Trabajo Práctico Grupal N°2

Pacman

1er. Cuatrimestre de 2012

75.10 Técnicas de Diseño

Facultad de Ingeniería, Universidad de Buenos Aires

Fecha de entrega: 7 de Junio 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

1 Implementación pedida

1.1 Modelo de dominio

En esta primera iteración se pide diseñar e implementar solamente un fantasma del juego Pacman.

El fantasma del juego está caracterizado por poder estar:

-  Muerto
-  Como fantasma cazador, con niveles de agresividad
-  Como fantasma Presa

El fantasma se puede mover, puede ser comido, puede incrementar su ira (incrementa su agresividad). Por naturaleza al comienzo es cazador y puede ser convertido en presa en cualquier momento.

Si un fantasma esta muerto no puede incrementar su ira, no se puede mover ni tampoco puede ser comido ni tampoco puede ser convertido en presa. Al morirse debe reiniciar el nivel de agresividad y en un tiempo x (ver aclaraciones) debe volver a "revivir" como cazador.

Si un fantasma es cazador no puede ser comido, puede incrementar su nivel de agresividad (hasta un máximo), puede moverse acorde a su nivel de agresividad (ver aclaraciones) y puede ser convertido en presa.

Si un fantasma es presa puede ser comido (y debe estar muerto luego de ser comido), no puede incrementar su nivel de agresividad y no puede ser convertido a presa. Al cabo de un tiempo x (ver aclaraciones) debe volver a un naturaleza cazador.

No se pide nada de implementación de interfaz gráfica ni ninguna otra funcionalidad del juego que la anteriormente descripta.

1.2. Test unitarios

Se piden un set de test unitarios sobre el modelo de dominio descripto en 1.1.

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.

1.3. Consola de prueba

El objetivo de la consola es proveer una interfaz de prueba que permita verificar que se cumplen con los requerimientos y restricciones pedidas en 1.1.

La consola debe permitir:

- Iniciar el fantasma (debería imprimir: "Fantasma Cazador")
- Comer el fantasma
- Convertir en presa el fantasma
- Mover el fantasma
- Mostrar fantasma (debería imprimir: "Fantasma X con agresividad Y" donde x puede ser "Cazador", "Muerto" o "Presa". Los valores de agresividad se dejan a criterio del grupo)
- Salir de la consola

La consola debe tener un archivo de configuración donde se carguen los siguientes valores:

- Tiempo en segundos en que un fantasma que está muerto, permanece muerto.
- Tiempo en segundos en que un fantasma presa, permanece como presa. Ante cada acción que se haga sobre la consola se debe imprimir un mensaje en pantalla (ver aclaraciones)

Ante cada acción que se haga sobre la consola se debe imprimir un mensaje en pantalla (ver aclaraciones)

2. Aclaraciones sobre la implementación

Si bien las siguientes etapas contemplaran como darle inteligencia de robot al fantasma, meterlo en el laberinto clásico del juego con sus bolitas (bolones y frutas) y agregar el Pacman, para esta entrega solo se pide implementar del

dominio del Pacman descripto en el punto 1.1, permitiendo con una consola de prueba obtener una salida en pantalla descriptiva , ejemplos:

- Si yo a un fantasma que es cazador y lo intento comer (opción de consola "Comer el fantasma"), debo obtener una salida en pantalla que diga algo parecido a: "No me podes comer, soy cazador". Si es presa "Soy presa y me comiste". Si está muerto "No me podes comer, estoy muerto".

- Si yo muevo un fantasma (opción de consola "Mover el fantasma") me tiene que decir: si es cazador "Soy cazador y me muevo con nivel de agresividad X", si está muerto "Estoy muerto, no me puedo mover", si es presa "Soy presa y me escapo".

- Es responsabilidad del fantasma permanecer muerto el tiempo configurado.

- Es responsabilidad del fantasma permanecer presa el tiempo configurado. Esto quiero decir que en esos métodos se implementaran salidas a pantalla. En la etapa posterior, cuando crezca el contexto del dominio, se decidirá si se lanza una excepción, si interaccionara con otros objetos, etc.

3. Criterios de corrección

Se evaluará:

- El diseño
- El diseño de código
- Los test unitarios
- Cumplir con toda la funcionalidad descripta en 1.1 y 1.3
- El informe completo. Carátula, índice, enunciado, diagramas UML con descripciones orientadas a los contenidos de la materia, 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).

