PROGRAMACION - PRACTICA 3

Objetivos

- definición de métodos
- operaciones con arrays de referencias
- sentencias de control

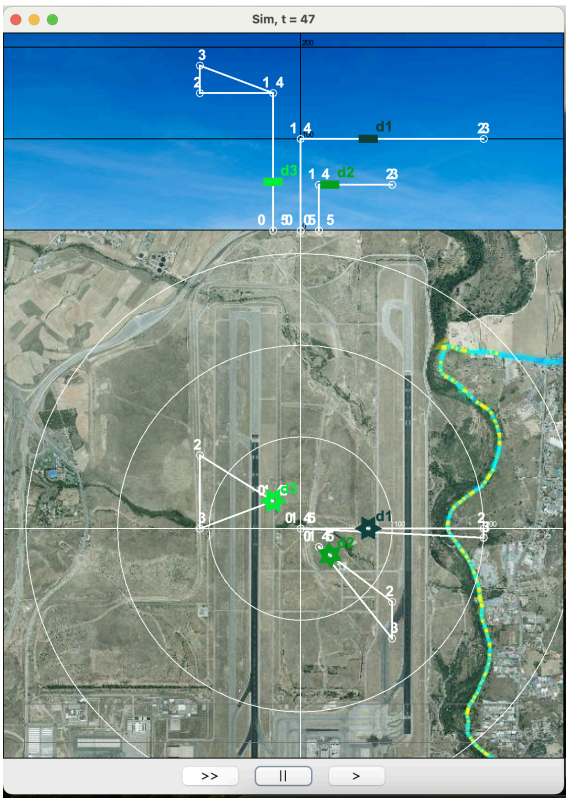
Recursos

Proyecto eclipse que contiene, entre otros ficheros y carpetas:

- Paquete `es.upm.dit.prog.practica3`
- `Prueba3.java` programa que prueba si funcionan bien las clases
- `PruebaInteractiva3.java` programa que recibe órdenes por teclado para probar el funcionamiento.

Descripción

Usaremos las clases `Dron`, `Vector` y `Mision` de las prácticas anteriores. Ahora vamos a desarrollar la clase `CentroControl`, que es la encargada de hacer el seguimiento de los `Drones` y de las `Misiones`. Usaremos arrays para guardar estos elementos. Como siempre que se usan los arrays, debemos tener cuidado con las celdas que contienen `null`. La imagen muestra un ejemplo consistente en 3 misiones cada una con su dron; se ha obtenido usando `PruebaInteractiva3`.



Creación del proyecto

1. Descargue de moodle el fichero zip que encontrará en la entrada `Práctica 3: 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 `PROG24practica3` y contiene varios archivos; dentro de la carpeta `src` se encuentra el paquete `es.upm.dit.prog.practica3` que es donde deberán estar las nuevas clases que desarrolle.
2. Copie las clases `Vector`, `Dron` y `Mision` de la práctica 2 en el paquete `es.upm.dit.prog.practica3`.

Programación

1. Cree la clase `CentroControl`, cuyos atributos son:
   - una constante entera `N` con valor `10`, para el número de celdas
   - un array `Dron[] drones`
   - un array `Mision[] misiones`

Escriba el constructor `public CentroControl()` que debe inicializar `this.drones` con `N` casillas vacías y `this.misiones` con `N` casillas vacías. No añada los modificadores ni accesores que genera Eclipse. Tampoco hace falta el método `equals`. Escriba un método `public String toString()`. Puede usar el que genera eclipse automáticamente.

2. Cree el método: `public void addDron(Dron d)`

Este método almacena el `Dron` pasado como parámetro, en `this.drones`, pero si `d` es `null` no hace nada. Supondremos que los `drones` llegan en orden adecuado, así que podemos recorrer el array, poner `d` en la primera posición nula que encontremos y acabar la ejecución del método.

Si por el contrario no ha encontrado hueco, la cosa se complica: quiere decir que el array está lleno, y que el elemento en la primera posición del array es el más antiguo y por tanto habrá que descartarlo. Podemos copiar, desde el segundo al último de los elementos, en la casilla anterior (el segundo al primero, el tercero al segundo… el último al penúltimo). Cuidado porque no se puede indexar el array `this.drones` con un índice negativo, ni con un valor del índice mayor o igual a `this.drones.length`. Después de esto, ya puede guardar el `dron d` en la última posición de `this.drones.`

3. Cree el método: `public Dron[] getDrones()`

Este método devuelve un nuevo array de `drones` que contiene la referencia de todos los drones de `this.drones`; el nuevo array no puede contener referencias `null`, y los `drones` deben ir en el mismo orden que tienen en `this.drones`; por lo tanto el nuevo array debe tener una longitud que coincida con el número de celdas de `this.drones` distintas de `null`. Se recomienda que en este método haga dos recorridos sobre el array `this.drones`: el primero para contar el número de casillas válidas, tras lo cual se creará un nuevo array de `drones` con ese número de casillas; y el segundo recorrido para copiar las casillas válidas de `this.drones` al nuevo

array. Si el atributo no contiene ningún `dron`, tendremos un array de longitud `0`. Sea como sea, debe devolver el array como resultado del método.

4. Cree el método: `public void addMision(Mision m)`
Haga lo mismo que en el método `addDron` pero con `this.misiones.`

5. Cree el método: `public Mision[] getMisiones()`
Haga lo mismo que en el método `getDrones` pero con `this.misiones.`

6. Cree el método: `public void update(long t)`
Este método recorre el array `this.drones`, y para cada uno que no sea nulo, llama al método `mover` pasando `t` como parámetro. Después recorre `this.misiones` y para cada una diferente de `null` llama al método `update` con `t` como parámetro.

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 `PruebaPractica3` (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 `PruebaInteractiva3`; 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 un vector posicion con x:double y:double z:double
`vel x y z`	crea this.vel que es un vector velocidad con x:double y:double z:double
`dron id t control`	crea this.dron1 nuevo con id:String this.pos t:long this.vel control:boolean, 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 this.dron1 está en peligro por this.dron2
`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 los drones de la estacion filtrados con el último selector
`adddron`	añade this.dron1 a this.cc
`getdrones`	obtiene drones no nulos de this.cc
`addmision`	añade this.mision1 a this.cc
`getmisiones`	obtiene misiones no nulas de this.cc
`updatecc t`	actualiza todos los elementos de this.cc con t:long
`sim tini tfin`	muestra una simulación del movimiento de los drones de la estacion con: tini:long tfin:long use los botones para parar, avanzar t+1 o continuar la simulación: || > >>

Puede copiar las órdenes del fichero `comandos3.txt` y comprobar los resultados.

Entrega y evaluación

No entregue la práctica hasta haber superado con éxito todas estas pruebas. Aunque el programa `PruebaPractica3` 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 CentroControl, 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 `PROG24practica3`, 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 `PROG24practica3.zip`. Asegúrese de que se guarda en formato zip con las carpetas necesarias.
3. Fuera de Eclipse, localice el fichero `PROG24practica3.zip` y súbalo a la tarea Práctica 3 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.