Trabajo Práctico Grupal N°3

Pacman

1er. Cuatrimestre de 2012

75.10 Técnicas de Diseño

Facultad de Ingeniería, Universidad de Buenos Aires

Fecha de entrega: 5 de Julio del 2012

Apellido y Nombre	Padrón	mail
Benitez, Cristian	78783	cbenez@gmail.com
Scoppa, Alfredo	89149	alfredo.scoppa@hotmail.com
Szperling, Leonel	88845	lszperling@gmail.com

Tabla de Contenidos

Implementación del Pacman – Segunda iteración

1 Implementación pedida

Se pide extender el modelo de dominio realizado en la iteración 1 para soportar los siguientes requerimientos

1.1 El laberinto y sus bolitas

El laberinto define la distribución de lugares validos de transito en donde en cada una de estas posiciones hay bolitas (y bolones) y por donde se desplazan los fantasmas y el pac-man. Tiene definido la posición de inició del pac-man y el lugar de donde salen los fantasmas (al comienzo y cuando pasan de estado muerto a estado cazador)

Las bolitas y los bolones son comidas únicamente por el pac-man. Cuando un bolón es comido todos los fantasmas que están en estado cazador son convertidos a presa (por un intervalo X) y todos los que son presa permanecen presa (por el mismo intervalo x)

Se pide modelar el laberinto del pac-man (con sus X bolones y sus portales)

El laberinto debe instanciarse a partir de un archivo, donde se indican las conexiones entre los eslabones del laberinto. Y que eslabones contienen bolitas o bolones. El formato del mismo se definirá a lo largo de la semana próxima.

No se pide modelar el manejo de partidas con niveles, ni tampoco de puntajes por bolita comida.

Dado que no hay interfaz gráfica, se pide serializar en archivos xml el estado de cada iteración del juego. Para ello ante cada iteración o tick del juego (entendiendo tick como el mensaje del juego a todos sus arcades para que se muevan) se solicita leer de un archivo el movimiento elegido para el pacman, realizar los movimientos correspondientes y luego serializar a un xml el estado del juego para ese tick. El formato solicitado para la serialización tanto para el movimiento elegido del pacman, como al del estado del juego, serán definidos durante la semana próxima también.

1.2. El pac-man

El mismo recibirá las órdenes del usuario para moverse en las cuatro direcciones. El pac-man siempre tiene una velocidad de 2 x Velocidad del fantasma en estado normal. (Los fantasmas tienen la velocidad minima (1) en estado normal, 2 en

estado molesto y 3 en estado furioso) La Velocidad quiere decir que el mas rápido se moverá ante cada tick del juego, y el mas lento cada 3 por ejemplo.

Cuando el pac-man come un fantasma cazador (o es cazado por un fantasma) muere.

2. Comportamiento autómata de los fantasmas.

Los fantasmas en estado cazador pueden desplazarse al menos en alguno de los siguientes 4 criterios de vivacidad:

- Zonzo: Se mueve aleatoriamente, solamente se lanza a comer al pac-man si este se encuentra a N1 (cuatro por ej) lugares de distancia, si este se escapa a mas de N1 lugares continúa moviéndose aleatoriamente.

- Perezoso: Se mueve aleatoriamente, solamente se lanza a comer al pac-man si este se encuentra a N2 (ocho por ej) lugares de distancia.

- Buscador: Se mueve aleatoriamente, si el pac-man se encuentra a N3 lugares de distancia lo persigue. Si el pac-man se escapa a más de N3 (diez por ej) lugares de distancia el fantasma buscador va hacia la última posición donde lo vio (no se mueve de manera aleatoria) y si en el camino lo ve se dirige hacia esa posición.

- Buscador temperamental: Es igual al buscador anterior, pero si el fantasma incrementa su nivel de ira incrementa en uno la visión (inicialmente tiene una visión de N4 lugares(12). Si se reinicia el nivel de ira entonces también se reinicia la visión.

El movimiento aleatorio mencionado anteriormente tiene que cumplir con los siguientes requerimientos:

- Una vez que entra a un canal del laberinto, sigue hasta llegar a un cruce o bifurcación, recién ahí vuelve a decidir hacia donde ir. Y no es opción regresar por donde vino, a menos que sea un canal sin salida.