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 7 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 TP02GXX (donde XX corresponde al número de grupo dado en el TP1) a ayudantes@tecnicasdedisenio.com.ar.

No hay re entrega. La idea es arrancar en esa fecha con una nueva iteración. Por esto se recomienda evacuar todas las dudas vía e-mail o en la clase anterior a la entrega y mantener contacto via mail.

Diagrama de clases

En el siguiente diagrama se priorizo la claridad y facilidad de lectura, por ejemplo en los casos donde los tipos de los parámetros son evidentes estos fueron omitidos, también se evito la redundancia, evitando repetir definiciones de métodos abstractos en subclases.

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.

Por otra parte, la Partida es responsable del manejo de tiempo y es quien notifica a los fantasmas del paso del mismo quienes a su vez notifican a su estados correspondientes.

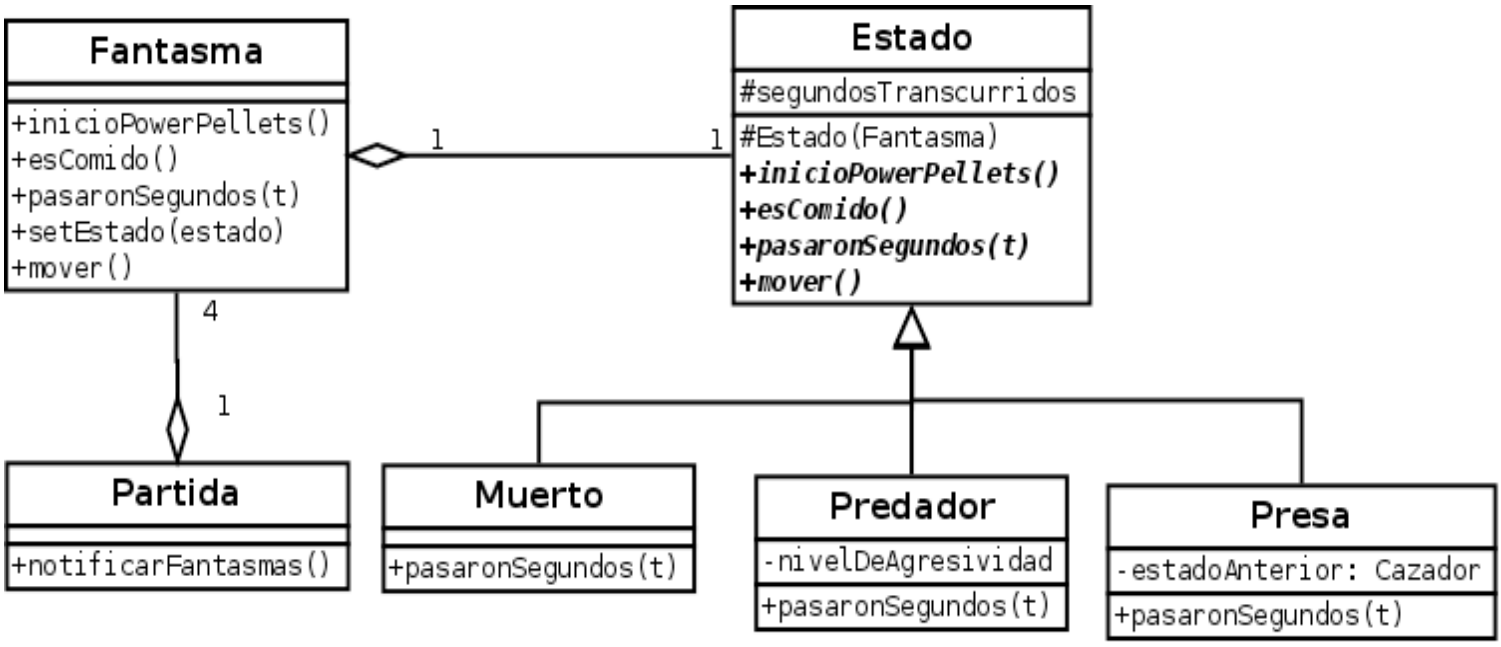
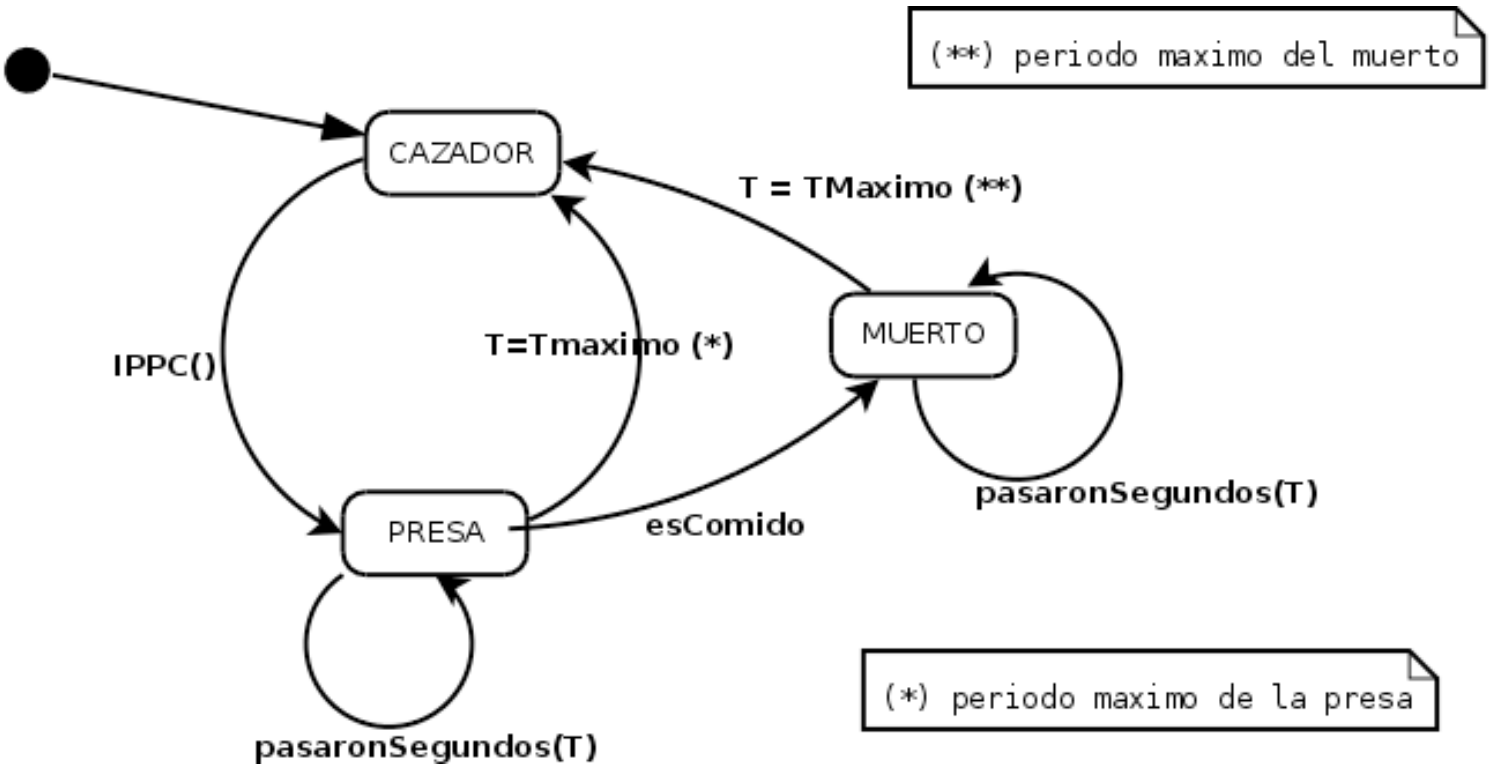


