Prácticas de Laboratorio. SINT

Amueblando el cerebro: árboles de decisión





ENTREGA

Fecha de entrega:

19 de Abril de 2022 23:59:59

Método de entrega:

Tarea en Moodle

Entregable (.zip):

Código fuente + poster explicativo (diagrama de clases, reglas, resultados, comparativas, etc)



OBJETIVOS GENERALES

En la práctica 2 aprendimos a:

HEMOS AMUEBLDO EL CEREBRO

En esta práctica tendremos como objetivo inicial:

CAMBIAR EL RAZONAMIENTO: de REGLAS a ARBOLES DE DECISIÓN

Como objetivo principal:

aplicar un razonamiento basado en árboles de decisión.



OBJETIVOS ESPECÍFICOS I

Objetivo 1:

Representación del entorno: YA LO TENEMOS

Objetivo 2:

creación de listas de objetos, asociados por su tipología: YA LO TENEMOS



OBJETIVOS ESPECÍFICOS II

Objetivo 3:

Creación de un motor con un árbol de decisión a partir de la clase abstracta que

permita cambiar el razonamiento

Abstracta class Razonador +piensa(): ACCION

class Árbol Decisión

Implementa piensa(): ACCION

class Árbol Acción

Implementa piensa(): ACCION



OBJETIVOS ESPECÍFICOS II

Objetivo 3:

ARBOL DE DECISION +RAIZ: NodoArbol **NODO ARBOL** +piensa(Mundo m): ACTION Devolver acción NODO ARBOL: PREGUNTA 2 +decide(Mundo m) NODO ARBOL: P4 NODO ARBOL: PREGUNTA 1 +decide(Mundo m) +decide(Mundo m) NODO ARBOL: PREGUNTA 3 +decide(Mundo m) NODO ARBOL: P3 +decide(Mundo m)



OBJETIVOS ESPECÍFICOS II





OBJETIVOS ESPECÍFICOS I

Nodo Arbol Abstracto: Define método abstracto decide (Mundo m)

Nodo Acción extiende Nodo Arbol Abstracto

Define método abstracto doAction(Mundo m) e Implementa decide(Mundo m)

Nodo Decision extiende Nodo Arbol Abstracto

Define método abstracto getBranch (Mundo m) e Implementa decide(Mundo m)

• Devuelve lo que diga su "hijo"



OBJETIVOS ESPECÍFICOS I

<abstract>>
Razonador
<<abstract>> +piensa(Mundo m):ACTIONS

<<abstract>>

ArbolDeDecision

Contiene la raíz del árbol Implementa piensa(Mundo m)

Devuelve la acción a realizar a través del método decide del nodo raíz

<<abstract>>

Decision

+getBranch(Mundo m): NodoArbol

Define getBranch() e Implementa decide

Extenderán los nodos que implementaran la decisión y devolverán el nodo elegido según se cumpla o no la pregunta

<<abstract>>

NodoArbol

+decide(Mundo m): NodoArbol

Método abstracto de decisión

<<abstract>>

Accion

+doAction (Mundo m): ACTIONS

Define doAction() e Implementa decide

Extenderán los nodos de acción que impelementes doAction() para devolver la acción a realizar



OBJETIVOS ESPECÍFICOS I

```
Agent02_01.java

    Agent03_01.java ×

☑ Agent02_01.java × ☑ Agent03_01.java
                                            1 package si2022.antoniopalancodedu.p03;
    package si2022.antoniopalancodedu.p0
                                             3 import core.game.StateObservation; □
  3⊕ import core.game.StateObservation;
 11
                                               public class Agent03 01 extends AbstractPlayer {
    public class Agent02 01 extends Abst
                                            16
 13
                                                    // Atributos persistentes del agente
         // Atributos persistentes del ag
 14
                                                   Razonador cerebro;
        Motor cerebro:
 15
                                                   MiMundo89 mundo;
                                            19
         MiMundo89 mundo;
 16
                                            20
 17
                                            219
                                                    public Agent03 01(StateObservation stateObs, ElapsedCpuTimer elapsedTimer)
 18⊖
         public Agent02 01(StateObservati
                                            22
 19
                                                        mundo = new MiMundo89(stateObs);
             mundo = new MiMundo89(state0
 20
                                            24
                                                        cerebro = new MotorArbol 89();
             cerebro = new Motor89();
 21
                                            25
 22
                                            26
 23
                                           △27⊝
                                                    public Types.ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer) {
         public Types.ACTIONS act(StateOb 28
△24⊝
                                                        mundo.actualizar(stateObs);
                                                                                                // PERCIBIR
 25
             mundo.actualizar(stateObs);
                                                       //mundo.pintar();
             //mundo.pintar();
 26
                                            30
                                                       ACTIONS a = cerebro.piensa(mundo);
                                                                                                // PENSAR
             Regla a = cerebro.disparo(mu 31
 27
                                                        return a; // ACTUAR
             return a.getAccion().doActio 32
 28
 29
                                            33 }
                                            34
 30 }
```



EJERCICIOS

EJERCICIO 1: (YA LO TENEMOS DE LA PRÁCTICA ANTERIOR)

Seleccionar el juego 89

Crear una clase Mundo que contenga la representación del tablero VACIO y un conjunto de listas con los OBJETOS del mapa.

Crear un método que lo pinte para verificar que la interpretación es correcta. Pintar las diferentes categorías...

Una vez comprobada, se puede obviar el método.

```
Carcel (&)?: 1
Nubes (*)?: 47
Alguno cayendo (@)?: 1
Buenos (o)?: 18
Hay disparos (-)?: 1
Hay metas (G)?: 2
Hay Malos (M)?: 3
Recursos (R)?: 0
Avatar ha comido (Y)?: 1
```



EJERCICIOS

Ejercicio 2: continuacion

Crear el razonamiento (cerebro) basado en árboles de decisión que extiende del abstracto.

Diseñar un Árbol de decisión que resuelva el juego 89 en todos sus niveles.

Es necesario crear los nodos de decisión y acción necesarios para resolver el juego.

