"Pac-Man"

Práctica de la Asignatura de Programación Orientada a Objetos

Escenario para el Curso 2013/2014 –Versión 1.1

Departamento de Lenguajes y Sistemas Informáticos

Escuela Técnica Superior de Ingeniería Informática - UNED

1.- Introducción

Los objetivos que se plantean en la realización de esta práctica son los siguientes:

- Familiarización con la Programación Orientada a Objetos (POO): definición de clases e instancias, uso de la herencia, definición/uso de métodos estáticos y abstractos,
- Realización del diseño orientado a objetos de un problema.
- Implementación de un programa sencillo donde se manejen conceptos relacionados con POO.

La práctica se va a implementar en Java 2 Estándar Edition (J2SE). El compilador de Java que se usará será BlueJ, tal y como se define en el programa de la asignatura. La asignatura además dispone de un curso virtual en aLF.

2.- Programación Orientada a Objetos en Java

El paradigma de programación orientada a objetos define un programa como una colección de entidades que se relacionan para resolver un problema. Estas entidades, que se conocen genéricamente como objetos, están tienen por un conjunto de propiedades y métodos, y están organizadas en torno a una jerarquía de clases.

En Java cada objeto puede tener variables y métodos privados y públicos. Se puede modificar dicha visibilidad de una clase usando los modificadores de acceso a miembros. Las dos maneras más habituales de especificar la accesibilidad son:

private – la variable o método está disponible solamente para esta clase,

public – la variable o método está disponible para todas las clases,

Una clase puede heredar los variables y métodos públicos de otra clase a través del mecanismo de herencia y la palabra clave extends. Por ejemplo:

```
//clase base que va a contener información sobre vehículos de nuestra empresa:

public vehiculo {

        private int noPuertas;

        private int noRuedas;

        private String modelo;

        public vehiculo(){}

        public void setNoPuertas(int np) {

                noPuertas = np;

        }

//etc.

}

//una clase para tratar a los coches en general...

 public coche extends vehiculo {
      private boolean airbags;

      public coche(){}

      public void setAirbags(Boolean a) {
              airbags = a;
      }
      //etc.
 }
 //y, por fin, una clase para tratar a los coches deportivos
 public final cocheDeportivo extends vehículo {

      private String capacidadMotor;

      private int maxVelocidad;

      public cocheDeportivo(){}

      public void setCapacidadMotor(String cm) {
              capacidadMotor = cm;
      }

      //etc.
      //se puede llamar a cualquier método en las superclases como
      //si estuvieran dentro
      //de esta misma clase, p.ej.:
      setNoPuertas(2);
 }
```

Notas: Las clases que extienden otras clases tienen el nombre de subclases y las clases que son extendidas por otras clases tienen el nombre de superclases.

Hay que tener cuidado a la hora de planificar las relaciones de herencia entre clases en Java porque una clase solamente puede heredar variables y métodos de otra (y sus superclases). Es decir, que no hay herencia múltiple en Java como hay en lenguajes como C++ (aunque se puede reproducir la técnica de herencia múltiple usando interfaces…). De todas formas, la manera más habitual para tratar está tema es simplemente usar una clase dentro de otra, por ejemplo, si hay una clase para el aparcamiento de una empresa que ya es una extensión de una clase base aparcamiento, dicha clase no puede heredar ninguna otra clase, por lo tanto, se incluirán las clases de coches, camiones, motos, etc., así:

```
public aparcamientoEmpresa extends aparcamiento {
      private String nombreEmpresa;
      private cocheDirector = new cocheDeportivo(…);
      public aparcamientoEmpresa(){}
      //etc.
      //para llamar a algún método en una clase hay que especificar
      //la variable de la instancia...
      cocheDirector.setCapacidadMotor("4.5l");
}
```

3.- Descripción de la Práctica

La práctica del presente curso va a estar basada en el legendario arcade "Pac-Man". Esto nos servirá para estudiar y practicar los mecanismos de la Programación Orientada a Objetos y hacer uso de sus características gráficas.

Historia del Juego

Según la descripción en Wikipedia, Pac-Man es un videojuego arcade creado por el diseñador de videojuegos Toru Iwatani de la empresa Namco (basado supuestamente en la forma de una pizza con un trozo faltante) a principios de los años 1980.

