Examen de Introducción al Software
(Ingeniería Informática)

Septiembre 2014

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

1) Se desea escribir un método que calcule y retorne el precio que fijará una compañía aérea para un billete de avión. El método recibe como parámetros el precio máximo y el porcentaje de ocupación del avión. El precio se calcula aplicando al precio máximo el descuento indicado en la tabla, de acuerdo con el porcentaje de ocupación:

Ocupación	Descuento
>80%	0%
>70%	20%
>55%	40%
>40%	60%
<=40%	80%

Por ejemplo, si el precio máximo es 200.0 euros y la ocupación es el 85%, no hay descuento por lo que el precio final es 200.0 euros; si la ocupación es del 70% el descuento que corresponde es el 40% y por tanto el precio final es 120.0 euros.

2) Escribir un método al que se le pasa como parámetro un array de números reales y que cuenta cuántos números son positivos, cuántos negativos y cuántos cero. El método retorna un array de tres enteros conteniendo los resultados (cuenta de positivos, negativos y ceros).

3) Escribir un método al que se le pasa un string que contiene una frase y retorna la parte del string original comprendida entre dos caracteres '#', o el string "!!" si no se encuentra ningún texto delimitado por esos caracteres '#'. En caso de haber varios substrings delimitados por '#' se retornará el primero de ellos.

4) Escribir el *pseudocódigo* de un método *recursivo* que calcula y retorna el máximo común divisor (`mcd`) de dos números enteros `a` y `b` que se le pasan como parámetros. En primer lugar, el método intercambia `a` y `b` si `a<b`. Posteriormente se distingue el caso directo y el caso recursivo. El caso directo se da cuando `b` es cero y basta retornar `a`. En el caso recursivo se retorna el máximo común divisor de `a-b` y `b`.

   Nota: este algoritmo solo funciona si `a` y `b` son no negativos y no son cero simultáneamente, pero por sencillez no es preciso comprobar estas condiciones.

5) Utilizar un *máximo* de *3 líneas* para cada respuesta:

   a. En una hoja de cálculo, ¿Qué aspectos de una celda que contiene números pueden definirse en el formato de la celda? Indicar 4 de estos aspectos.

   b. La función MAX de una hoja de cálculo permite obtener el máximo de un conjunto de valores. Escribir una fórmula para calcular el máximo de una columna de

valores entre las celdas A10 y A51, ambas incluidas.

c. En una hoja de cálculo, ¿crees que es posible que una celda tenga un color diferente según sea positiva o negativa? Contestar *razonadamente*.

d. En una base de datos, indica *razonadamente* la diferencia entre una consulta y una tabla.

e. Resume cómo pueden asociarse datos de dos tablas en una base de datos. Por ejemplo de una tabla de profesores y otra de asignaturas.

Nota: en esta cuestión, las respuestas correctas suman 0.2 puntos, las incompletas o las que pasen de tres líneas ni suman ni restan y las erróneas restan 0.1 puntos. Se valora la *precisión* de la respuesta.

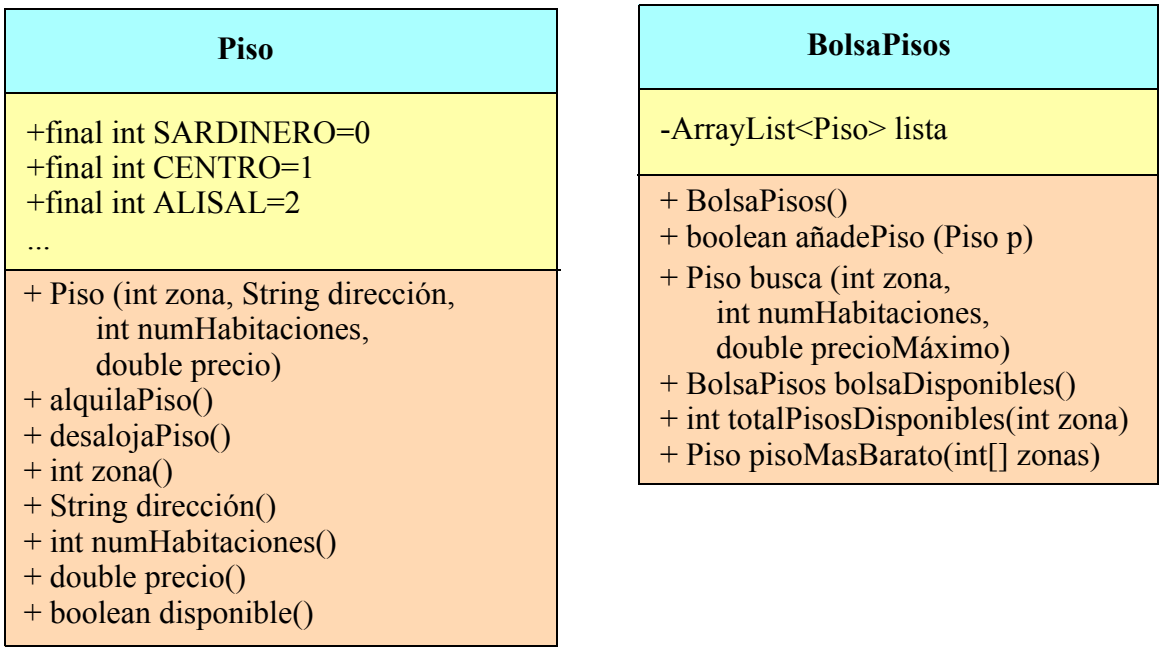
Examen de Introducción al Software
(Ingeniería Informática)

