Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2012

Primera parte (5 puntos, 50% nota del examen)

1) Se dispone de una clase enumerada definida de este modo:

```
public enum Color{VERDE, ROJO, AZUL, AMARILLO, BLANCO, NEGRO}
```

Suponer que `c` y `valor` son variables definidas del siguiente modo:

```
Color c;
double valor;
```

Escribir en Java un fragmento de programa que use instrucciones "`if`" y haga lo siguiente:

```
si c es VERDE entonces
    si valor está comprendido entre 0 y 10
        escribir en pantalla "Verde - Correcto"
    si no
        escribir en pantalla "Verde - Error"
    fin si
si no, si c es ROJO entonces
    si valor está comprendido entre 10 y 15
        escribir en pantalla "Rojo - Correcto"
    si no
        escribir en pantalla "Rojo - Error"
    fin si
si no
    //c no es ni ROJO ni VERDE
    escribir en pantalla "Color Desconocido"
fin si
```

Además, escribir un fragmento de programa que haga lo mismo pero usando la instrucción "`switch`" para la comprobación de los colores.

2) Escribir un método Java con la siguiente cabecera que crea y retorna un array multidimensional de `m` filas y `n` columnas, con las casillas de la primera fila a valor `1.0`, y todas las demás casillas con el valor -1.0.

```
public double[][] creaMatriz(int m, int n)
```

3) Escribir el *pseudocódigo* de un método iterativo que invoque al método estático `calcula()` de la clase `Raices` y lo haga dentro de un bucle cuyas características serán:

- se creará y usará la variable `i` de tipo `double` cuyo valor inicial es 1.0, y que se incrementa en 0.1 unidades cada vez que se repite el bucle
- se creará una variable `suma` para guardar la suma de todos los valores obtenidos al llamar al método `calcula()`
- condiciones de permanencia del bucle: variable `i` menor o igual a 10.0, y valor retornado por `calcula()` en la iteración anterior mayor que 1.0e-9

En la llamada a `calcula()` se usará como parámetro la variable `i`. El método `calcula()` tiene la siguiente cabecera:

```
public static double calcula(double i)
```

El método retornará la variable `suma`. Usar la notación del pseudocódigo vista en clase.

4) Se dispone de un método `proximo()` cuya cabecera se muestra abajo y que siempre retorna el nombre de la próxima persona a la que hay que atender en un servicio. Escribir un *método recursivo* en Java llamado `muestraInverso()` que permita mostrar en pantalla `n` nombres retornados por `proximo()` pero en orden inverso. Las cabeceras de los métodos serán:

```
public String proximo()
public void muestraInverso(int n)
```

En el método `muestraInverso()` primero hay que llamar a `proximo()` y guardar el valor obtenido en una variable `p`. Luego hay que resolver el caso directo y el recursivo. El caso directo se da cuando `n` es 0 y el caso recursivo cuando `n>0`. En el caso directo no hay que hacer nada. En el caso recursivo hay que invocar al mismo método `muestraInverso()` para `n-1` y luego hay que mostrar en pantalla la variable `p`.

5) Contestar *razonadamente* a las siguientes preguntas. Utilizar un *máximo* de *3 líneas* para cada respuesta:

a. En una hoja de cálculo una celda tiene la fórmula indicada abajo. Indicar qué valor se mostrará si D5 vale 9.0 y D6 vale 9.5

```
=SI(D5<D6; 10,0; 12,0)
```

b. ¿Qué diferencias hay entre las referencias a celdas relativas y absolutas en una hoja de cálculo? Pon ejemplos de cómo se expresa cada una.

c. Cuando cambiamos los datos de una hoja de cálculo, ¿cómo conseguimos que una gráfica se actualice para tener en cuenta los nuevos datos?

d. ¿Es cierta esta afirmación? En un registro de una base de datos siempre debe haber una clave extranjera.

e. ¿Qué es el SQL?

