PRIMERA PRÁCTICA

Programación

Curso 2006-2007

Ingeniería en Informática

Universidad Carlos III de Madrid

1. Instrucciones generales

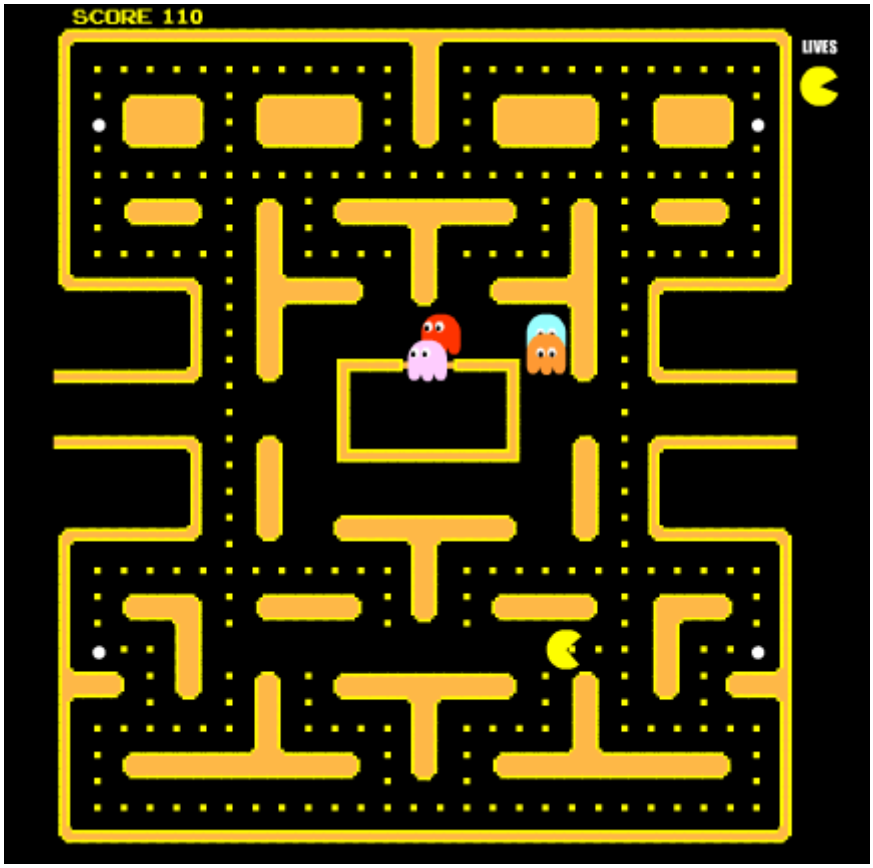
- Durante este curso se deberán realizar **tres prácticas**, cuyas fechas de entrega se pueden encontrar en la página Web de la asignatura: (http://galahad.plg.inf.uc3m.es/~docweb/pr-inf/)
- Las prácticas se realizarán en Java, usando la máquina virtual de Java 5 (versión 1.5) y el entorno de desarrollo 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 de 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 primera

2.1 Introducción

El objetivo de las tres prácticas del curso es que los alumnos creen el clásico juego del **comecocos** (también denominado **PacMan**). El juego consiste en mover al protagonista por un tablero de juego sin que sea alcanzado por los enemigos. Al moverse por el tablero de juego, el protagonista va "comiendo" puntos situados en las casillas de juego, incrementando así su contador de puntos. El juego termina cuando un enemigo alcanza al protagonista (juego perdido) o el protagonista ha conseguido comerse todos los puntos del tablero (juego ganado).

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:

- Que el alumno se familiarice con un entorno gráfico de programación en Java como es Eclipse. El alumno debe ser capaz de instalar una Máquina Virtual de Java y el entorno Eclipse y hacerlos funcionar adecuadamente.
- Que el alumno conozca la sintaxis de java: tipos de datos, instrucciones, tipos de ficheros que se generan.
- Que el alumno cree clases, con sus correspondientes métodos y propiedades, y objetos pertenecientes a esas clases, realizando un programa mediante la técnica de orientación a objetos.

2.3 ¿Qué se pide?

- Todas las clases creadas deberán pertenecer al paquete (package) "**practicas**".
- Definir una clase **Casilla** para representar cada una de las casillas del tablero de juego. Para cada casilla se deberá al menos poder indicar su tipo (muro/vacía/punto/punto especial). La clase casilla deberá proporcionar métodos para:

  - Obtener el tipo de una casilla.
  - Obtener la representación de la casilla en forma de cadena de texto.
  - Fijar o cambiar el tipo de una casilla.

- Definir una clase **Protagonista** para representar al protagonista del juego. La clase deberá contener al menos variables para almacenar la siguiente información acerca del protagonista:
  - Su posición en el tablero de juego.
  - Su dirección actual.

  La clase protagonista deberá al menos incluir un método para mover al protagonista en la dirección actual. Dicho método deberá conocer las casillas del tablero de juego, y la tecla pulsada por el jugador, y realizará la opción indicada según la tecla pulsada por el jugador:
    - W: Para cambiar la dirección del protagonista hacia arriba.
    - S: Para cambiar la dirección del protagonista hacia abajo.
    - A: Para cambiar la dirección del protagonista hacia la izquierda.
    - D: Para cambiar la dirección del protagonista hacia la derecha.
    - Q: Para salir del juego.
    - Cualquier otra tecla no modificará la dirección del protagonista, pero lo moverá en la dirección actual si fuese posible.
- Definir una clase **Enemigo** para representar a cada uno de los enemigos del juego. La clase contendrá al menos variables para contener la siguiente información acerca de un enemigo:
  - Su posición en el tablero de juego.
  - Su dirección actual.
  - Un identificador único de enemigo que se asignará automáticamente a un enemigo cada vez que se cree (valor 1 para el primer enemigo, 2 para el segundo, etc.).

