PROGRAMACION - PRACTICA 1

Objetivos

- Uso de clases predefinidas.
- Manejo de referencias.
- Creación de clases sencillas.

Recursos

Proyecto Eclipse que contiene, entre otros ficheros y carpetas:

- Paquete `es.upm.dit.prog.practica1`
- `PruebaPractica1.java`: programa que prueba si funcionan bien las clases.
- `PruebaInteractiva1.java`: programa que recibe órdenes por teclado para probar el funcionamiento.

Descripción

Los drones son aeronaves no tripuladas y, entre otros usos, son útiles en la detección temprana y el control eficiente de incendios forestales. Equipados con cámaras térmicas y sensores avanzados, los drones pueden identificar focos de calor en sus etapas iniciales, permitiendo una respuesta rápida antes de que el fuego se propague fuera de control. Su capacidad para acceder a zonas de difícil alcance garantiza una perspectiva aérea en tiempo real, facilitando la toma de decisiones y la descarga precisa de retardantes de fuego sobre áreas críticas. Recientemente un equipo de investigadores de la UPM ha ganado una competición internacional sobre el uso de drones frente a incendios <https://www.upm.es/?id=CON05856&prefmt=articulo&fmt=detail>.

En las prácticas de Programación vamos a construir una aplicación simplificada que simulará el centro de control de un aeródromo de drones: será capaz de plantear misiones de observación con sus drones, hará el seguimiento de los drones en tiempo real, identificará los drones que están en peligro por pasar cerca unos de otros, o las misiones activas.

Iremos añadiendo funciones de manera incremental en las prácticas. En la práctica 1, deberá crear la clase que representa el concepto de `Dron` y otra clase para representar una `Posicion` en el espacio 3D respecto a un origen de coordenadas.

Un `Dron` tiene un identificador o nombre, una `Posicion` con su ubicación, el valor del tiempo de la última vez que se ha recibido información, y otra `Posicion` que usaremos como si fuera un vector 3D para almacenar su velocidad. Para manejar valores de tiempo usaremos un entero `long` que permite almacenar valores muy grandes, y que normalmente cuenta el número de milisegundos desde el 1 de enero de 1970. Prácticamente todos los ordenadores tienen una parte de su sistema operativo que actualiza el reloj del sistema de esta forma.

El resto del documento contiene las indicaciones para crear el proyecto, programar las clases, probar que funcionan y entregar la práctica. Léalas detenidamente y sígalas en orden.

Creación del proyecto

1. Descargue de Moodle el fichero zip que encontrará en la entrada `Práctica 1: proyecto para empezar`. En Eclipse debe hacer `File` → `Import` → `General` → `Existing Projects into Workspace`, e indicar la localización del fichero zip usando la opción `Select archive file`. El fichero se llama `PROG24practica1` y contiene varios archivos; dentro de la carpeta `src` se encuentra el paquete `es.upm.dit.prog.practica1` que es donde deberá poner las nuevas clases.

Programación

1. Cree la clase `Posicion` que tiene los siguientes atributos en este orden:

```
private double x;
private double y;
private double z;
```

2. Cree el constructor de la clase `Posicion` con un parámetro para sus atributos en el orden que se ha indicado (el orden de los parámetros es importante).

3. Cree los métodos accesores (`double getX()`, `double getY()`, `double getZ()`) y los métodos modificadores (`setX(double x)`, `setY(double y)`, `setZ(double z)`).

4. A continuación, incluya los métodos `equals` y `toString`, que Eclipse es capaz de crear automáticamente a partir de los atributos (`Menú Source` → `Generate toString()`, y `Menú Source` → `Generate hashCode() and equals()`). Una `Posicion` es igual a otra si lo son los valores de sus atributos.

5. Escriba el método `double modulo()`, se calcula como la raíz cuadrada de la suma de los cuadrados de sus coordenadas: `this.x`, `this.y`, `this.z`.

6. Por último en `Posicion`, escriba el método `double distancia(Posicion pos)`, que toma la `Posicion pos` como parámetro y devuelve la distancia entre esta y la `Posicion this`. Se puede conseguir fácilmente construyendo en el método una `Posicion` auxiliar cuyas coordenadas `x,y,z` sean la resta de las coordenadas de `pos` y de las de `this`, y calculando el módulo de esta nueva `Posicion`. Al acabar de ejecutar el método `distancia` y devolver el resultado, la `Posicion` auxiliar desaparecerá de la memoria.

7. Cree la clase `Dron` con estos atributos y en este orden:

```
private String id;
private Posicion pos;
private long t;
private Posicion vel;
```

Donde `id` es el identificador de un `Dron`; `pos` es la `Posicion` en la que ha sido visto por última vez, y `t` el valor de tiempo en el que se ha visto por última vez (almacenado como un número entero largo `long` que cuenta el número de milisegundos transcurridos desde el 1 de enero de 1970). `Posicion vel` representa la velocidad del dron en el tiempo `t`.

Para manejar la velocidad del `Dron` vamos a considerar que una `Posicion` es a la vez un punto en el espacio 3D y un vector que une la coordenada `0,0,0` con ese punto del espacio. Así, para calcular la nueva `pos1` que ocupará si estaba en la `pos0` y pasan `t` unidades de tiempo, se tiene la siguiente fórmula vectorial donde la última parte es el producto del escalar tiempo por el vector velocidad:

$$\vec{pos}_1 = \vec{pos}_0 + (\vec{vel} \cdot t)$$

8. Cree un método constructor con los parámetros necesarios para dar valor a todos los atributos; debe crear también los métodos modificadores y accesores para cada atributo.

