PROGRAMACION - PRACTICA 2

Objetivos

- Realizar cálculos aritméticos
- Crear objetos y manejarlos mediante referencias
- Almacenar referencias a objetos en un array

Recursos

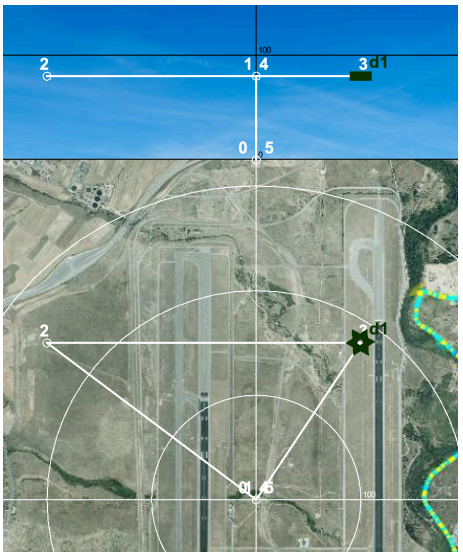
Proyecto eclipse que contiene, entre otros ficheros y carpetas:

- Paquete `es.upm.dit.prog.practica2`
- `PruebaPractica2.java`: programa que prueba si funcionan bien las clases
- `PruebaInteractiva2.java`: programa que recibe órdenes por teclado para probar el funcionamiento.

Descripción

Los drones se pueden usar para obtener imágenes de un incendio. Para ello, se define una Misión de reconocimiento con un determinado Dron; la misión consta de un conjunto de posiciones que debe alcanzar el Dron junto con la indicación del tiempo de llegada a cada una. La Misión contiene un método para comprobar si el dron ha alcanzado la posición correspondiente en el tiempo esperado y si es así, darle la orden necesaria para ir a la siguiente posición.

La imagen de la derecha representa la misión `m1` que deberá seguir el dron `d1` con éste en una de sus posiciones. El panel del cielo corresponde al frontal (visualización XZ) y el del mapa corresponde al suelo (visualización XY). Los atributos de esta misión son los siguientes:




```
Mision [id=m3,
  d=Dron [id=d1,
    pos=Posición [x=100.0, y=150.0, z=80.0],
    t=200,
    vel=Posición [x=-1.0, y=-1.5, z=0.0]],
  posiciones=[
    Posición [x=0.0, y=0.0, z=0.0],
    Posición [x=0.0, y=0.0, z=80.0],
    Posición [x=-200.0, y=150.0, z=80.0],
    Posición [x=100.0, y=150.0, z=80.0],
    Posición [x=0.0, y=0.0, z=80.0],
    Posición [x=0.0, y=0.0, z=0.0]],
  tiempos=[0 10 110 200 300 310],
  nPosiciones=6,
  posicion=3]
```

Usaremos un array de objetos de la clase Posición para el conjunto de posiciones, y un array de enteros `long` para los tiempos. Inicialmente la `Mision` tiene los arrays vacíos, así que definiremos un método para añadir una nueva posición con su tiempo a estos arrays; el atributo `nPosiciones` controla cuántas posiciones se han añadido, y `posicion` indica en cual de ellas se encuentra el dron. Cuando el dron ha alcanzado el tiempo previsto, se calcula la velocidad que debe tomar para alcanzar la siguiente posición en el tiempo esperado, y se da la orden al `Dron` de moverse con esta velocidad. Usaremos las clases `Posición` y `Dron` de la práctica anterior sin cambios.

Creación del proyecto

1. Descargue de moodle el fichero zip que encontrará en la entrada `Práctica 2: 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 `PROG24practica2` y contiene varios archivos; dentro de la carpeta `src` se encuentra el paquete `es.upm.dit.prog.practica2` que es donde deberá poner las nuevas clases.
2. Copie las clases `Posicion` y `Dron` de la práctica 1 en el paquete `es.upm.dit.prog.practica2`.

Programación

1. Cree la clase `Mision` con estos atributos y en este orden:

```
private String id;
private Dron dron;
private Posicion[] posiciones;
private long[] tiempos;
private int nPosiciones;
private int posicion;
```

Ponga un constructor `public Mision(String id, Dron dron, int maxPosiciones)` que inicializa los atributos con el constructor, el array `posiciones` tendrá `maxPosiciones` casillas, igual que el array `tiempos`, y tanto `nPosiciones` como `posicion` deben iniciarse a 0.