- A la llegada de un cruce si no sabe a donde ir, dado su visión, entonces define aleatoriamente que camino tomar.

- Siempre se mueven, una vez entrado a un canal sigue hasta salir del mismo, al llegar a un cruce decide a donde ir. es decir nunca están quietos. El fantasma en estado presa debe alejarse del pac-man, es decir que se debe moverse según se describió pero alejándose en los casos que

antes se acercaba al pac-man.

1.4. Test unitarios

Se piden un set de test unitarios sobre el modelo de dominio descripto en 1. El set de test unitarios que se desarrolle debe representar un código valioso, debe motivar al equipo de trabajo a mantenerlo en el tiempo por la utilidad que brinda. Debe seguir las buenas prácticas de implementación de test unitarios.

1.5. Consola de prueba

● No habrá consola de prueba en este caso, ya que se podrá monitorear el comportamiento viendo los archivos que serializan el estado del juego.

2. Aclaraciones sobre la implementación

Queda fuera de la iteración los siguientes requerimientos:

- Interfaz grafica del pac-man.

- Sistema de partidas: Jugadores y turnos, vidas y puntajes, nuevos laberintos. (El juego arranca en estado inicializado según el archivo que lo describe, y termina al morir el pac-man o al comer todas las bolitas)

-Lógica de cómo un fantasma muerto, regresa al inicio (como se desplazaría de muerto a su lugar de inicio) Directamente desaparece y aparece cuando revive del punto de inicio.

- Otras estrategias de los fantasmas cazadores de cómo atacar al pac-man (si eligen el camino mas corto o más largo, si lo encierran en grupo, etc)

- En caso de tomar una decisión de implementación que este fuera del alcance descripto en 1, esta no debe contradecir ningún requerimiento pedido y se debe enunciar en el informe la hipótesis que la justifique, las mismas deben validarse con el ayudante.

3. Criterios de corrección

Se evaluará:

- El diseño
- El diseño de código
- Los test unitarios

- Cumplir con toda la funcionalidad descripta en 1

- El informe completo. Carátula, índice, enunciado, diagramas UML con descripciones orientadas a los contenidos de la materia, hipótesis tomadas, extractos de códigos de ejemplo y conclusiones.

4. Sobre la entrega

Se debe entregar el informe completo, impreso y vía e-mail (no hace falta imprimir el código fuente, pero si los test unitarios que realicen).

El código fuente se entrega vía e-mail, con las instrucciones de cómo realizar el deploy de la consola de prueba.

La fecha de entrega del tp es el Jueves 21 de Junio. La entrega vía mail debe hacerse antes de las 12 hs del mismo dia, enviando un mail (informe + fuentes + instrucción de deploy y configuración de la consola) con el asunto TP03GXX (donde XX corresponde al número de grupo dado en el TP1) a ayudantes@tecnicasdedisenio.com.ar

Los grupos que no lleguen a terminar alguna funcionalidad y por lo tanto entregar el tp completo en fecha podrán realizar la entrega al estado actual (y avanzado), indicando el estado actual del mismo y solicitar una nueva fecha de entrega con 2 semanas adicionales para terminar lo pendiente. Pero de todas formas debe realizarse la entrega de la manera indicada.

Se recomienda, ademas de en las clases, realizar consultas del diseño, validaciones de codigo o consultar cualquier dudas vía e-mail al ayudante.

Diagrama de clases

A continuación se presentan distintos diagramas de clases que constituyen los bloques conceptuales fundamentales del diseño propuesto.

El modelo presentado se fue construyendo en forma incremental y paralela a la elaboración de un diagrama de estado para Fantasma. Es por eso que nos resulto natural la elección del patrón **State.**

También nos resulto adecuado que sean los estados los responsables de saber cuanto tiempo deben permanecer y posiblemente cual debiera ser el próximo estado. Si hubiésemos optado por cargar con estas responsabilidades al Fantasma, en lugar de sus estados, estaríamos continuamente verificando en Fantasma el tipo de estado, para saber por ejemplo si puede ser comido o si es que se puede mover o no. Al delegar esta tarea en sus estados, son ellos quienes realizan este tipo de validaciones.