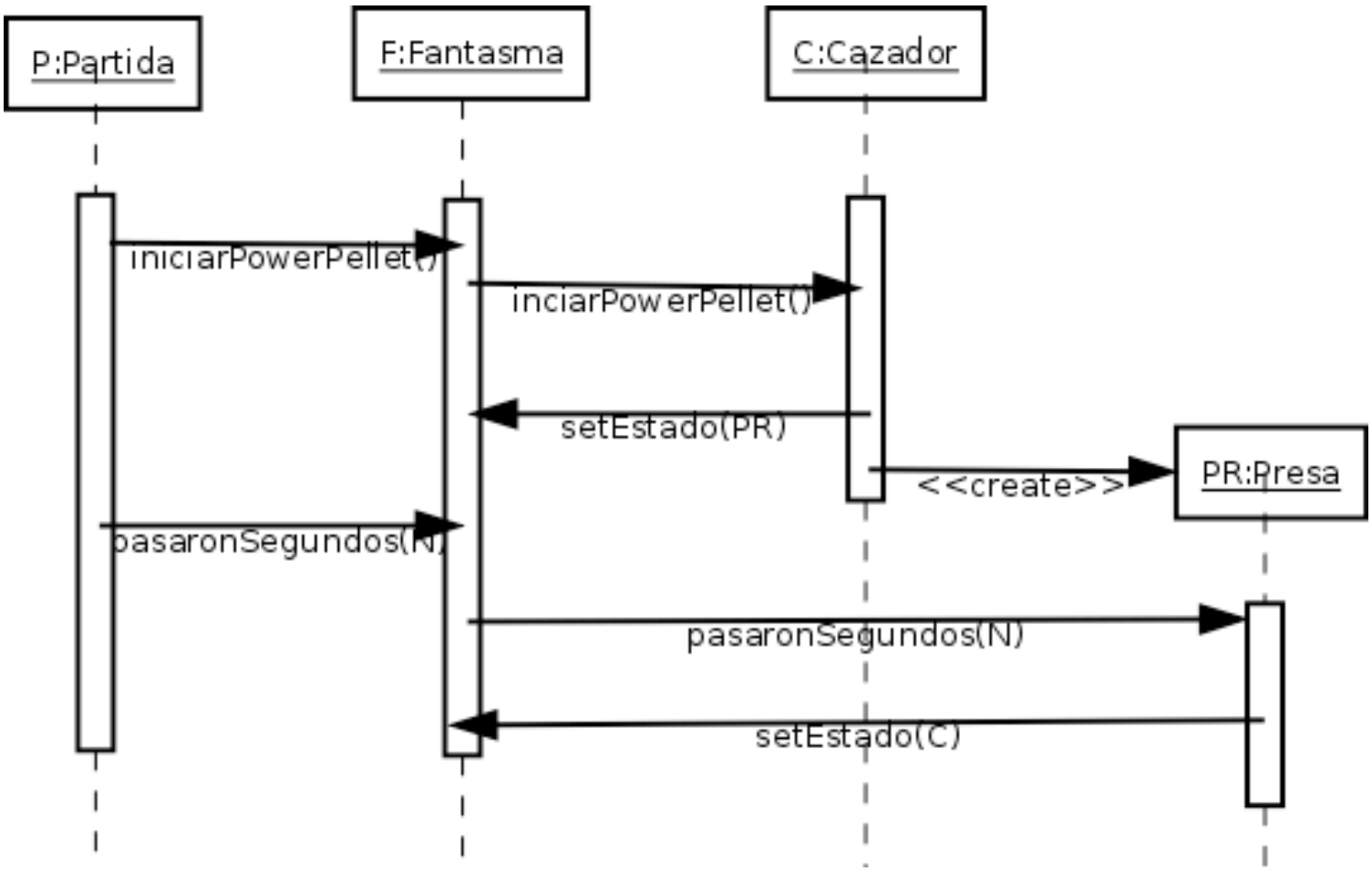
Diagrama de estado Fantasma



Diagramas de secuencia

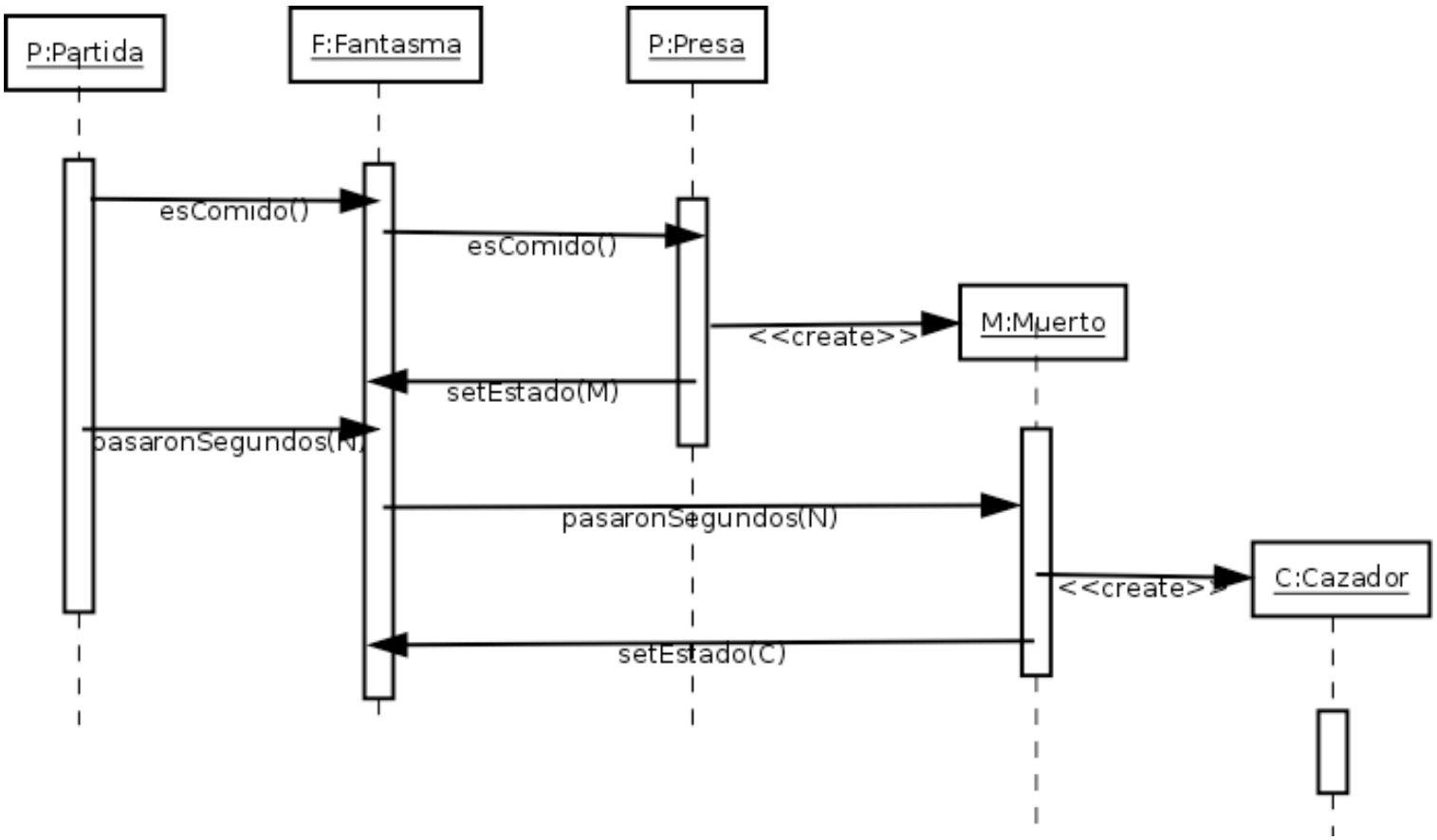
Convertir a presa

Escenario: Se crea un fantasma con estado Cazador.



Comer Presa

Escenario: El fantasma en estado presa es comido y se muere.



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

Aplicación de Consola

Para poder ejecutar la aplicación la misma debe compilarse usando algún IDE (como ser eclipse) o compilarse usando el archivo ant provisto (build.xml). El código se encuentra adjuntado a esta entrega dentro de la carpeta **codigo.**

Luego en el archivo pacman.properties se pueden configurar distintos parámetros como ser: el tiempo en que un pacman permanece muerto, el tiempo que permanece como presa y el máximo valor de agresividad.

Configuración de la consola

pacman.properties

```
#periodo de muerte
periodoMuerte=30

# Periodo de presa
periodoPresa=20

# Maximo nivel de agresividad
maximoNivelAgresividad=10
```

Compilar la aplicación con ant

```
$ tar -zxvf TP2_GRUPO_6_zip.zip
$ cd TP2_GRUPO_6_zip
$ cd codigo
$ ant
$ cp pacman.properties bin
$ cd bin
$ java consola/Consola
```

El uso de la aplicación es bastante claro y el menú es tal cual esta explicitado en el enunciado.

```
************************************************************
Trabajo practico grupal N° 2: Pacman
************************************************************
1.- Iniciar Fantasma
2.- Comer el fantasma
3.- Convertir en presa el fantasma
4.- Mover el fantasma
5.- Mostrar el fantasma
6.- Salir
Opción:
```