2. Cree los métodos accesores, `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 `Mision` es igual a otra si lo son los valores de su atributo `id`. No es necesario que ponga métodos modificadores porque todas las operaciones se van a hacer con los métodos que haremos a continuación. Deje el accesor del atributo `nPosiciones` que genera Eclipse, con nombre `getnPosiciones`.

3. Escriba el método `public void addPosT(Posicion pos, long t)` que añadirá una posición y valor de tiempo a los arrays `posiciones` y `ts`, sólo si `pos` es distinto de `null`. El atributo `this.nPosiciones` controla cuántas se han añadido hasta el momento, así que si este número es mayor o igual al tamaño del array `this.posiciones` no lo podremos añadir porque no hay sitio. En caso contrario sí

que vamos a añadir `pos` al array en la casilla `this.nPosiciones`, igual que el valor `t` en el array `this.tiempos`. Finalmente, sumaremos 1 a `this.nPosiciones`.

4. Programe el método `public boolean activa()` que indica si la `Mision` está activa; lo está cuando estamos en una posición válida porque está en el array y su casilla no es `null`. Para ello, compruebe que `this.posicion` es menor que `this.nPosiciones` y la casilla `this.posicion` del array `this.posiciones` es distinta de `null`.

5. Por último añada el método `public void update(long t)`. El objetivo de este método es actualizar el estado de la misión, de forma que si se ha alcanzado una posición se pasa a la siguiente. Esto sólo se puede hacer si coinciden los valores de tiempo del dron, el de esta fase de la misión y el parámetro. Si se cumplen esas condiciones y la misión está activa, pasaremos a la siguiente fase calculando y poniendo en el dron su nueva velocidad; en caso de que fuera la última posición, esta velocidad será cero. Por último, incrementaremos en 1 el atributo `this.posicion` para indicar que pasamos a la siguiente. El siguiente párrafo explica cómo programarlo.

Para comprobar que no estamos en la última posición tenemos que asegurarnos de que `this.posicion` es menor que `this.nPosiciones -1`. Y para calcular la nueva velocidad del dron antes de hacer `this.dron.setVel(vel)`, tenemos que saber la diferencia de tiempos entre la siguiente posición y la actual, accediendo a las casillas correspondientes de `this.tiempos`. Si la diferencia de tiempos es menor o igual a 0 pondremos velocidad 0; en caso contrario calculamos la velocidad como una nueva `Posicion` (recuerde que una posición puede comportarse como un vector para almacenar la velocidad, por ejemplo), tomando en cada coordenada la diferencia entre la posición siguiente a la actual y la posición del dron en este momento, dividida por la diferencia de tiempos.

Pruebas

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

2. Si no tiene errores de compilación, puede ejecutar las pruebas empaquetadas que hay en `PruebaPractica2` (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. No entregue la práctica hasta haber superado con éxito todas estas pruebas. Deberá leer los mensajes para comprobar que todo funciona como se espera. Este programa proporciona una nota provisional aproximada.

3. Otra forma de probar que funcionan bien es mediante el programa `PruebaInteractiva2`; 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
`mision id n`	crea una misión con id:String, this.dron1 y n posiciones
`addpos t`	añade this.pos y y t:long a this.mision
`update t`	actualiza this.mision con t:long
`show`	muestra un diagrama con la situación actual de this.mision y this.dron1

Puede copiar las órdenes del fichero `comandos2.txt` y comprobar los resultados. El gráfico que aparece al principio ha sido generado con la orden `show`.

Entrega y evaluación

No entregue la práctica hasta haber superado con éxito todas estas pruebas. Aunque el programa `PruebaPractica2` 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, Dron o Mision.
```

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 `PROG24practica2`, 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

`PROG24practica2.zip`. Asegúrese de que se guarda en formato zip con las carpetas necesarias.
3. Fuera de Eclipse, localice el fichero `PROG24practica2.zip` y súbalo a la tarea Práctica 2 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.