Para la lógica autómata de los fantasmas lo resolvimos aplicando el patrón **strategy,** se elaboro una estrategia para cada tipo de comportamiento. En resumen, se crea una cadena de delegación entre el fantasma, su estado y su estrategia.

Diagrama de clases de la jerarquía Comible

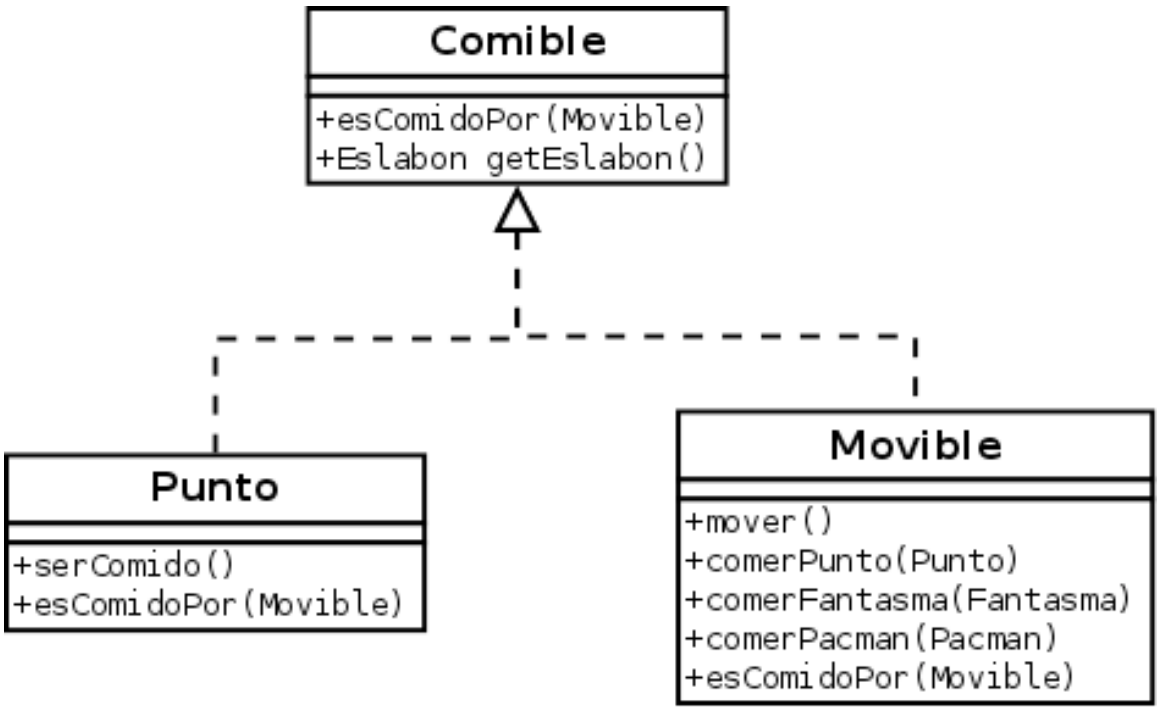


Diagrama de clases de las estrategias

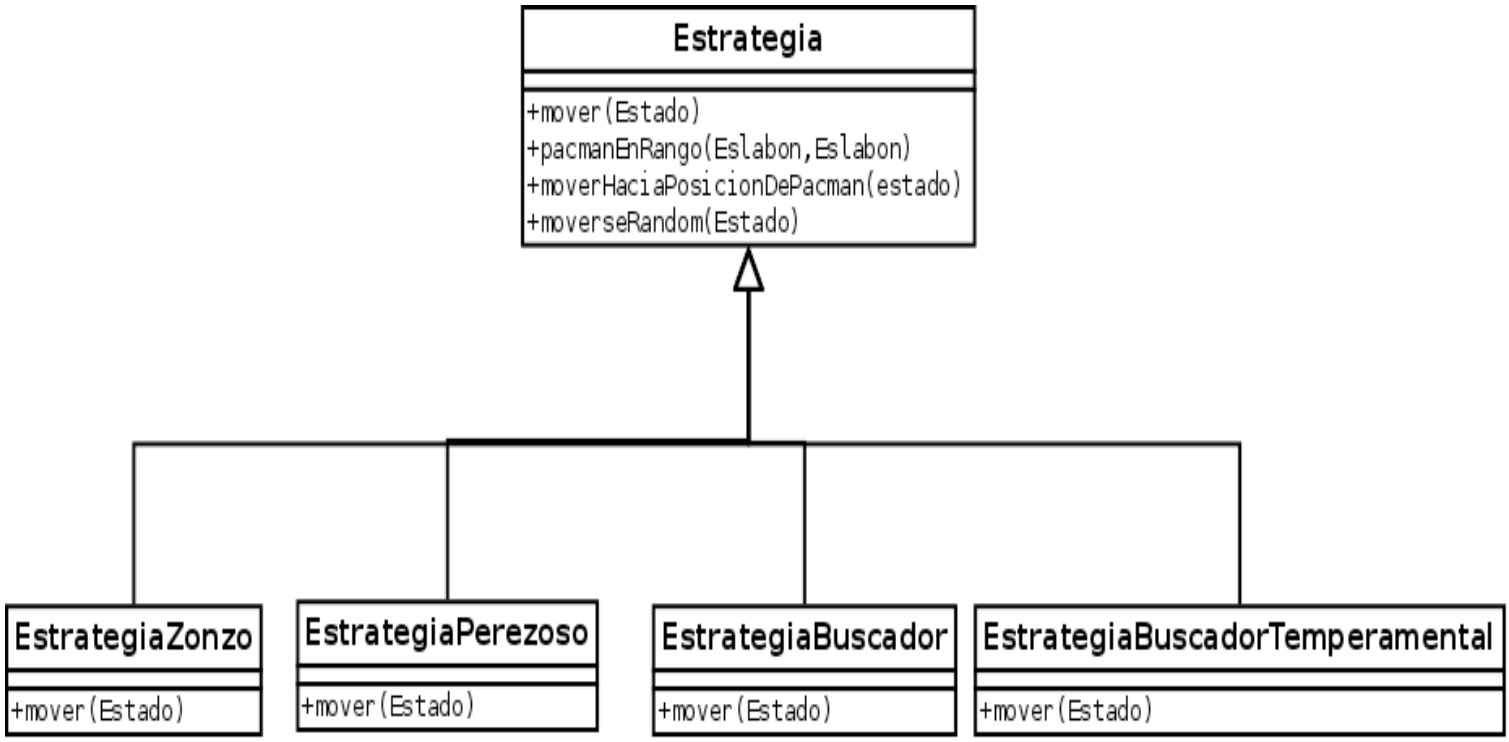


Diagrama de clases de laberinto

