostra os valores obtidos pelos sensores de luz (Light)");
    System.out.println("rt Direccao: Permite efectuar um movimento de rotaçao");
    System.out.println("\t Direccao pode tomar valores: esquerda, direita, tras");
    System.out.println("execute Tarefa: Lança uma thread que executa uma certa operação");
    System.out.println("\t Tarefa pode tomar os seguintes valores:");
    System.out.println("\t passear - Tarefa para passear num mundo desviando-se de obstáculos");
    System.out.println("\t linha - Tarefa para seguir uma linha preta num chão com paragens (branco) envolve calibração de sensores");
    System.out.println("\t linhaFast - Tarefa para seguir uma linha preta num chão sem paragens (branco) envolve calibração de sensores");
    System.out.println("\t detect - Tarefa para passear detectar outro robot (objecto em movimento) num espaço");
    System.out.println("\t passearFast - Tarefa para passear num mundo desviando-se de obstáculos (algoritmo mais agressivo)");
    System.out.println("\t passearTurnos - Tarefa para passear num mundo desviando-se de obstáculos (algoritmo mais agressivo com pausas)");
    System.out.println("\t passearDetect - Tarefa para passear num mundo desviando-se de obstáculos e parar quando detecta um robot");
    System.out.println("\t passearDetectBeacon - Tarefa anterior que só para se estiver perante um beacon (robot em rotaçao)");
    System.out.println("Stop: coloca valores zero em ambos os motores do robot");
}

private void makeReadLight() {
    int[] values = robot.readLight();
    for(int i = 0; i<values.length; i++) {
        System.out.print(values[i] + ";");
    }
    System.out.println("Chao: " + values[esquerda] + " : " + values[direita] );
}

private void makeRead() {
    int[] values = robot.readSonar();
    for(int i = 0; i<values.length; i++) {
        System.out.print(values[i] + ";");
    }
    System.out.println("Chao: " + values[esquerda] + " : " + values[direita] );
}

private void makeMove(Scanner sc, boolean stop) throws InterruptedException {
    int esquerda = sc.nextInt();
    int direita = sc.nextInt();
    robot.setSpeed(esquerda, direita);
    if(stop) {
        Thread.sleep(1000);
    }
}

```

```

        robot.setSpeed(0,0);
    } else {
        System.exit(0);
    }
}

/**
 * @param args
 * @throws InterruptedException
 */
public static void main(String[] args) throws InterruptedException {
    AdvancedHemissonConsole c = new AdvancedHemissonConsole(args);
    try{
        c.runConsole();
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}

```