

# Práctica 9: Tráfico aéreo

**Objetivos:** Practicar con los recorridos y búsquedas en arrays de objetos

**Descripción:** Se desea realizar una parte del software perteneciente a un sistema de control de tráfico aéreo

**Software suministrado:** Se dispone de las clases `Pantalla`, `Radar` y `Posicion`, ya realizadas

| Pantalla                                                                                                                                   |
|--------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                            |
| + Pantalla()<br>+ void pintaPuntoTry(Posicion p)<br>+ void pintaAvion<br>(int idAvion, Posicion p)<br>+ void muestraAlarma(String mensaje) |

| Radar                                                |
|------------------------------------------------------|
|                                                      |
| + static Posicion leePosicionActual<br>(int idAvion) |

# Software suministrado

La clase **Radar** es la implementación informática de un sistema de radar

- se ofrece una simulación

La clase **Pantalla** permite crear ventanas donde representar información sobre trayectorias de aviones

- esta clase también utiliza las clases **Dibujo2** y **ColorFig2**
- y necesita el fichero **avion.png** con la imagen de un avión

La clase **Posicion** almacena la latitud y longitud que definen las coordenadas en la tierra (en grados)

| Posicion                                                                                                                            |
|-------------------------------------------------------------------------------------------------------------------------------------|
| - double lat<br>- double lng                                                                                                        |
| + Posicion (double lat, double lng)<br>+ double lat()<br>+ double lng()<br>+ String toString()<br>+ double distancia (Posicion pos) |

# Clase Trayectoria

Lo que se pide es construir la clase **Trayectoria** que almacene la trayectoria de un avión, y la represente en un objeto de la clase **Pantalla**

Atributos:

- **lista**: almacena la trayectoria que ha ido siguiendo el avión, es decir, las posiciones de los puntos por los que ha pasado, obtenidas del radar
- **idAvion**: el identificador del avión
- **ventana**: objeto donde se pinta la trayectoria

| Trayectoria                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| - Posicion[] lista<br>- int idAvion<br>- Pantalla ventana                                                                                                                                                                         |
| + Trayectoria(int idAvion)<br>+ void muestraAvion()<br>+ void recogePosiciones(int n)<br>+ boolean pasaCerca<br>(Posicion p, double dist)<br>+ Posicion[] estanDentroRect<br>(Posicion esquinaSupDcha,<br>Posicion esquinaInfIzq) |

parte avanzada

# Métodos de la clase Trayectoria

---

- *Constructor*: Almacena el identificador del avión `idAvion` que se pasa como parámetro. Crea un objeto de la clase `Pantalla` y lo guarda en `ventana`. Pone el atributo `lista` a valor `null`
- `muestraAvion()`: Esta operación muestra en la ventana todos los puntos de la trayectoria del avión (llamando sucesivamente a `pintaPuntoTry`), y finalmente llama a `pintaAvion` especificando el identificador del avión y la última posición almacenada en su trayectoria
- `pasaCerca()`: Retorna un booleano indicando si alguno de los puntos de la trayectoria pasa cerca del punto `p` o no. Se considera que dos puntos están cerca si están a una distancia menor al parámetro `dist` (expresado en Km)
  - Para este método utilizar alguno de los algoritmos de búsqueda vistos en clase de teoría

# Métodos de la clase Trayectoria

---

- `recogePosiciones()`: Crea el array `lista` de tamaño `n`. Luego va rellenando este array hasta que el número de datos añadidos sea `n`. Para ello seguir el siguiente pseudocódigo:

```
método recogePosiciones(entero n)
  crea el array lista de tamaño n
  entero añadidos=0
  mientras añadidos<n
    Posicion p=Radar.leePosicionActual(idAvion)
    si p es null entonces
      poner en ventana un mensaje de error
    si no
      lista[añadidos]=p
      añadidos++
    fin si
  fin mientras
fin método
```

# Programa principal

---

Hacer un `main` en una clase aparte que permita probar el funcionamiento de todos los métodos de la clase `Trayectoria`

- crea una trayectoria para el vuelo de `idAvion=8834`
- llama a `recogePosiciones()` con 200 posiciones
- llama a `muestraAvion()`
- Crea una posición de `lat=70°`, `long=70°`
- Muestra en pantalla si la trayectoria pasa cerca de esa posición a una distancia de 300 Km
- Lo mismo con una distancia de 200 Km

# Parte avanzada:

---

- `estanDentroRect()`: Escribir este método, que retorna las posiciones de la lista que están dentro de un rectángulo
  - El rectángulo es el definido por los parámetros esquina superior derecha y esquina superior izquierda
  - Para que un punto se considere dentro del rectángulo su latitud debe estar comprendida entre las latitudes inferior y superior del rectángulo, y su longitud debe estar comprendida entre las longitudes izquierda y derecha

Para este método habrá que hacer dos recorridos del array:

- Uno para contar cuántas posiciones están en el rectángulo. Con ese número crearemos el array a retornar
  - Un segundo recorrido para rellenar el array a retornar
- Asimismo, modificar el main para que use este método para un rectángulo en el que haya puntos dentro, y para otro en que todos están fuera. En ambos casos mostrar en pantalla los resultados.

# Entregar

---

- Código de la clase `Trayectoria`
- Código de la clase con el `main`
- Captura de pantalla con la trayectoria dibujada y los resultados de ejecutar el `main`
- Parte avanzada: nuevo método y capturas de pantalla de los resultados del `main` al utilizar el nuevo método