Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2015

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

1) Escribir el *pseudocódigo* de un método al que se le pasa como parámetro un array de números reales, y que calcula y retorna la amplitud de sus valores, que es la diferencia entre el máximo y el mínimo valor.

2) Escribir un método Java al que se le pasa como parámetro un String `str` que contiene una instrucción en un lenguaje ensamblador. El método debe analizar la instrucción y retornar un valor del tipo enumerado `TipoInstruccion` definido de la forma siguiente:

```
public enum TipoInstruccion {COMENTARIO, SUMA, RESTA, OTRA}
```

El método retornará:

- `COMENTARIO` si la instrucción comienza por un asterisco: *
- `SUMA` si la instrucción contiene la palabra `ADD` entre dos espacios en blanco y no es un comentario
- `RESTA` si la instrucción contiene la palabra `SUB` entre dos espacios en blanco y no es un comentario
- `OTRA` si no se cumple ninguna de las tres condiciones anteriores

Nota: se puede usar el método `contains(String s)` de la clase `String`, que retorna un booleano indicando si el string contiene la secuencia de caracteres definida por `s`.

3) Escribir un método Java que reciba como parámetros los siguientes valores reales relativos a una boya:

- j: fase del movimiento oscilatorio, en grados
- t: tiempo en segundos
- A: amplitud del movimiento en metros
- m: masa de la boya en Kg
- h: altura de la boya en m
- S: superficie de la boya en m^2
- ρ_s : densidad de la boya en Kg/m^3
- ρ_f: densidad del agua en Kg/m^3

y que calcule y retorne el desplazamiento x obtenido mediante las siguientes expresiones, que corresponden al movimiento oscilatorio de una boya:

$$T = 2\pi\sqrt{\frac{m+\rho_s hS}{\rho_f Sg}}$$

$$\omega = 2\pi / T$$

$$x = A\sin(\omega t + j)$$

donde g es la gravedad.

4) Escribir un método iterativo que calcule y retorne el siguiente desarrollo en serie del arco seno:

$$arc\sin(x) \cong \frac{1}{1}x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{15}{336}x^7 + \frac{105}{3456}x^9 \qquad si \qquad -1 \le x \le 1$$

El método debe hacer un bucle en el que se repite 5 veces la suma de un término al resultado. Para hacer más eficiente el cálculo no debe usarse la función "elevar a". Observaremos que cada potencia se obtiene multiplicando la anterior por x^2. Los números usados en los numeradores y denominadores se guardarán en sendos arrays conteniendo respectivamente num={1,1,3,15,105} y den={1,6,40,336,3456}. Cada término es por tanto num[i]*(x^{2i+1})/den[i]

El método recibe como parámetro el valor de *x*. Si éste es menor que -1 o mayor que 1 se debe retornar `Double.NaN` para indicar el error.

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

   a. En una hoja de cálculo, ¿qué diferencia hay entre una fórmula con referencias absolutas y otra con referencias relativas? ¿Para qué se usan unas y otras?

   b. Indicar los pasos que darías para contar el número de aprobados (casillas de valor superior o igual a 5.0) de una asignatura; las notas están en la columna C entre las filas 3 a 54.

   c. Se parte de una hoja de cálculo con diversas columnas de datos ya metidos. Describe brevemente los pasos a realizar para añadir una columna nueva que contenga la media de los valores de cada fila.

   d. Indica al menos tres tipos de datos enteros que están disponibles en una base de datos de tipo `base`. ¿Por qué hay distintos tipos y no solo uno?

   e. Indica para qué es útil relacionar datos de diferentes tablas en una base de datos

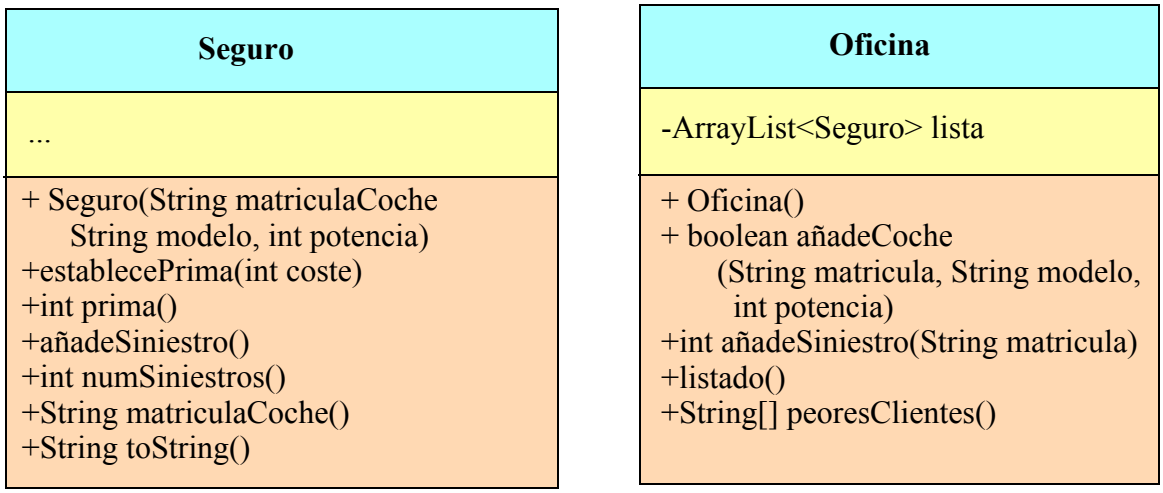
Nota: en esta cuestión, las respuestas correctas suman 0.2 puntos, las incompletas o las que pasen de 5 líneas ni suman ni restan y las erróneas restan 0.1 puntos. Se valora la *precisión* de la respuesta.

Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2015

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

Se desea realizar una parte del software perteneciente a una oficina de seguros de coches. Para ello se dispone de la clase Seguro ya realizada, cuyos objetos almacenan los datos de una póliza de seguro de un coche concreto. Se desea crear la clase Oficina. Los diagramas de estas clases se muestran aquí:



Los métodos de la clase Seguro hacen lo siguiente:

- *Constructor*: construye el seguro a partir de la matrícula, el modelo y la potencia del coche, que se pasan como parámetros
- establecePrima(): almacena el coste de la prima anual que se debe pagar por el seguro, que se pasa en el parámetro coste
- prima(): Retorna el coste de la prima anual
- añadeSiniestro(): Suma uno al número de siniestros declarados
- numSiniestros(): Retorna el número de siniestros declarados
- matriculaCoche(): Retorna la matrícula del coche
- toString(): Retorna un string que contiene la matrícula, modelo y potencia del coche, pero no la prima ni el número de siniestros

La clase Oficina dispondrá de un único atributo que es un ArrayList de objetos de la clase Seguro y que contiene la lista de los seguros pertenecientes a la oficina. Sus métodos deberán hacer lo siguiente:

- *Constructor*: Crea la lista vacía
- añadeCoche(): Busca si en la lista existe un seguro con la misma matrícula que matricula. Si se encuentra un coche de la misma matrícula se debe retornar false. En caso contrario se crea un nuevo seguro usando los datos del coche que se pasan como parámetros y se establece su prima según los datos de esta tabla:

potencia	prima
$0 \leq potencia \leq 50$	200
$50 < potencia \leq 75$	250
$75 < potencia \leq 120$	280
$potencia > 120$	320

Por último, se añade el nuevo seguro a la lista y se retorna true.

- añadeSiniestro(): Busca si en la lista existe un seguro para la matricula que se pasa como parámetro (utilizar para ello uno de los esquemas de búsqueda vistos en clase). Si se encuentra, aumenta su número de siniestros, aumenta su prima en un 10%, y retorna su número de siniestros ya actualizado. En caso contrario, retorna -1 para indicar el error.
- listado(): Hace un listado en pantalla de todos los seguros de la oficina, cada uno con su matrícula, modelo y potencia, su número de siniestros y su prima. Un seguro por línea.
- peoresClientes(): Retorna un array con las matrículas de los peores clientes, que son aquellos con un número de siniestros igual al máximo.

  Para el método peoresClientes() utilizar el siguiente pseudocódigo:

```
método peoresClientes() retorna array de String
   si tamaño de lista>0 entonces
      // calcula el máximo número de siniestros
      // y cuántas casillas hay con esos siniestros
      entero maxSiniestros=número de siniestros de la casilla 0 de lista
      entero numSeguros=1
      // recorremos todos los índices excepto el primero
      para i desde 1 hasta número de elementos de lista menos uno hacer
         entero num= número de siniestros de casilla i de lista
         si num > maxSiniestros entonces
            maxSiniestros=num
            numSeguros=1
         si no
            si num=maxSiniestros entonces
               numSeguros++
            fin si
         fin si
      fin para
      //Creamos el array a retornar
      peores= nuevo array de String de tamaño numSeguros
      // Rellenamos el array peores con los peores clientes
      entero n=0
      para cada Seguro seg en lista hacer
          si número de siniestros de seg = maxSiniestros entonces
              peores[n]=matricula de seg
              n++
          fin si
      fin para
      // retornamos los peores clientes
      retorna peores
```

```
        si no
            // la lista está vacía
            // retornamos array de cero elementos
            retorna nuevo array de 0 strings
        fin si
    fin método
```

Valoración:

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

2) añadeCoche: 1.5 puntos

3) Resto de los métodos: 1 punto cada uno