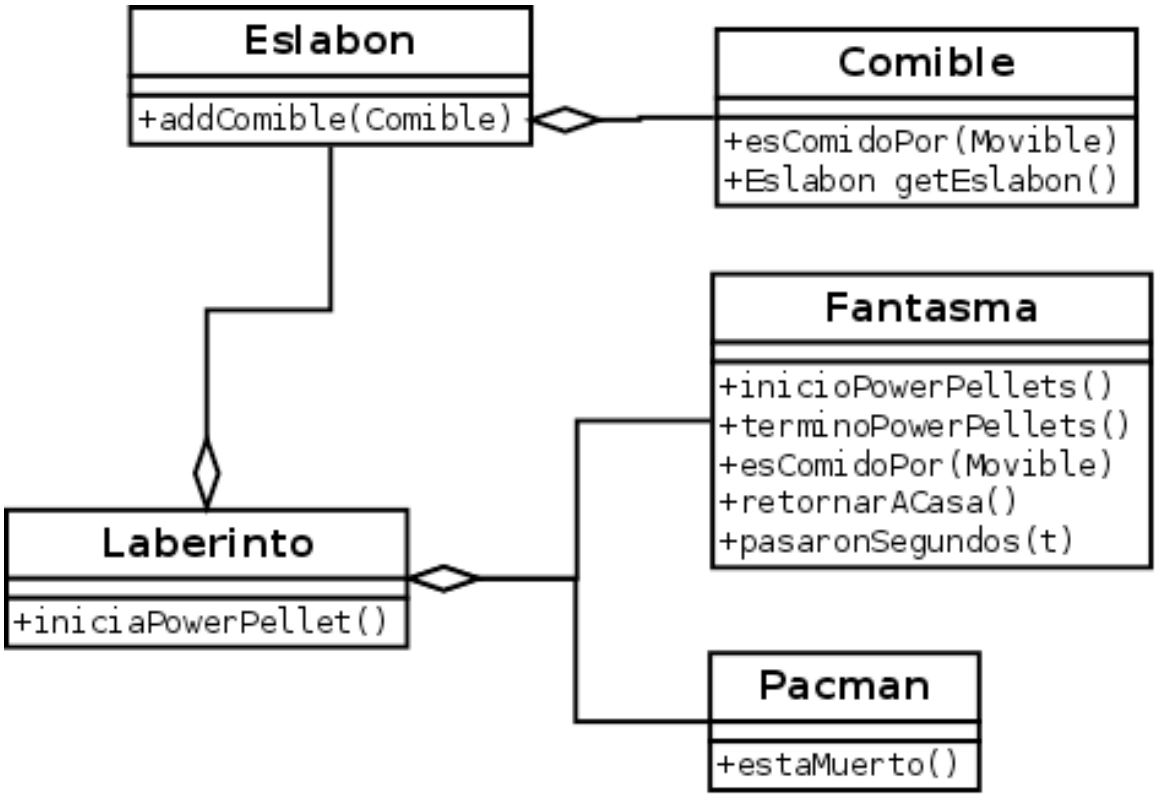


Diagrama de clases de Movible

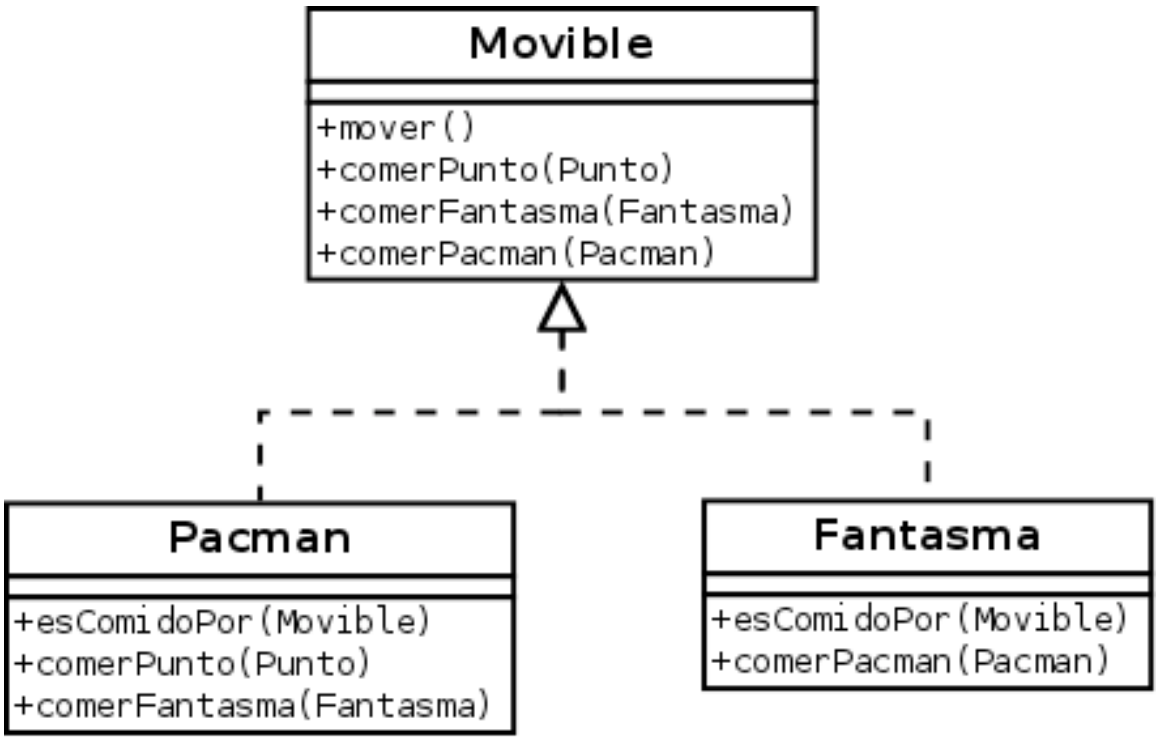
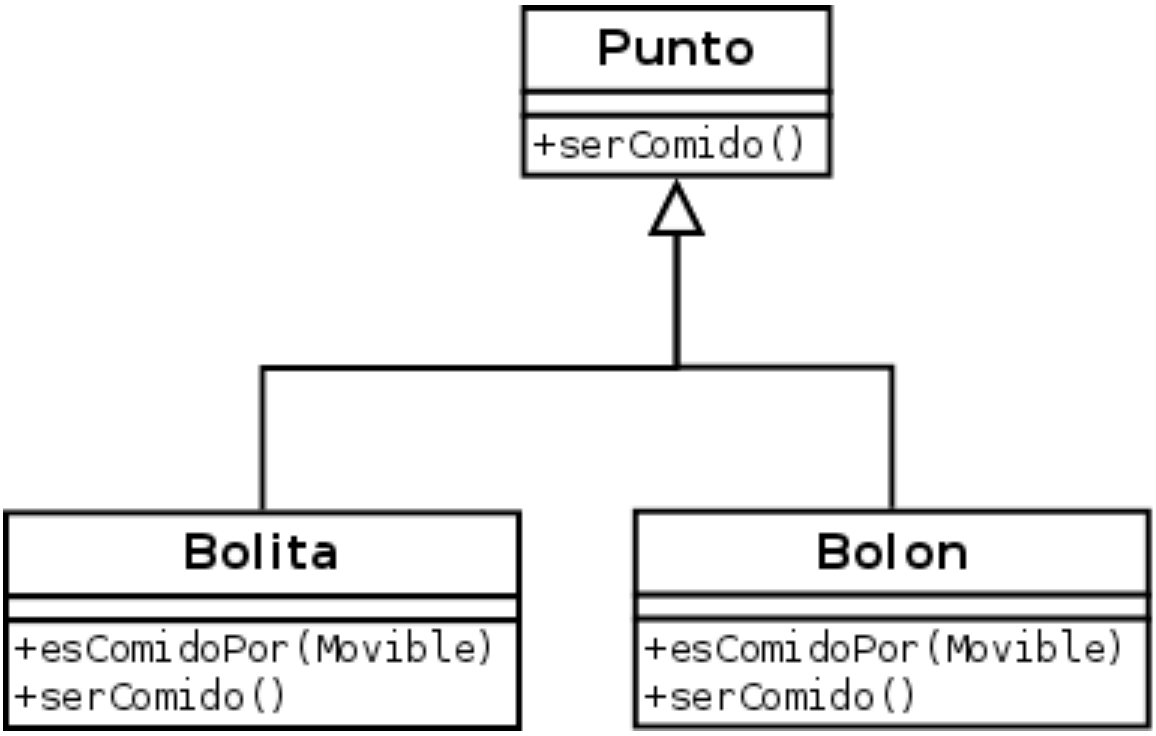