Septiembre 2014

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

Se desea realizar una parte del software para gestionar una bolsa de pisos en alquiler. Para ello se dispone de la clase Piso ya realizada, cuyos objetos almacenan la zona donde se encuentra el piso, su dirección, número de habitaciones, precio y si está disponible o no.

Se desea crear la clase BolsaPisos. Los diagramas de estas clases se muestran aquí:



Los métodos de la clase Piso (ya hecha) hacen lo siguiente:

- *Constructor*: construye el piso a partir de sus datos, que se pasan como parámetros. Lo deja disponible. La zona debe ser una de las constantes públicas de la clase (SARDINERO, CENTRO, ALISAL, ...).
- alquilaPiso(): pone disponible a false
- desalojaPiso(): pone disponible a true
- zona(), dirección(), numHabitaciones(), precio(), disponible(): son observadores de los respectivos atributos privados

La clase BolsaPisos (que se pide) dispondrá de un único atributo que es un ArrayList de objetos de la clase Piso y que contiene la lista de los pisos pertenecientes a la bolsa. Sus métodos deberán hacer lo siguiente:

- *Constructor*: Crea la lista vacía
- añadePiso(): añade el piso especificado por p a la lista si no existe ya un piso en la misma dirección. Retorna true si se pudo añadir, o false si no se pudo añadir por estar la dirección duplicada.

- busca(): Busca y retorna el primer piso que se encuentre en la zona indicada, con el número de habitaciones indicado, con un precio inferior o igual al precio máximo indicado, y que esté disponible. En caso de no encontrar ninguno de esas características, retorna null.
- bolsaDisponibles(): Retorna una bolsa de pisos igual a la actual, pero solo con aquellos pisos que estén disponibles. Para ello, crea una nueva bolsa de pisos, le añade todos los pisos de la lista que estén disponibles, y finalmente retorna la nueva bolsa.
- totalPisosDisponibles(): Retorna el número total de pisos que hay en la bolsa que estén en la zona indicada por el parámetro y que estén disponibles.
- pisoMasBarato(): Retorna el piso más barato que se encuentre en una de las zonas indicadas en el array zonas que se pasa como parámetro, o null si no hay ninguno en esas zonas. Para este método utilizar el siguiente pseudocódigo:

```
método pisoMasBarato(entero[0..n-1] zonas) retorna Piso
    // variable que recoge el piso más barato encontrado hasta el momento
    Piso masBarato=null

    // recorrido de toda la lista
    para cada Piso p en lista hacer
        // primero vemos si la zona del piso p está en el array zonas
        booleano pertenece=falso
        entero i=0 // indice para recorrer los elementos de zonas
        mientras i<n y no pertenece hacer
            si zona de p = zonas[i] entonces
                // la zona de p coincide con una de las que hay en zonas
                pertenece=verdad
            si no
                incrementa i
            fin si
        fin mientras

        // si la zona de p esta en zonas hay que mirar si este es el piso más barato
        si pertenece entonces
            si masBarato=null o si no precio de p < precio de masBarato entonces
                // p es ahora el más barato y lo anotamos en masBarato
                masBarato=p
            fin si
        fin si

        // acaba el recorrido de la lista
    fin para

    // retornamos el piso más barato encontrado, o null si no se encontró ninguno
    retorna masBarato
fin método
```

Valoración:

1) Encabezamiento de la clase, atributo y constructor: 0.5 puntos

2) añadePiso: 0.5 puntos

3) Resto de los métodos: 1 punto cada uno