Tetris

Práctica de la Asignatura de Programación Orientada a Objetos
Escenario para el Curso 2010/2011 – Febrero de 2011 – Versión 1.2

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 y una dirección en Internet relativa a la práctica y a otros aspectos de la asignatura, que es http://www.lsi.uned.es/poo.

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
     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 ser diseñar e implementar una versión del conocido juego Tetris. Esto servirá para estudiar y practicar los mecanismos de la Programación Orientada a Objetos.

Historia

Tetris es un juego desarrollado por Alekséi Pázhitnov. Éste juego se inspira en otro juego previo que se llamaba pentaminós. El número de cuadros que compone cada una de las piezas (siempre constante a cuatro) es lo que da nombre al juego (tetra, de cuatro, da el nombre de tetris). Su éxito vino con su portabilidad al IBM PC y a otras plataformas, como Apple II, Commodore 64, Atari ST y Sinclair ZX Spectrum.

Tetris ha sido históricamente uno de los videojuegos más versionados. Después del éxito de este juego, muchos otros trataron de imitarlo. Existe incluso una versión gratuita muy popular en Internet denominada TetriNET, que proporciona una versión multijugador en arquitectura cliente-servidor en el que se pueden enfrentar a través de la red de 2 a 6 personas con la posibilidad de crear equipos.

Mecánica del Juego

Distintos tetriminos, figuras geométricas compuestas por cuatro bloques cuadrados unidos de forma ortogonal, caen de la parte superior de la pantalla. El jugador no puede impedir esta caída pero puede decidir la rotación de la pieza (0°, 90°, 180°, 270°) y en qué lugar debe caer. Cuando una línea horizontal se completa, esa línea desaparece y todas las piezas que están por encima descienden una posición, liberando espacio de juego y por tanto facilitando la tarea de situar nuevas piezas. La caída de las piezas se acelera progresivamente. El juego acaba cuando las piezas se amontonan hasta salir del área de juego.

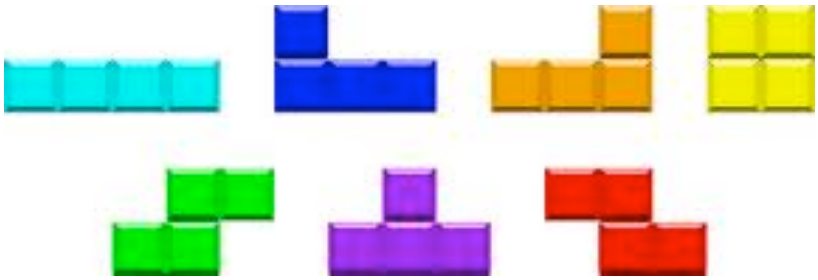

Figura 1: Posibles Piezas del Juego

Para simplificar el juego, se van a hacer las siguientes suposiciones:

- La pantalla está formada por 25 líneas (contadas desde la parte inferior a la parte superior).
- Puesto que las piezas caen de la zona superior, y aparecen inicialmente en el formato que se muestra en la figura, el área de juego efectivo son 23 líneas (restando dos líneas por la colocación inicial de las piezas).
- El ancho de la pantalla será de 12 casillas.
- Una pieza nueva no aparece hasta que no ha caído otra.
- No se produce aceleración en la caída de piezas a lo largo del juego. Éstas caerán una velocidad de 1 fila por segundo, de tal manera que una pieza tarda 23 segundos en caer desde la parte superior a la inferior, si no se producen rotaciones.
- Como máximo el juego deja hacer cuatro rotaciones por segundo.

- Las piezas se desplazarán a derecha e izquierda con los cursores, y rotarán sólo en un sentido (en el de las agujas del reloj), mediante la flecha superior de los cursores.
- No se habilitará la opción de dejar caer la pieza como ocurre en el juego original
- No se establecerá sistema de puntuaciones
- No se establecerán niveles diferentes en el juego.
- Las piezas siempre aparecen por la parte central del tablero.