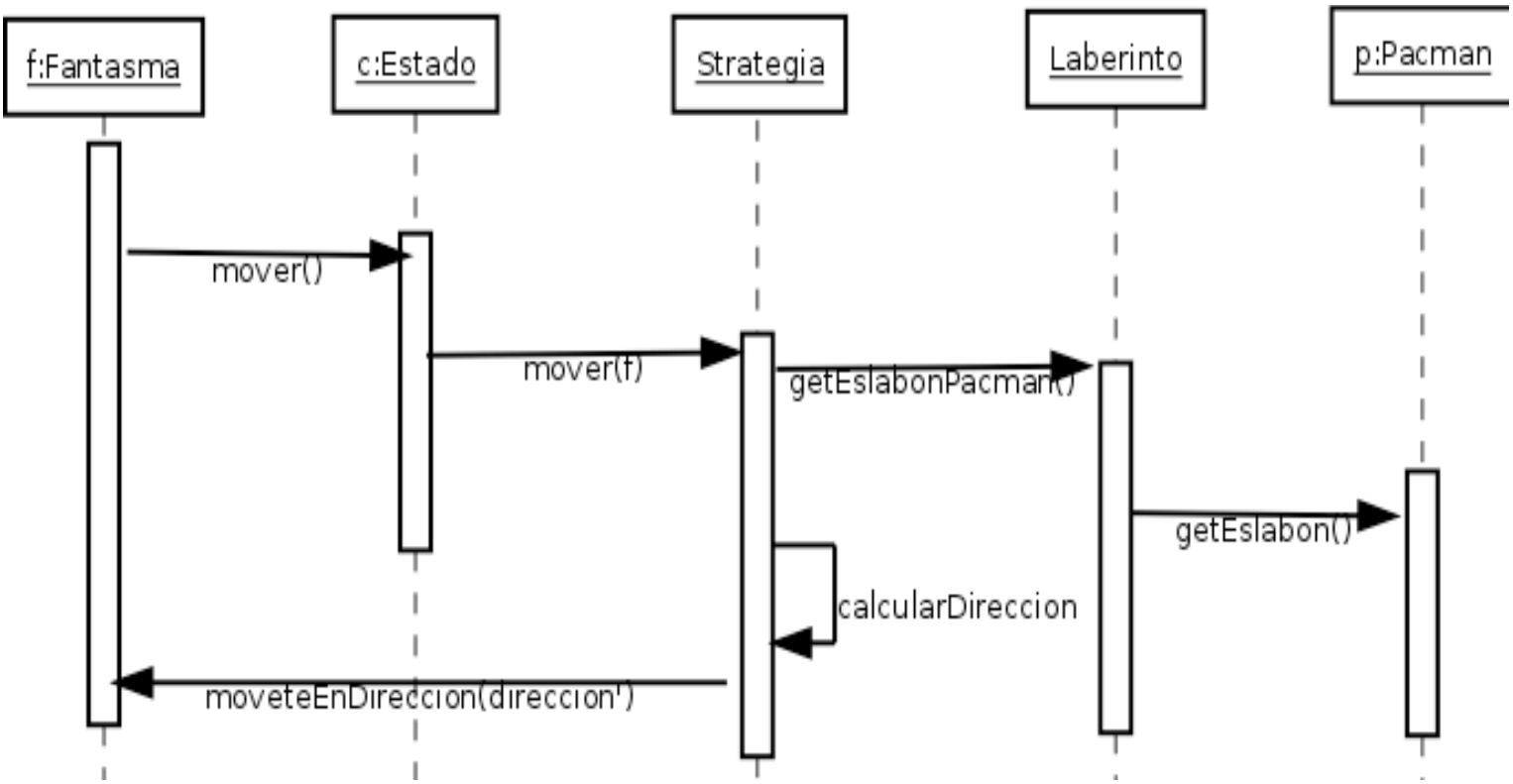
Diagrama de clases de punto



Diagramas de secuencia

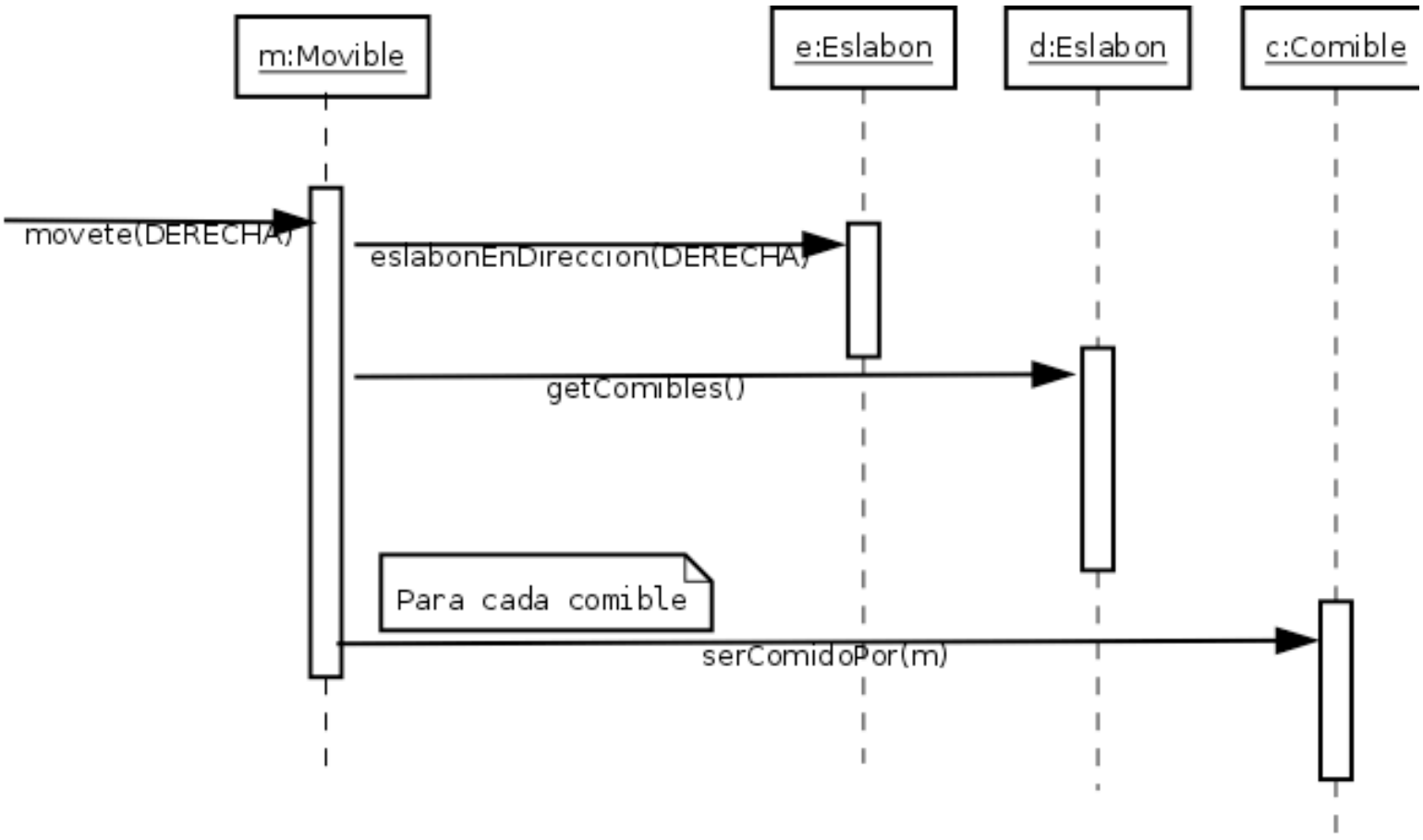
Mover fantasma.

Escenario: Se mueve un fantasma, con estado cazador.



Mover movible

Escenario: Mover cualquier movible.



Test unitarios

Las pruebas unitarias se encuentran adjuntadas como parte de la entrega, todas ellas obedecen el siguiente formato:

```
public class XXXTest extends TestCase{
        //Inicialización de logica comun a todos los test
        public void setUp(){}

        public void testXXXPositivo(){}

    //Uno o mas de este tipo si hay varios casos negativos
    public void testXXXNegativoPorXXX(){}
}
```

Fantasma

Se prueba la creación y la transición de estados para el fantasma, contemplando casos negativos y positivos.

Archivo: testFantasma.java

EstrategiaZonzo

Se prueba la creación y asignación de esta estrategia a un fantasma y luego que el fantasma reaccione adecuadamente a la presencia del pacman.

Archivo: testEstrategiaZonzo

Bolita

Se prueba la creación de una bolita y que sea comida por un pacman.

Archivo: testBolita.java

Laberinto

Se prueba la creación de un laberinto construyéndolo desde un archivo xml y mockeando sus eslabones.

Archivo: testLaberinto.java

Pacman

Se prueba la creación de un pacman.

Archivo: testPacman.java

Eslabon

Se prueba la creación de un eslabon y la asignación de sus eslabones adyacentes.

Archivo: testEslabon.java

Persistencia

Se prueba la persistencia de los diferentes actores del pacman.

Archivo: testPersistencia.java