Modo de Juego

El protagonista del videojuego Pac-Man es un círculo amarillo al que le falta un sector por lo que parece tener boca. Aparece en laberintos donde debe comer puntos pequeños (llamados Pac-dots en inglés), puntos mayores y otros premios con forma de frutas y otros objetos. El objetivo del personaje es comer todos los puntos de la pantalla, momento en el que se pasa al siguiente nivel o pantalla. Sin embargo, cuatro fantasmas o monstruos, Shadow (Blinky), Speedy (Pinky), Bashful (Inky) y Pokey (Clyde), recorren el laberinto para intentar comerse a Pac-Man. Estos fantasmas son, respectivamente, de colores rojo, rosa, cyan y naranja. Los fantasmas no son iguales, así mientras Blinky es muy rápido, y tiene la habilidad de encontrar a Pac-Man en el escenario, Inky es muy lento y muchas veces evitará el encuentro con Pac-Man.

Hay cuatro puntos más grandes de lo normal situados cerca de las esquinas del laberinto nombrados en inglés Power Pellets, y que proporcionan a Pac-Man la habilidad temporal de comerse a los monstruos (todos ellos se vuelven azules mientras Pac-Man tiene esa habilidad). Después de haber sido tragados, los fantasmas se regeneran en "casa" (una caja situada en el centro del laberinto). El tiempo en que los monstruos permanecen vulnerables varía según la pantalla, pero tiende a decrecer a medida que progresa el juego, y al cabo de muchas pantallas los puntos especiales no tienen ningún efecto sobre los fantasmas.

Además de comer los puntos, Pac-Man puede obtener puntuación adicional si se come alguno de los objetos que aparecen dos veces por pantalla justo debajo de la caja en el centro del laberinto de donde salen los monstruos. El objeto cambia cada pantalla o dos, y su valor en puntos aumenta, de forma que dos cerezas (el premio de la primera pantalla) valen 100 puntos, mientras que el último objeto, la llave, vale 5.000.

Fantasmas

Los fantasmas están limitados por el laberinto de la misma manera que Pac-Man, pero por lo general se mueven ligeramente más rápido que el jugador, aunque se vuelven más lentos cuando se activan los puntos especiales de las esquinas.

Blinky, el fantasma rojo, aumenta su velocidad después de que un cierto número de puntos sean comidos (este número disminuye en niveles más altos).

Pinky. Rodea los obstáculos al contrario de las manecillas del reloj.

Inky. No es tan rápido como Blinky pero actúa de manera errática así que es difícil predecir cómo va a reaccionar.

Clyde. Él no persigue a Pac-Man, si no que deambula sin una ruta especifica.

Premios

A lo largo del juego, Pac-Man puede encontrar diversos premios:

Nivel 1: Cereza 50 puntos.

Nivel 2: Fresa 300 puntos.

Niveles 3 y 4: Naranja 500 puntos.

Niveles 5 y 6: Manzana 700 puntos.

Niveles 7 y 8: Uvas 1000 puntos.

Niveles 9 y 10: Galaxian 2000 puntos.

Niveles 11 y 12: Campana 3000 puntos.

Niveles 13 al 255: Llave 5000 puntos.

En cada nivel aparecen dos veces los premios.

Si Pac-Man pierde una vida cuando aparece un premio, este desaparece a la vida siguiente.

Puedes ver el juego en acción en este vídeo en You Tube:

http://www.youtube.com/watch?v=XlPq7dgQrQ0

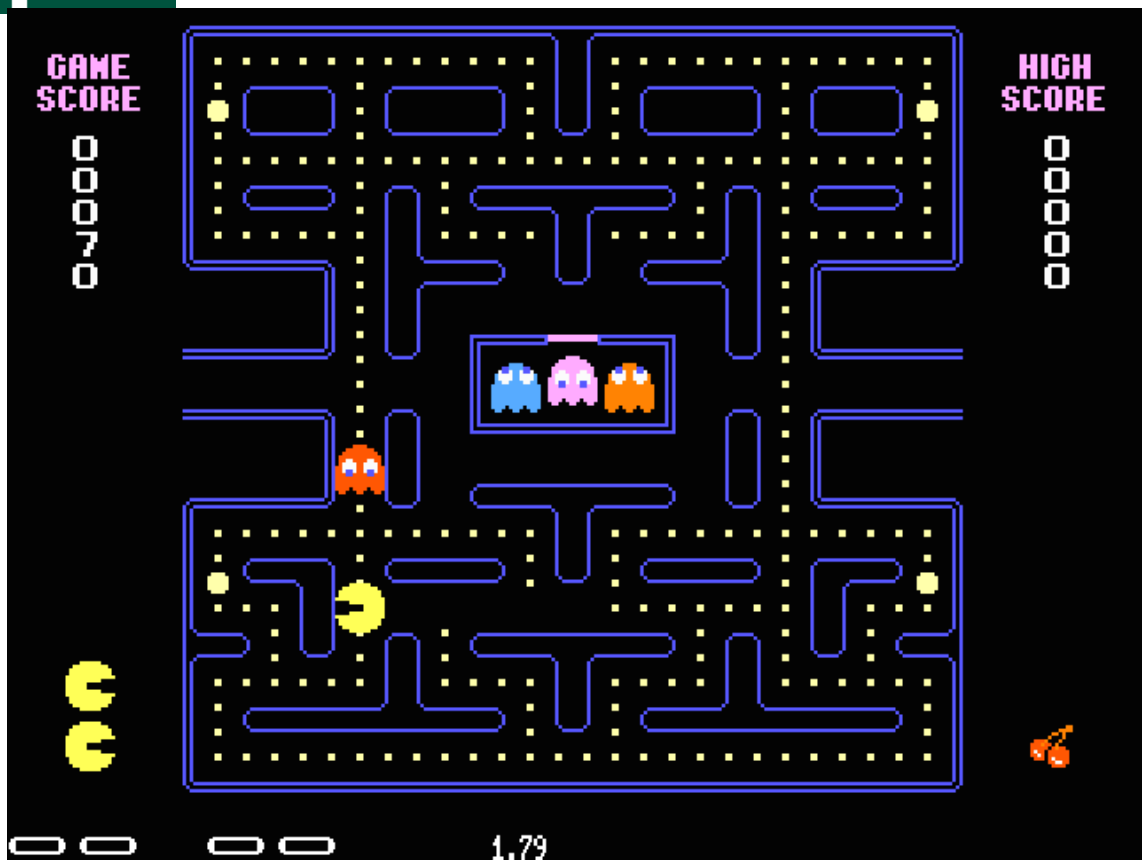


Figura 1. Imagen del juego original

Implementación Obligatoria

El alumno deberá implementar un juego del estilo Pac-Man cumpliendo los siguientes requisitos:

1- El juego constará de un solo nivel donde el jugador deberá comer todos los puntos de la pantalla.
2- El jugador controlará a Pac-Man y dispondrá de 1 vida.
3- Los fantasmas serán controlados por el ordenador teniendo en cuenta el comportamiento diferente de cada uno.
4- Pac-Man podrá moverse (Utilizando las flechas del teclado) arriba (Tecla Up), abajo (Tecla Down), izquierda (Tecla Left) y derecha (Tecla Right). Así mismo podrá pausar el juego pulsando la tecla "P".
5- El área de movimiento permitido para Pac-Man y los fantasmas será el mapa del único nivel disponible.
6- Será necesario comprobar que tanto Pac-Man como los fantasmas no superen los límites del mapa.
7- Los caminos del mapa solo permiten el paso de un individuo al mismo tiempo, por tanto habrá que tener en cuenta las colisiones.
8- Los fantasmas deben implementar comportamientos diferentes:
    a. Blinky, el fantasma rojo, buscará colisionar con Pac-Man. Para acercarse a Pac-Man calculará la distancia (por ejemplo medido en filas y columnas) e intentará primero acercarse verticalmente y luego horizontalmente.
    b. Pinky. Buscará colisionar con Pac-Man. Para acercarse a Pac-Man

calculará la distancia (por ejemplo medido en filas y columnas) e intentará primero acercarse horizontalmente y luego verticalmente.