Ejemplo de pantalla posible de juego:

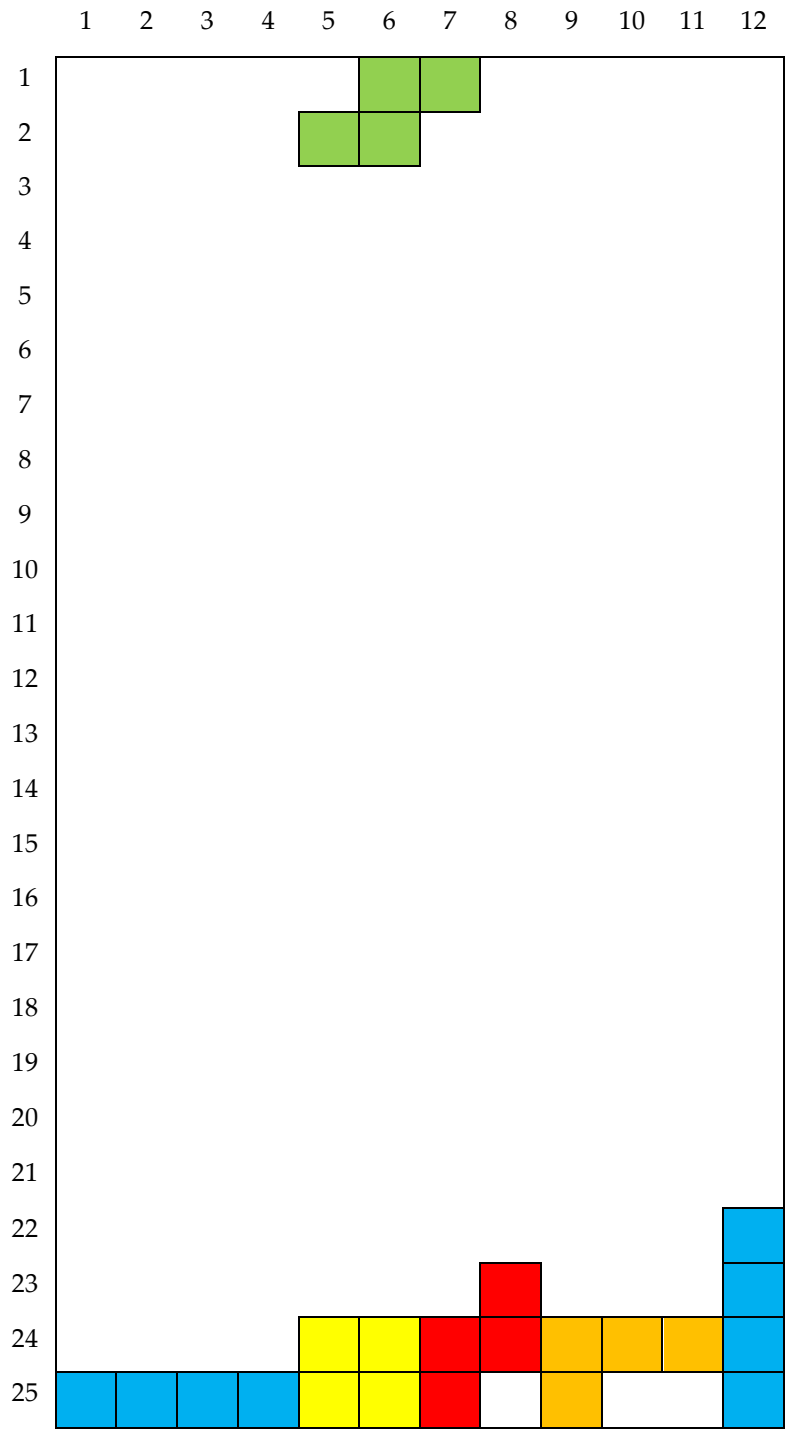

Figura 2: Posibles Pantalla del Juego

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 "tetris.class".
- A modo de indicación, se muestra una posible planificación temporal para desarrollar la práctica:
    - Semana 1 / 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 2011" 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. Sólo se evaluará el examen si la práctica ha sido previamente 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, está práctica ha de desarrollarse íntegramente empleando el entorno de desarrollo BlueJ, que es el que se muestra en el libro de texto básico de la asignatura.
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. Cada tutor establecerá unas fechas para la realización de la práctica. El tutor puede organizar las sesiones que le parece necesarias para la práctica pero tiene que haber al menos dos sesiones presenciales obligatorias: una a la mitad del curso para comprobar que los alumnos han hecho bien el diseño de la jerarquía de clases necesarias para resolver el problema de la práctica, y otra sesión al final del curso para evaluar tanto el programa como la memoria. 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.
6. 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.
7. El equipo docente tendrá en cuenta prácticas con notas altas para aquellos alumnos cuyo examen esté cercano al aprobado.

8. 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.
9. 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 dos sesiones obligatorias de seguimiento de la práctica: una a la mitad del curso para comprobar el diseño de la jerarquía de clases necesarias para resolver el problema de la práctica, y otra sesión al final del curso para evaluar tanto el programa como la memoria.
- 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.