9. A continuación, incluya los métodos `equals` y `toString`, que Eclipse es capaz de crear automáticamente a partir de los atributos (`Menú Source → Generate toString(), y Menú Source → Generate hashCode() and equals()`). Al generar `equals` , es necesario utilizar sólo el valor de `id` para la comparación. Para ello, indíquelo en el menú, marcando sólo este atributo. Este método vale para comprobar si dos drones son en realidad el mismo.

10. Escriba el método `void mover(long t)` , que será el encargado de calcular la posición del dron cuando se alcance el tiempo `t`. Si el parámetro `t` es menor que el valor del atributo `this.t` , entonces no hay que hacer nada (se ha llamado al método mover con un valor de tiempo del pasado). En el resto de casos hay que calcular el nuevo valor de la posición del dron que se calculará como la velocidad multiplicada por el tiempo que ha pasado entre `this.t` y `t`, a lo que hay que sumar la posición inicial. La fórmula es la indicada más arriba y hay que hacerla para cada coordenada; por ejemplo para la coordenada x corresponderá a esta fórmula:
pos_x = pos_x + (vel_x * (t - this.t))

Escriba el código de cada coordenada, cree una `Posicion` y póngala como nueva `pos` del dron. Ponga como nuevo valor del tiempo el del parámetro `t`.

11. Defina la constante:

```
private static final long SAFETY_DISTANCE = 2;
```

Escriba el método `boolean peligro(Dron otro)` que indica si `this` y `otro` están en peligro. Lo están cuando otro no es `null`, es diferente (`! equals`) de `this` , y la distancia entre las posiciones actuales de ambos es menor que `SAFETY_DISTANCE.`

Pruebas

1. Cuando haya terminado de programar las dos clases, puede pasar a probar su funcionamiento.

2. Si no tiene errores de compilación, puede ejecutar las pruebas empaquetadas que hay en `PruebaPractica1` (marque este fichero y ejecute el programa: `Run → Run Java application`.). Se crean objetos de las clases que estamos probando, se llama a sus métodos, y se comprueba que los efectos y resultados son los

correctos. Si hay errores en la ejecución del programa, aparecerán mensajes en la pantalla; deberá leerlos para intentar arreglar el código incorrecto en sus clases. Al final, este programa proporciona una nota aproximada.

3. Otra forma de probar que funcionan bien es mediante el programa `PruebaInteractiva1`; para ejecutarlo marque el fichero y haga: `Run` → `Run Java application`. En la consola de Eclipse, aparecerá un mensaje de saludo, y un cursor para que pueda escribir alguna de estas órdenes; el resultado se mostrará en la siguiente línea, con el cursor preparado para la siguiente orden:

ORDEN	FUNCIONAMIENTO
`hello`	Saluda
`status`	Muestra los valores de las variables
`help`	Muestra la lista de órdenes
`exit`	Acaba el programa
`clear`	Inicia las variables
`pos x y z`	Crea `this.pos` que es una `Posicion` con `x:double y:double z:double`
`vel x y z`	Crea `this.vel` que es una velocidad con `x:double y:double z:double`
`dron id t`	Crea `this.dron` nuevo con `id:String this.pos t:long this.vel` y guarda el anterior en this.dron2
`addvel`	Añade `vel` a `this.dron1`
`mover1 t`	Mueve el `this.dron1` con `t:long`
`mover2 t`	Mueve el `this.dron2` con `t:long`
`peligro`	Comprueba si el `this.dron1` y `this.dron2` están en peligro

Puede copiar las órdenes del fichero `comandos1.txt` y comprobar los resultados.

Entrega y evaluación

No entregue la práctica hasta haber superado con éxito todas estas pruebas. Aunque el programa `PruebaPractica1` proporciona una nota aproximada, si aparece el siguiente mensaje, su código no es correcto por lo que cuando se ejecute en el servidor para la corrección final aparecerán errores en la compilación y la nota será `0.0`.

```
Error en las cabeceras de métodos de Posicion o Dron.
```

No se considerará entregada la práctica hasta que no la suba a moodle -dentro del plazo indicado-; para hacer la entrega de la práctica a través de moodle debe:

1. Con el ratón sobre el proyecto `PROG24practica1`, en Eclipse seleccione el menú: `File` → `Export...` → `General` → `Archive File` (también puede aparecer en español como `Fichero zip`)
2. Seleccione la carpeta `src` de su proyecto, y allí los ficheros Java; en "`To Archive file`" indique la carpeta de su sistema de ficheros donde quiere

colocar el fichero zip y el nombre de este fichero que debe ser `PROG24practica1.zip`. Asegúrese de que se guarda en formato zip con las carpetas necesarias.

3. Fuera de Eclipse, localice el fichero `PROG24practica1.zip` y súbalo a la tarea `Práctica 1` del Moodle de Programación.

Cuando termine el plazo de entrega de la práctica, se recogerán y comprobarán todas las entregas. Se asignará la nota 0.0 a las entregas que no se hayan recibido a tiempo, que no contengan los ficheros indicados, o que no compilen correctamente. Para cada una, se pasarán pruebas parecidas a las de la nota provisional. Las notas definitivas aparecerán en la entrada de moodle un tiempo después del cierre de la entrega, cuando los profesores corrijan todas las prácticas y revisen que no hay copias de código.

Se recuerda que la copia de código supone el suspenso automático de la práctica y de la asignatura, tanto para quien copia el código como para quien lo cede sin distinción. Además, se emprenderán medidas disciplinarias contra dichos alumnos ante la Escuela y la Universidad.