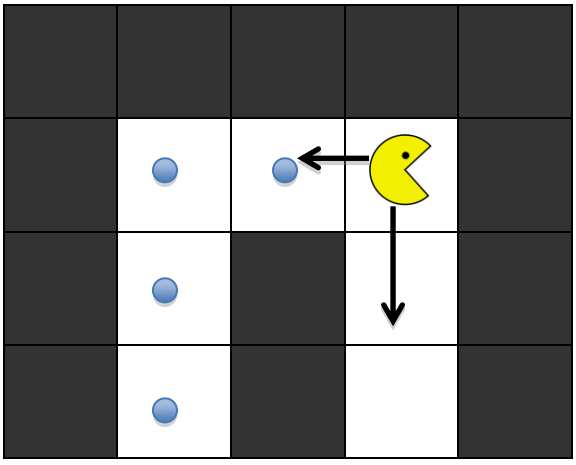
   c. Clyde. Él no persigue a Pac-Man, si no que deambula sin una ruta especifica.

9- Se deberán de detectar dos tipos de colisiones.
   a. Las colisiones entre Pac-Man y los fantasmas, lo que supondrá la pérdida de una vida o el final del juego en caso de no disponer de más vidas.
   b. Las colisiones entre los fantasmas, que supondrá un cambio de dirección en los fantasmas involucrados.
10- Habrá cuatro puntos más grandes de lo normal situados cerca de las esquinas del laberinto y proporcionarán a Pac-Man la habilidad temporal (5 segundos) de comerse a los fantasmas (todos ellos se vuelven azules mientras Pac-Man tiene esa habilidad). Después de haber sido tragados, los fantasmas se regeneran en "casa de fantasmas".
11- Será necesario implementar un contador con los puntos obtenidos en cada momento, teniendo en cuenta los objetos comidos. Un punto pequeño supone 10 puntos. Comer un fantasma 100 puntos.
12- Si el jugador finaliza el nivel del juego deberá aparecer un mensaje de felicitación y se volvería a mostrar la página inicial.

Recomendaciones

Dado que el juego a desarrollar dispone de un mapa por el cual deben circular los elementos móviles del juego (Pac-Man y fantasmas) es necesario planificar su implementación para evitar problemas. A pesar de que las decisiones de implementación son libres para cada estudiante, desde el equipo docente recomendamos una alternativa que podría facilitar su desarrollo. El mapa podría ser considerado como una matriz A x B, en la que cada celda sería controlada por un código (por ejemplo):

- “-1” Para la pared.
- “0” Para una celda vacía.
- “1” Para un punto pequeño.
- “2” Para un punto grande.



Esta alternativa hará que los gráficos se muevan por la pantalla de manera más brusca. Otra aproximación para la resolución de la práctica orientada a que los gráficos se muevan más suavemente se puede llevar a cabo utilizando la detección gráfica de colisiones y el movimiento pixel a pixel explicada en los videotutoriales del curso.

4.- Plan de Trabajo

Para realizar la práctica se seguirá el siguiente método de trabajo:

- En primer lugar se leerá detenidamente el enunciado de esta práctica.
- A continuación hay que diseñar, utilizando un paradigma orientado a objetos, los elementos necesarios para la aplicación explicada en el apartado anterior. Debe hacerse uso de los mecanismos de herencia siempre que sea posible. Se valorará un buen diseño que favorezca la reutilización de código y facilite su mantenimiento.
- El alumno deberá implementar una interfaz gráfica en Swing que permita interactuar con el juego a partir de las clases y la lógica definida. No es necesario hacer uso de hilos para implementar la práctica.
- El código estará debidamente comentado.
- La clase principal que abre la aplicación deberá llamarse "pacman.class".
- A modo de indicación, se muestra una posible planificación temporal para desarrollar la práctica:
    - **Semana 2 / 3**: Tareas iniciales de desarrollo de la práctica. Planteamiento inicial de la práctica. Identificación de las clases.
    - **Semana 4 / 5 / 6**: Establecer el conjunto de atributos y métodos asociados a cada una de las clases identificadas y definidas anteriormente. Establecimiento de los diferentes constructores. Desarrollo de los principales métodos. Inicio de labores de codificación.
    - **Semana 7 / 8**: Refinamiento de clases. Establecimiento de estructuras de almacenamiento auxiliares necesarias. Labores de codificación de métodos.
    - **Semana 9 / 10**: Documentación del código generado. Establecimiento del plan de pruebas de la práctica.
    - **Semana 11 / 12**: Verificación y validación de la práctica. Establecimiento de estrategias de control de errores y excepciones. Mantenimiento de código y eliminación de errores detectados en la fase de verificación y validación.
    - **Semana 13**: Ultimas pruebas. Generación de documentos finales asociados a la práctica.