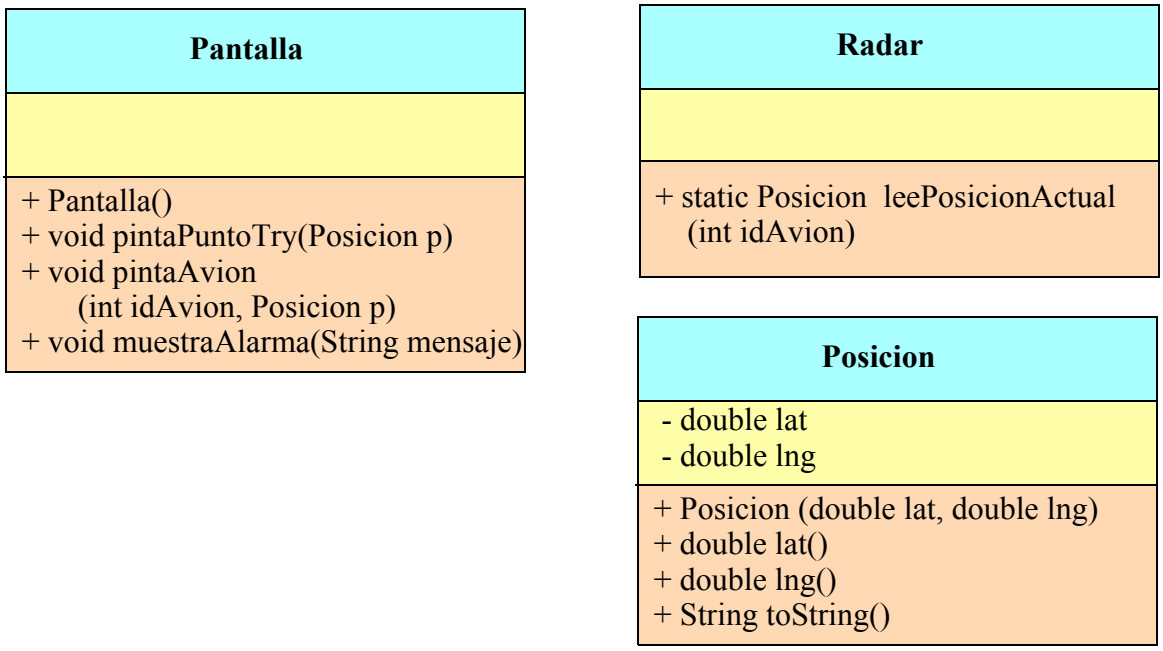
Nota: en esta cuestión, las respuestas correctas suman 0.2 puntos y las incorrectas restan 0.1 puntos. Se valora la *precisión* de la respuesta.

Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2012

Segunda parte (5 puntos, 50% nota del examen)

Se desea realizar una parte del software perteneciente a un sistema de control de tráfico aéreo. Para ello, se dispone de las siguientes clases, ya realizadas, que funcionan de acuerdo a los siguientes diagramas de clases:



La clase `Posicion` almacena la latitud y longitud que definen las coordenadas en la superficie de la tierra. Sus métodos son:

- *Constructor*, al que se le pasan la latitud y longitud, en grados
- `lat()`: observador de la latitud; la devuelve en grados. La latitud aumenta de abajo a arriba en el mapa.
- `lng()`: observador de la longitud; la devuelve en grados. La longitud aumenta de izquierda a derecha en el mapa.
- `toString()`: retorna las coordenadas convertidas a texto

La clase `Radar` es la implementación informática de un sistema de radar. Tiene un método:

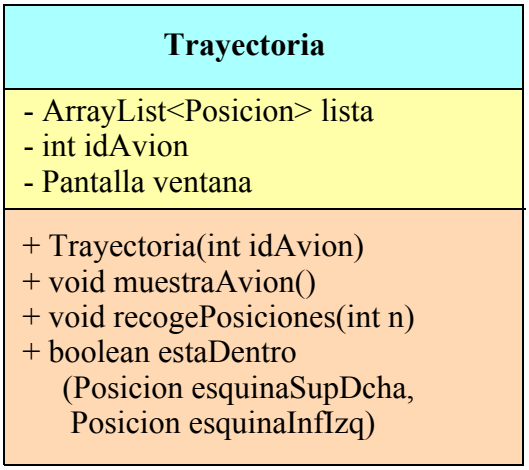
- `leePosicionActual()`: retorna la posición actual del avión identificado por `idAvion`. Si el radar no logra encontrar el avión, retorna `null`

La clase `Pantalla` permite crear ventanas donde representar información sobre trayectorias de aviones. Dispone de los siguientes métodos:

- *Constructor*. Crea una ventana para mostrar información de trayectorias.
- `pintaPuntoTry()`. Pinta en la ventana un punto de la trayectoria que ha seguido un avión, cuya posición es `p`. Llamando muchas veces a este procedimiento con diferentes posiciones, se puede conseguir que aparezca en pantalla una trayectoria completa.

- `pintaAvion()`. Muestra en la ventana la posición actual del avión (especificada por `p`), junto a su identificador (especificado por `idAvion`).
- `muestraAlarma()`: Muestra en pantalla el mensaje de alarma especificado por `mensaje`.

Lo que se pide es construir la clase `Trayectoria` que almacene la trayectoria de un avión, y la represente en la pantalla. La clase tendrá el siguiente diagrama de clases:



La clase `Trayectorias` debe almacenar en `lista` la trayectoria que ha ido siguiendo el avión, es decir, las posiciones de los puntos por los que ha pasado, obtenidas del radar. Asimismo debe almacenar el identificador del avión en `idAvion`. Sus métodos son:

- *Constructor*: Almacena el identificador del avión `idAvion` que se pasa como parámetro. Crea un objeto de la clase `Pantalla` y lo guarda en `ventana`. Crea la `lista` vacía.
- `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.
- `recogePosiciones()`: Llama `n` veces al método `leePosicionActual()` de la clase `Radar`. Añade cada dato correcto a la `lista`, pero si el dato obtenido es `null` pone en la `ventana` un mensaje de error que incluye el identificador del avión.
- `estaDentro()`: Hace una búsqueda para comprobar si alguna de las posiciones almacenadas en `lista` está dentro de un rectángulo y retorna `true` en caso afirmativo o `false` en caso negativo. El rectángulo es el definido por los parámetros: esquina superior derecha y esquina inferior 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.

Valoración:

1) Constructor: 0.5 puntos

2) Resto de los métodos: 1.5 puntos cada uno