"R-Type"

Práctica de la Asignatura de Programación Orientada a Objetos

Escenario para el Curso 2012/2013 –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 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 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 "R-Type". 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, R-Type es un videojuego de género matamarcianos creado por Irem en 1987 para las máquinas recreativas o Arcade.

En este juego el jugador controla a un caza espacial llamado R-9 que se caracteriza por un láser que se puede cargar para aumentar la fuerza de impacto contra los enemigos y a la vez, con una cápsula, llamada Force (fuerza) que se deja anclar a la nave y otorga un arma definida de acuerdo con el color o la letra que tiene el Power-Up que se recoge a los siguientes. El poder de "Fuerza" aumenta conforme se recogen más armas. También existen extras para aumentar la velocidad de despliegue de la nave, cápsulas fijas sobre el techo y debajo de la base de R-9 y misiles detectores (homing) como añadido eficiente al disparo que ya poseemos. Lo que más distinguió a R-Type de los juegos de la competencia era la novedad de poder utilizar la cápsula "Force" tanto para atacar como para defenderse de los disparos enemigos y que esta se podía lanzar y separar de la nave para utilizarla como un satélite, además de forma libre, lo que habilitaba al jugador de anclar "Fuerza" en la cola de la nave y así dispara hacia atrás, es decir hacia la izquierda de la pantalla. R-Type fue el único juego que disponía de estas características y de un apartado técnico impecable. Daba igual la cantidad de enemigos o su tamaño en pantalla, R-Type no se ralentizaba ni se dañaba la apariencia en otros objetos que simultáneamente se movían sobre la pantalla. Este era un error muy común en aquellos tiempos pero Irem demostró su maestría en crear un shooter horizontal sin estos errores comunes. Este último hecho solo se entiende para la versión original de R-Type en máquina recreativa. Las conversiones para consolas y ordenadores solían tener algunos fallos pero era debido a que el hardware no cumplía con los requisitos para poder programar a un R-Type vistoso y sin ralentizaciones.

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

http://www.youtube.com/watch?v=pVWtI0426mU

Figura 1. Imagen del juego original

Implementación Obligatoria

El alumno deberá implementar un juego del estilo R-Type satisfaciendo los siguientes requisitos:

1- El juego comenzará con una pantalla de bienvenida a partir de la cual se podrá seleccionar el modo de juego (FACIL, NORMAL, COMPLICADO, IMPOSIBLE) y comenzar a jugar.
2- El juego constará de un único nivel donde el jugador deberá acabar con una horda de naves alienígenas. El número de alienígenas con los que acabar dependerá del modo de juego seleccionado. Fácil=10, Normal=15, Complicado=20, Imposible=30.
3- El jugador controlará la nave aliada y dispondrá de 1 sola vida.
4- Las naves alienígenas serán controladas por el ordenador.
5- Las naves alienígenas no disparan.
6- No hay que implementar relieve. Es decir, no hay que mostrar ningún tipo de suelo o techo como en el juego original.
7- La nave aliada podrá moverse arriba (Tecla Q), abajo (Tecla A), izquierda (Tecla O) y derecha (Tecla P). Así mismo podrá disparar su laser utilizando la tecla ESPACIO.
8- El área de movimiento permitido para la nave será toda la pantalla, aunque habrá que comprobar que la nave no salga de estos límites.
9- El disparo que realiza la nave aliada es continuo, es decir, no es necesario esperar a que el misil disparado abandone la pantalla para que la nave aliada pueda volver a disparar.
10- La nave aliada sólo puede realizar un tipo de disparo que se desplazará horizontalmente hacia la derecha de la pantalla, sin variar su trayectoria y a velocidad constante.
11- Las naves alienígenas se mueven a velocidad constante y podrán ser de dos tipos:
    a. **Nave Alienígena Tipo A.** Aparecen por la parte derecha de la pantalla y se mueven horizontalmente hacia la izquierda a velocidad constante sin variar su trayectoria, es decir, su coordenada “y” no varia en todo el

desplazamiento.

b. **Nave alienígena Tipo B.** Aparecen por la parte derecha de la pantalla y se mueven horizontalmente hacia la izquierda a velocidad constante. La principal diferencia con las Naves de Tipo A es que éstas pueden variar su trayectoria, es decir, en su desplazamiento horizontal pueden variar su coordenada "y" de manera aleatoria.

12- La velocidad a la que se mueven las naves alienígenas dependerá del modo de juego seleccionado. Todas las naves se mueven a la misma velocidad.

13- Cuando las naves alienígenas alcancen la parte izquierda de la pantalla volverán a aparecer por la parte derecha de ésta.

14- Se deberán de detectar dos tipos de colisiones.

a. Las colisiones entre la nave aliada y las naves alienígenas, lo que supondrá el final del juego.

b. Las colisiones entre los misiles disparados por la nave aliada y las naves alienígenas, lo que supondrá la destrucción de la nave alienígena contra la que ha chocado el misil.

15- Si el jugador finaliza el nivel del juego deberá aparecer un mensaje de felicitación y se volvería a mostrar el menú inicial.

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 "RType.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 2013" 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- El proyecto final deberá entregarse en formato BlueJ para que el tutor de prácticas pueda verificar su correcta compilación. Por esta razón recomendamos al alumno que realice la práctica utilizando este entorno. Si por alguna razón algún alumno deseara utilizar otro IDE podrá hacerlo siempre y cuando el proyecto final lo entregue en formato BlueJ. Además, en estos casos el equipo docente no puede garantizar el apoyo del profesor tutor ya que el IDE oficial de la asignatura es BlueJ.

3- 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. Si algún alumno quisiera utilizar otro entorno de desarrollo puede hacerlo pero, una vez concluido

4- 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.
5- 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.
6- 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.
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 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.