5.- Material a Entregar

Memoria: La memoria constará de los siguientes apartados:

- Portada con título "Práctica de Programación Orientada a Objetos – Junio 2014" y los datos del alumno: Nombre, Apellidos, dirección de correo electrónico y teléfono.
- Análisis de la aplicación realizada, mostrando el funcionamiento del programa, estrategias implementadas, decisiones de diseño establecidas y, en general, toda aquella información que haga referencia a las diferentes decisiones tomadas a lo largo del desarrollo de la práctica, junto a una justificación de dichas decisiones.
- Diagrama de clases, detallando claramente el tipo de relación entre ellas (uso, agregación, herencia, ...).
- Un texto en el que se describa cada clase/objeto, justificación de su existencia, métodos públicos que contiene y funcionalidad que realizan.
- Anexo con el código fuente de las clases implementadas.

Código: incluyendo todos los ficheros *.java y *.class, así como la memoria en formato electrónico (preferiblemente html o pdf). El soporte estará libre de virus. No se corregirá ni se tendrá en cuenta ninguna práctica que esté infectada por un virus.

Una vez terminada la práctica el alumno tiene que hacer un archivo comprimido (rar o zip) de la memoria y el código y, en el apartado de "Entrega de trabajos", subir una copia a la plataforma aLF según las instrucciones puestas en el curso virtual.

6.- Normas de Realización de la Práctica

1- La realización de la práctica es obligatoria. Para aprobar la asignatura es necesario tener la parte práctica aprobada.
2- Aunque si bien el desarrollo de aplicaciones Orientadas a Objetos usando el lenguaje de programación Java no requiere el uso concreto de ningún entorno de desarrollo, pero se recomienda desarrollar esta práctica íntegramente empleando el entorno de desarrollo BlueJ, que es el que se muestra en el libro de texto básico de la asignatura. En el examen podría hacerse referencia a este entorno concreto.
3- Después de un trabajo inicial de diseño, las prácticas se hacen en los centros asociados supervisadas por el tutor de la asignatura. El trabajo a realizar con la práctica tiene dos partes: el diseño y la implementación. Cada alumno deberá llevar su diseño el día en que se realicen las prácticas para que el tutor lo supervise y el alumno puede realizar la implementación.
4- La práctica es individual. Las prácticas cuyo código coincida total o parcialmente con el de otro alumno serán motivo de suspenso para todos los implicados (copiadores y copiados), no pudiéndose examinar ninguno de ellos en el presente curso académico.
5- La detección de copias implica el suspenso de la parte práctica de la asignatura en las dos convocatorias (Junio y Septiembre).
6- Cada tutor establecerá unas fechas para la realización de la práctica. El tutor puede

organizar las sesiones que le parezca necesarias para la práctica pero tiene que haber al menos una sesión presencial obligatoria. El tutor entrará en el espacio virtual de la asignatura dentro de aLF, antes del 1 de junio, para meter las notas para sus alumnos.

7- No habrá sesión extraordinaria de prácticas ya que la asignatura ya debe estar implantada en todos los centros asociados. En caso de que algún alumno no tuviera tutor, deberá dirigirse a cualquier otro centro asociado donde se imparta la asignatura.
8- El equipo docente tendrá en cuenta prácticas con notas altas para aquellos alumnos cuyo examen esté cercano al aprobado.
9- El alumno debería dirigirse a su tutor para cualquier duda que tenga sobre su práctica y solamente al equipo docente (por correo electrónico) en el caso de que su tutor no pueda resolver su problema. En este caso pedimos al alumno que, además de sus datos personales, nos envíe el nombre del centro asociado en el que está matriculado y el de su tutor.
10- Evidentemente se puede usar los foros para realizar consultas a los compañeros pero nunca para intercambiar código.