  La clase Enemigo además proporcionará métodos para las siguientes acciones:
  - Obtener el identificador único del enemigo: Método que devolverá el identificador único del enemigo.
  - Mover al enemigo, conociendo las casillas del juego, la posición del protagonista y la posición de los demás enemigos del juego. Este método no deberá ser necesariamente "inteligente", pero el comportamiento mínimo que deberá cumplir un enemigo es el siguiente:
    - Deberá de cambiar de dirección al chocar contra un muro.
    - No deberá poder ponerse en una casilla en la que ya se encuentre otro enemigo.
- Definir una clase **Tablero** para representar el tablero de juego. La clase deberá contener variables para almacenar al menos la siguiente información:
  - Las casillas del tablero de juego.
  - El protagonista del juego.
  - La lista de enemigos del juego.
  - Los puntos de la partida actual.
  - Las vidas que le quedan al protagonista.

  Asimismo, la clase Tablero dispondrá de métodos para realizar al menos las siguientes acciones:
  - Representar el tablero de juego en forma de cadena de caracteres.
  - Comprobar si se ha ganado la partida.

  - Comprobar si se ha perdido la partida.
  - Comprobar si se ha perdido una vida. En este caso, si no se ha perdido la partida, se debe devolver al protagonista y los enemigos a su posición inicial.
  - Realizar el siguiente movimiento del tablero de juego (mover tanto al protagonista como a los enemigos, e incrementar los puntos si fuese necesario).

- Realizar un programa de prueba en una clase **Practica1,** en el que se cree un tablero de juego y se permita jugar al juego del comecocos, con los siguientes requisitos:
  - La forma y posición inicial del tablero será la siguiente ("@" identifica al protagonista, "**%**" a los enemigos, "#" son los muros y "**.**" los puntos):

```
####################
#@.......#........#
#.##.###.#.###.##.#
#..................#
#.##.#.#####.#.##.#
#....#%..#...#...%#
####.###.#.###.####
####.#.......#.####
####...#####...####
####.#.......#.####
####.###.#.###.####
#....#...#...#....#
#.##.#.#####.#.##.#
#..................#
#.##.###.#.###.##.#
#.....%..#........#
####################
```

  - Para cada movimiento, se recibirá una tecla del usuario por pantalla, que podrá ser una de las siguientes:
    - W: Para cambiar la dirección del protagonista hacia arriba.
    - S: Para cambiar la dirección del protagonista hacia abajo.
    - A: Para cambiar la dirección del protagonista hacia la izquierda.
    - D: Para cambiar la dirección del protagonista hacia la derecha.
    - Q: Para salir del juego.
    - Cualquier otra tecla no modificará la dirección del protagonista, pero el juego continuará.
  - En función de la tecla, se realizará el siguiente movimiento del tablero, y se comprobará:
    - Si se ha ganado, se mostrará un mensaje por pantalla y se saldrá de la aplicación.
    - Si se ha perdido, se mostrará un mensaje por pantalla y se saldrá de la aplicación.
    - Si no se ha ganado ni se ha perdido, se continuará con el juego (se volverá a pedir una tecla al usuario, etc.).

2.4 Actividades optativas

- Modificar la "inteligencia" de los enemigos para que intenten moverse primero en la dirección en la que se aproximen al protagonista.
- Implementar el **modo pánico**:
  - Existen unos puntos especiales, más gruesos que los normales, de manera que cuando el protagonista se los come se entra en modo pánico.

  - El modo pánico tiene una duración de 15 movimientos después de haberse comido el punto especial.
  - Durante estos 15 movimientos, el protagonista puede comerse a los enemigos, de manera que éstos desaparecerán del tablero de juego si el protagonista les alcanza.
- Mejorar la "inteligencia" de los enemigos para que en modo pánico huyan del protagonista.

2.5 Entrega de la práctica

La práctica se deberá entregar antes del **jueves 29 de marzo 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 o 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 práctica1.
- 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)**. Se le asignará 0,1 puntos por la total y correcta funcionalidad de cada una de las cuatro clases: Protagonista, Enemigo, Tablero y Practica1.
- Calidad y legibilidad del **código generado (0,3 puntos)**.
  - Modularidad, encapsulación: 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:
  - Modificar inteligencia de los enemigos: 0,1 puntos.
  - Implementar modo pánico: 0,2 puntos.
  - Mejorar inteligencia de los enemigos: 0,1 puntos.
- Es requisito **fundamental** para calificar la práctica que el código entregado **compile sin errores**.

2.7 Códigos de Ayuda

Los siguientes códigos de ejemplo pueden ayudar al alumno a realizar algunas de las labores necesarias para la práctica:

1) Obtención de tecla por parte del usuario:

```
package practicas;

import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;

public class EjemploTecla {
    public static void main(String[] args) {
        System.out.print("Introduce una tecla y pulsa enter: ");
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String teclaPulsada = null;
        try {
            teclaPulsada = br.readLine();
        } catch(IOException ioe) {
            System.out.println("Excepción al leer de teclado");
            System.exit(-1);
        }
        System.out.println("Has pulsado: " + teclaPulsada);
    }
}
```

2) Generación de números aleatorios

```
package practicas;

import java.util.Random;

public class EjemploAleatorio {
    public static void main(String[] args) {
        Random rnd = new Random(System.currentTimeMillis());
        int numero = rnd.nextInt(10);
        System.out.println("Número aleatorio entre 0 y 9: " + numero);
    }
}
```