Prácticas de Laboratorio. SINT

Amueblando el cerebro: máquinas de estado finitas

Universidad de Huelva

ENTREGA

Fecha de entrega:

11 de Mayo de 2022 23:59:59

Método de entrega:

Tarea en Moodle

Entregable (.zip):

Código fuente + ¿?

OBJETIVOS GENERALES

En la práctica 2 aprendimos a:

HEMOS AMUEBLDO EL CEREBRO

En esta práctica tendremos como objetivo inicial:

CAMBIAR EL RAZONAMIENTO: Máquinas de estados finitos

Como objetivo principal:

aplicar un *razonamiento basado en máquinas de estados finitos*.

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

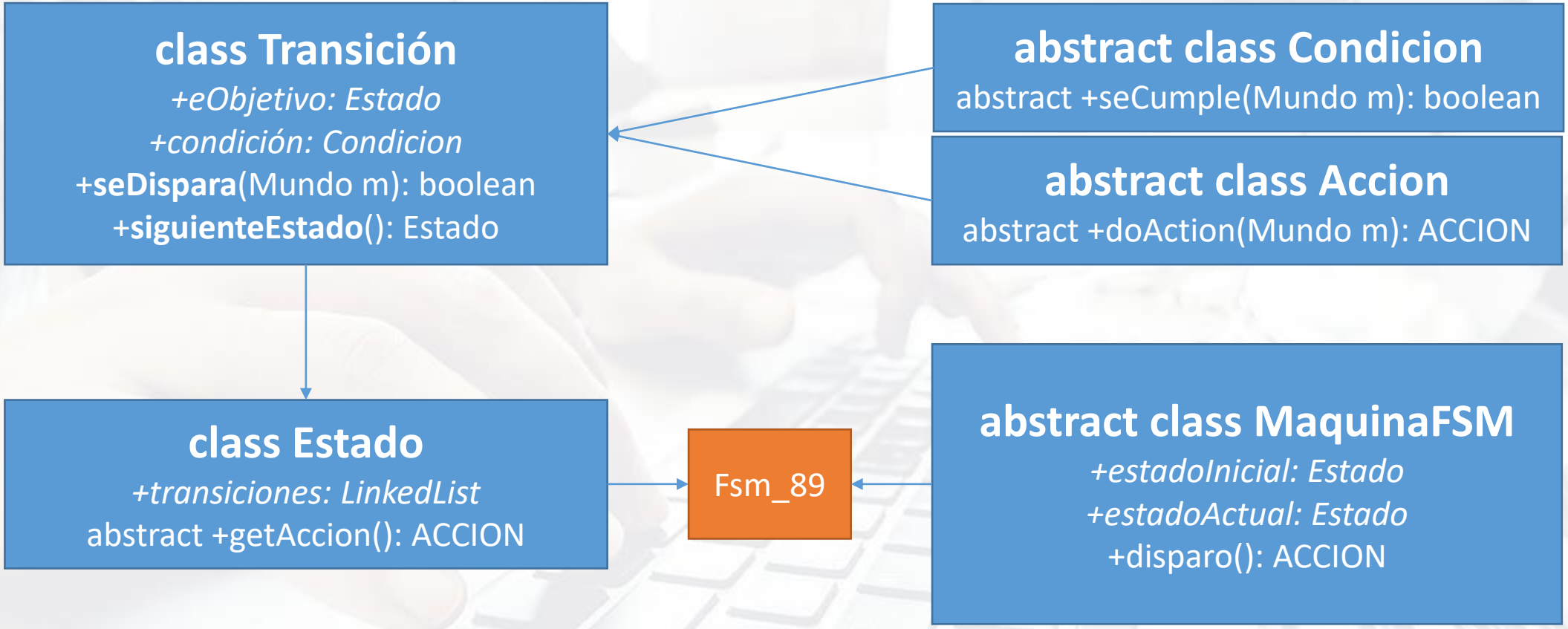
Objetivo 3:

Creación de un motor con DIAGRAMA DE ESTADOS FINITOS

- Estados: serán nuestras acciones (IrALaCarcel, VaciarSaco…)
- Condiciones: la percepción del entorno (estoyEnPeligro, TripaLlena,…)
- Transiciones: Pensar.

OBJETIVOS ESPECÍFICOS II

Objetivo 3:



MAQUINA DE ESTADOS FINITA

CLASE MSF: ¿Qué es lo que hace esta clase abstracta?

- Tendrá el estado inicial y el estado actual
- Implementa el método disparo, que busca entre las transiciones del estado actual si alguna de ellas se cumple, en caso afirmativo, se dispara dicha transición y se obtiene el siguiente estado, almacenándolo en el estado actual.
- Esto ocurrirá por cada tic del juego.

OBJETIVOS ESPECÍFICOS I

CLASE MSF_89: extiende de la clase MSF

Define los estados

Inicializa el estado actual al inicial.

Y añade las diferentes transiciones necesarias para la resolución del juego.

