SEGUNDA PRÁCTICA

Programación

Curso 2006-2007

Ingeniería en Informática

Universidad Carlos III de Madrid

1. Instrucciones generales

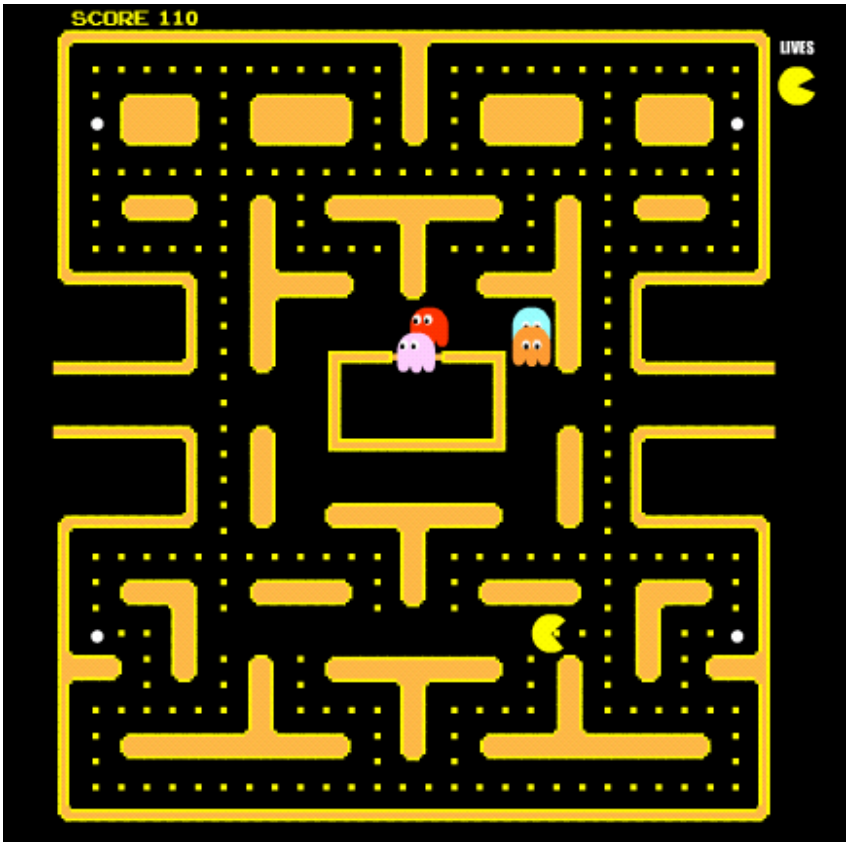
- Durante este curso se deberán realizar **tres prácticas**, cuyas fechas de entrega se pueden encontrar en la página Web de la asignatura: (http://galahad.plg.inf.uc3m.es/~docweb/pr-inf/)
- Las prácticas se realizarán en Java, usando la máquina virtual de Java 2 (versión 1.5) y el entorno de desarrollo gráfico Eclipse (versión 3.1.1)
- Estas prácticas deberán realizarse **obligatoriamente en grupos de dos** alumnos.
- Las **tres prácticas están relacionadas** entre sí: la segunda es continuación de la primera y utiliza el código creado para ésta, y a su vez la tercera es continuación de la segunda. No obstante, para facilitar la realización de las prácticas y homogeneizar los resultados, **después de la entrega de la primera práctica** los profesores pondrán a disposición de los alumnos, en la página Web de la asignatura, el **código Java que se debe usar como punto de partida en la segunda práctica**. De igual forma, después e la entrega de la segunda práctica, se publicará el código Java de partida de la tercera.
- **Se puede entregar una práctica aunque no se haya entregado la anterior**, partiendo siempre del código Java publicado por los profesores.
- Cada práctica se calificará con un máximo de **un punto (1 punto)**.
- Las prácticas tendrán una **parte obligatoria** y **partes opcionales** que servirán para **subir nota**. Si un alumno realiza solamente la parte obligatoria podrá obtener la nota máxima, aunque no haya realizado ninguna de las partes opcionales. La nota máxima será en cualquier caso 1 punto.

2. Práctica segunda

2.1 Introducción

En esta segunda práctica se va a continuar con la implementación del juego comecocos (también llamado **PacMan**), añadiendo nuevas funcionalidades a las de la primera práctica. Los alumnos deben partir del **código Java publicado por los profesores**, en **ningún caso** deben utilizar como punto de partida **su propio código** de la primera práctica, aunque podrán usar partes de él si lo estiman conveniente.