7.- Normas para los Tutores

Como se puede apreciar, el papel del tutor es fundamental en todos los aspectos de la práctica tanto el planteamiento del problema, el diseño Orientado a Objetos del programa, su desarrollo y su depuración. Tratándose de una asignatura obligatoria, cada alumno debería tener acceso a un tutor.

Los tutores deben seguir los siguientes pasos:

- Ayudar a los alumnos al principio del curso con el planteamiento de la práctica.
- Indicar a los alumnos que habrá al menos una sesión obligatoria de seguimiento de la práctica.
- Una vez terminada y entregada la práctica, el tutor tiene que entrar en el espacio virtual de la asignatura dentro de aLF, antes del 1 de junio, para meter las notas para sus estudiantes.
- Comunicar la calificación a sus alumnos.

8.- Centros Asociados vs. Prácticas en Asignaturas Obligatorias

Las prácticas son esenciales en las titulaciones de Informática porque, entre otras cosas, permiten a los alumnos adquirir conocimientos importantes sobre los aspectos más aplicados de ciertas asignaturas, lo cual resulta de gran relevancia e interés a la hora de acceder a un puesto laboral relacionado con la Informática. Para

orientar y ayudar a los alumnos, así como para comprobar que realmente un alumno ha realizado su práctica de forma satisfactoria, ésta se debe realizar en un Centro Asociado bajo la supervisión de un tutor, quien decide, en última instancia, la forma en la cual se organiza el desarrollo de la misma en su Centro Asociado (existencia o no de sesiones presenciales obligatorias, forma de entrega, etc.).

De vez en cuando sucede que un alumno se pone en contacto con un Equipo Docente del Departamento de Lenguajes y Sistemas Informáticos (L.S.I.) porque se ha matriculado en una asignatura obligatoria en un Centro Asociado que no le proporciona un tutor para supervisar la práctica, aún cuando se le ha permitido matricularse. El alumno busca en el Equipo Docente que se le proporcione una solución a este problema, como por ejemplo, la posibilidad de asistir a unas sesiones extraordinarias de prácticas en la Sede Central de la U.N.E.D. en Madrid o la posibilidad de realizar la práctica por su cuenta en casa, enviándola a continuación al Equipo Docente para su corrección. Sin embargo, los Equipos Docentes de L.S.I. no disponen de recursos para poder llevar a cabo ninguna de estas dos alternativas.

Un Centro Asociado que ha permitido a un alumno matricularse en una asignatura obligatoria de una carrera de Informática debería ayudarle a encontrar una solución al problema de la realización de las prácticas. Si se trata de una asignatura donde no se han matriculado muchos alumnos, quizás el centro no cuente con recursos para proporcionar un tutor específicamente para la asignatura. Si hay otro Centro Asociado cerca que dispone de tutor, quizás el alumno pueda realizar la práctica allí. Pero si no es así, el Centro Asociado debería proporcionar un tutor para supervisar y corregir las prácticas de sus alumnos. Lo más razonable sería que fuera un tutor de otra asignatura de Informática en el mismo Centro el que hiciera la sesión de prácticas para los alumnos de la asignatura en cuestión, y al final de la sesión evaluara los trabajos de los alumnos, según las pautas marcadas por el Equipo Docente, haciendo llegar a éste las calificaciones otorgadas.

Por lo tanto, un alumno que tras haberse matriculado en una asignatura obligatoria en un Centro Asociado, se encuentre con que el centro no tiene tutor para dicha asignatura, debería dirigirse al Director del Centro Asociado, para solicitar de él una solución, tal como se ha presentado aquí, es decir, alguien que pueda supervisar y corregir su práctica con plenas garantías. En el caso de que el Director no le proporcione una solución, el alumno debería comunicárselo, por escrito, lo antes posible, al Director del Departamento de L.S.I., Dr. D. Julio Gonzalo.