OBJETIVOS ESPECÍFICOS I

CLASE MSF_89: extiende de la clase MSF

Define los estados

Inicializa el estado actual al inici

Y añade las diferentes transicion

```
public MaquinaFSM_89() {

    Estado inicial      = new Inicial();
    Estado fin          = new Inicial();
    Estado escape       = new SalirAgujero();
    Estado endescarga   = new EnDescarga();
    Estado descenso     = new Descenso();

    this.EActual = inicial;

    inicial.addTransicion(new Transicion(new ultimafila(),escape));
    inicial.addTransicion(new Transicion(new True(),descenso));

    escape.addTransicion(new Transicion(new tripallena(),endescarga));

    //añadir trancisiones
}
```

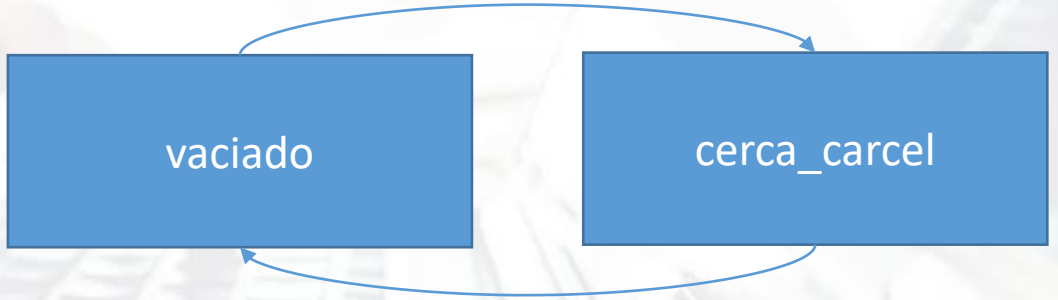
ANOTACIONES

NOTAS

Siempre debe haber un estado activo

Cada estado puede tener más de una transición

Cuidado con los saltos continuos



Hacer el diagrama de estados en papel antes de empezar

NUESTRO AGENTE

Agent02_01.java × Agent03_01.java

```java
package si2022.antoniopalancodedu.p02.agente89;

import core.game.StateObservation;

public class Agent02_01 extends AbstractPlayer {

    // Atributos persistentes del agente
    Motor cerebro;
    MiMundo89 mundo;

    public Agent02_01(StateObservation stateObs, ElapsedCpuTimer elapsedTimer)
    {
        mundo = new MiMundo89(stateObs);
        cerebro = new Motor89();
    }

    public Types.ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer) {
        mundo.actualizar(stateObs);              // PERCIBIR
        //mundo.pintar();
        Regla a = cerebro.disparo(mundo);        // PENSAR
        return a.getAccion().doAction(mundo);    // ACTUAR
    }
}
```

NUESTRO AGENTE

Agent02_01.java × Agent03_01.java

```java
package si2022.antoniopalancoded

import core.game.StateObservatio

public class Agent02_01 extends

    // Atributos persistentes de
    Motor cerebro;
    MiMundo89 mundo;

    public Agent02_01(StateObser
    {
        mundo = new MiMundo89(st
        cerebro = new Motor89();
    }

    public Types.ACTIONS act(Sta
        mundo.actualizar(stateOb
        //mundo.pintar();
        Regla a = cerebro.dispar
        return a.getAccion().doA
    }
}
```

```java
package si2022.antoniopalancodedu.p04;

import core.game.StateObservation;

public class Agent04_01 extends AbstractPlayer {

    // Atributos persistentes del agente
    MaquinaFSM_89 cerebro;
    MiMundo89 mundo;

    public Agent04_01(StateObservation stateObs, ElapsedCpuTimer elapsedTimer)
    {
        mundo = new MiMundo89(stateObs);
        cerebro = new MaquinaFSM_89();
    }

    public Types.ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer) {
        mundo.actualizar(stateObs);              // PERCIBIR
        //mundo.pintar();
        Estado a = cerebro.disparo(mundo);       // PENSAR
        return a.getAction().doAction(mundo);    // ACTUAR
    }
}
```

EJERCICIOS

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

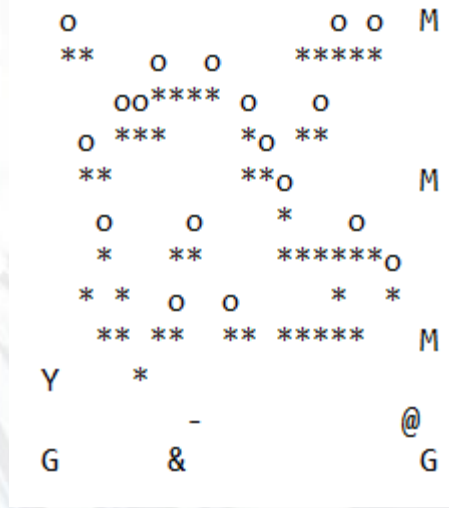
Seleccionar el juego 89

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:

Crear el razonamiento (cerebro) basado en MAQUINA FINITA DE ESTADOS de forma abstracta.

Diseñar una máquina de estados que extienda de la anterior y que contenga los estados, transiciones y acciones necesarias para resolver el juego 89 en sus 4 niveles