Como se indicó en la práctica anterior, en Internet se pueden encontrar numerosas versiones de este juego, como por ejemplo la que existe en http://www.minijuegos.com/juegos/jugar.php?id=2135 cuya pantalla se incluye abajo. El alumno puede probar alguna de estas versiones para entender mejor el desarrollo del juego y lo que se pretende realizar en las prácticas de este curso.



Nota: El protagonista siempre tiene una dirección de movimiento, y si no se pulsa ninguna tecla para modificar dicha dirección sigue avanzando hasta chocar con algo que le impida seguir.

2.2 Objetivos docentes de la práctica

Los objetivos docentes de esta práctica son:

- Afianzar los conocimientos adquiridos en la práctica anterior.
- Utilizar la herencia para crear nuevas clases a partir de clases ya existentes.
- Comprender y utilizar los distintos algoritmos de búsqueda, inserción y ordenación en listas estudiados en las clases teóricas.
- Implementar algoritmos recursivos para la resolución de tareas.

2.3 ¿Qué se pide?

Todas las clases creadas deberán pertenecer al paquete (package) "**practicas**".

- Modificar las clases proporcionadas e implementar las que resulten necesarias para introducir el concepto de herencia estudiado en la asignatura. En concreto se creará una clase **Individuo** que sea la superclase de las clases **Enemigo** y **Protagonista**. Mover las variables y/o métodos que puedan tener en común ambas clases (Enemigo y Protagonista) a esta clase padre de la que ambas heredarán.
- Modificar la clase **Enemigo** de tal forma que permita incluir las siguientes características:
    - Asignar un color y un nombre a cada uno de los enemigos.
    - Implementar un **algoritmo recursivo** para que en cada turno cada Enemigo encuentre un posible camino que le lleve a la posición del Protagonista. El enemigo debe moverse siguiendo ese camino.
    - Cada uno de los enemigos tendrá un atributo interno que almacene en cada momento la distancia a la que está del protagonista (medida en número de casillas que tendría que moverse). Para ello utilizará el método recursivo anterior.
- Implementar una clase, que se denominará **Lista**, y que tendrá como atributo, además de los que los alumnos consideren oportunos, una lista de enemigos implementada en forma de array.
    - La clase Lista tendrá al menos un constructor que recibirá como parámetro un array de Enemigos.
    - Deberá incluir métodos para buscar a un Enemigo en la lista según su nombre, empleando tanto búsqueda secuencial como binaria.
    - Deberá incluir una serie de métodos que permitan ordenarla por distancia al protagonista siguiendo cada uno de los cinco algoritmos de ordenación estudiados en las clases teóricas de la asignatura.
- Crear al inicio de la partida un objeto de la clase **Lista** que contenga un array en el que se encuentren todos los Enemigos presentes en una partida. Esta lista se ordenará cada turno por distancia al protagonista mediante un algoritmo de ordenación seleccionado al inicio.

- Modificar el método correspondiente de la clase **Tablero** de manera que la salida por pantalla muestre, a lo largo de la partida, un marcador con los nombres de los Enemigos y sus correspondientes puntuaciones ordenadas de mayor a menor.
- Realizar un programa de prueba en una clase **Practica2,** en el que se cree un tablero de juego y se permita jugar al juego del comecocos. Los caracteres correspondientes a cada uno de los nuevos elementos del tablero son los siguientes:
  - **Protagonista: "@"**
  - **Enemigos: La inicial de su nombre** (los enemigos no deberían tener nombres que empezaran por la misma inicial)
  - **Muro: "#"**
  - **Guarida: "O"** (sólo si se hace la parte opcional)
  - **Punto: "."**
  - **Punto Especial: "*"**
  - **Fruta: "$"** (sólo si se hace la parte opcional)
  - **Vacía: " "**
- La posición inicial del tablero será la siguiente:

```
############################
#............##............#
#.####.#####.##.#####.####.#
#*####.#####.##.#####.####*#
#.####.#####.##.#####.####.#
#..........................#
#.####.##.########.##.####.#
#.####.##.########.##.####.#
#......##....##....##......#
######.####  ##  ####.######
######.####  ##  ####.######
######.##          ##.######
######.## ##OOOO## ##.######
######.## ######## ##.######
######.   ########   .######
######.## ######## ##.######
######.## ######## ##.######
######.##          ##.######
######.## ######## ##.######
######.## ######## ##.######
#............##............#
#.####.####..##..####.####.#
#.####.####..##..####.####.#
#*..##.......@........##..*#
###.##.##.########.##.##.###
###.##.##.########.##.##.###
#......##....##....##......#
#.##########.##.##########.#
#.##########.##.##########.#
#..........................#
############################
```

2.4 Actividades optativas

- Incluir dos nuevos tipos de **Casilla**: Fruta y Guarida:
  - **Casilla Fruta:** Si el Protagonista ocupa dicha casilla, el Protagonista se comerá la fruta, recibiendo una bonificación especial en su puntuación (diez veces el valor de un punto). Aparecerá una única fruta por pantalla en un lugar aleatorio del tablero, una vez que el Protagonista se haya comido la mitad de los puntos normales del tablero.
  - **Casilla Guarida:** Cada uno de los enemigos tiene asignada una casilla Guarida específica en la que aparecerá tanto al comienzo del juego como cada vez que sea eliminado en el Modo Pánico. Estas casillas serán tratadas como muros, a efectos de movimiento y colisión, por el Protagonista y los Enemigos en el transcurso normal del juego.
- Diseñar un **comportamiento "inteligente" diferente** para cada uno de los Enemigos, de manera que se aproximen al Protagonista (y huyan de él en el modo pánico) utilizando diversas funciones.
- Construir un sistema mediante el cual el jugador pueda jugar **varias pantallas**. Una vez que el Protagonista complete una pantalla, comiéndose todos los puntos del tablero, comenzará una nueva pantalla (que puede basarse en el tablero original o en otro). El sistema inicializará todos sus valores como al comienzo del juego, a excepción del número de vidas restante del Protagonista, y las puntuaciones del Protagonista y los Enemigos.

2.5 Entrega de la práctica

La práctica se deberá entregar antes del **lunes 14 de mayo a las 20:00 horas.** La documentación a entregar consta de:

- Una **memoria explicativa** de los desarrollos realizados, que en general NO debe contener listados de código salvo los necesarios para la correcta explicación del mismo. Deberá entregarse **impresa en papel,** preferiblemente a doble cara. Se podrá entregar directamente a los profesores depositar en los casilleros de los profesores de prácticas. La memoria constará de:
  - Portada: con el nombre, apellidos, número de alumno, grupo (81, 82, 83) y campus (Leganés o Colmenarejo) de cada uno de los alumnos.
  - Índice: tabla de contenido del documento.
  - Manual Técnico: descripción (que no el código) de las clases implementadas, sus atributos y sus métodos. Tipo de relación existente entre todas ellas. Se deben justificar las decisiones tomadas.
  - Manual de Usuario: descripción del uso de los programas realizados.

  - Mejoras opcionales: descripción de las mejoras opcionales introducidas sobre la propuesta inicial, así como otros requisitos optativos.
  - Observaciones: comentarios, problemas encontrados, críticas constructivas a la práctica2.
- El **código desarrollado**, que se enviará a través de la aplicación "Aula Global". Este código consta de:
  - Los **archivos .java** creados por los alumnos (se podrán incluir los archivos .class u otros).
  - La **memoria en formato .pdf** (preferiblemente) o .doc, además de impresa.

2.6 Evaluación de la práctica

La práctica se evaluará de acuerdo a los siguientes criterios:

- Claridad y completitud de la **memoria (0,3 puntos).**
  - Presentación, claridad, ortografía: 0,1 puntos.
  - Calidad manual técnico y usuario: 0,1 puntos.
  - Calidad de las decisiones tomadas: 0,1 puntos.
- **Funcionalidad (0,4 puntos)**.
  - Modificaciones en la clase Enemigo: 0,2 puntos
  - Implementación de la clase Lista: 0,1 puntos.
  - Modificaciones en el Tablero para mostrar la lista: 0,1 puntos.
- Calidad y legibilidad del **código generado (0,3 puntos)**.
  - Modularidad, encapsulación y herencia: 0,1 puntos.
  - Calidad de la implementación (ahorro de computación y control de errores): 0,1 puntos.
  - Claridad del código y calidad de los comentarios: 0,1 puntos. Se recomienda encarecidamente a los alumnos que el código esté **comentado** exhaustivamente.
- Temas opcionales:
  - Implementar los nuevos tipos de casilla: 0,1 puntos.
  - Mejorar y diferenciar inteligencia de los enemigos: 0,1 puntos.
  - Implementar sistema de pantallas: 0,1 puntos.
- Es requisito **fundamental** para calificar la práctica que el código entregado **compile sin errores**.