Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2016

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 el número de elementos comprendidos en el intervalo cuyo límite inferior y límite superior se pasan también como parámetros (incluyendo ambos límites en el intervalo).

2) Escribir un método Java al que se le pasa como parámetro un String `str` que contiene una reserva de hotel. El método debe analizar la reserva y en base a ello crear y retornar un objeto de la clase `Reserva` cuyo constructor tiene este encabezamiento.

```
public Reserva(String nombre, TipoServicio servicio)
```

`TipoServicio` es un enumerado definido de la forma siguiente:

```
public enum TipoServicio {SOLO_ALOJAMIENTO, ALOJAMIENTO_Y_DESAYUNO,
                          MEDIA_PENSIÓN, PENSIÓN_COMPLETA}
```

El String `str` tendrá en primer lugar un código de dos letras que indica el tipo de servicio, de acuerdo con la tabla siguiente, seguido de un espacio en blanco y del nombre del cliente. No es preciso tener en cuenta posibles errores en el formato de este String.

Código	Tipo de servicio
SA	SOLO_ALOJAMIENTO
AD	ALOJAMIENTO_Y_DESAYUNO
MP	MEDIA_PENSIÓN
PC	PENSIÓN_COMPLETA

Nota: se valorará el uso de la instrucción condicional más eficiente para este caso.

3) Escribir un método Java que reciba como parámetros los siguientes valores reales relativos a un cohete de *Torricelli* que se mueve por la reacción a la salida de un líquido de un depósito cilíndrico:
   - R: radio del depósito en m
   - r: radio del orificio de salida en m
   - ρ: densidad del líquido en kg/m^3
   - M: masa del depósito vacío en kg
   - m_0: masa inicial de líquido en kg
   - t: tiempo transcurrido en s

   El método calcula y retorna la velocidad del cohete, v, en m/s obtenida mediante las siguientes expresiones, siendo g la gravedad:

$$v = \frac{2}{R}\sqrt{\frac{2g'}{\pi\rho}}\left(kt + \sqrt{M}\cdot \operatorname{atan}\frac{\sqrt{m_0} - kt}{\sqrt{M}} - \sqrt{M}\cdot \operatorname{atan}\sqrt{\frac{m_0}{M}}\right)$$

$$k = \frac{r^2}{R}\sqrt{\frac{\pi\rho g'}{2}}$$

$$g' = \frac{g}{1 - r^4/R^4}$$

4) Algunos computadores muy básicos no disponen de operación "elevar a". Escribir un método recursivo Java que calcule y retorne el valor de *x* elevado a *n* siendo *x* un número real y *n* un número natural. Ambos números se pasan como parámetros. No se debe usar la función "elevar a" de Java. Se usará esta definición recursiva:

$$x^n = \begin{cases} x(x^2)^{(n-1)/2} & \text{si n es impar} \\ (x^2)^{n/2} & \text{si n es par} \end{cases}$$

El método tendrá dos casos directos: cuando *n* sea 0 hay que retornar 1 y cuando *n* sea 1 hay que retornar *x*. En el caso recursivo se distingue si *n* es par o impar y se utilizan las expresiones de arriba. Como es lógico, para calcular x^2 usar el producto $x \cdot x$.

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

   a. En una hoja de cálculo tenemos esta fórmula escrita en la celda `D1`: `=A1*B1`. ¿Qué fórmula aparecerá en la celda `E2` si copiamos allí esta fórmula? ¿Y si la fórmula original fuese `=$A1*B1`?

   b. Disponemos de una hoja de cálculo con valores entre las filas 3 a 53 en las columnas `B`, `C` y `D`. Indicar los pasos que darías para que en la columna `E` aparezca el número de casillas no vacías de la fila correspondiente.

   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 las casillas no vacías de la fila.

   d. Indica los motivos por los que al diseñar una base de datos es preciso elegir bien el tipo de datos a utilizar para guardar campos numéricos.

   e. Indica las diferencias entre una consulta y una tabla en una base de datos, expresando para qué se utiliza cada una.

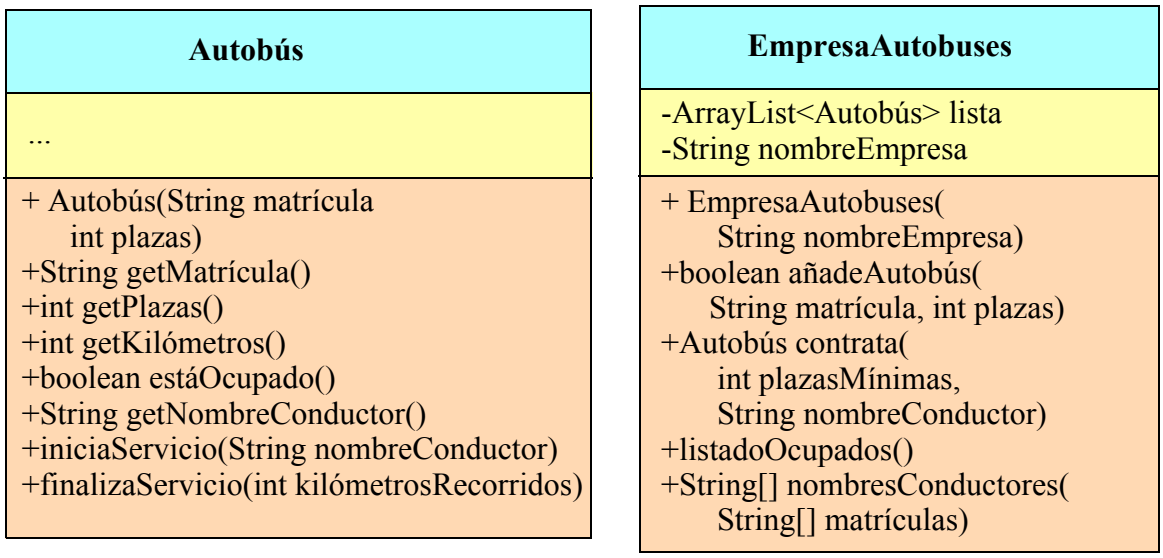
   Nota: en esta cuestión, las respuestas correctas suman 0.2 puntos, las incompletas o las que pasen de 5 líneas ni suman ni restan y las erróneas restan 0.1 puntos. Se valora la *precisión* de la respuesta.

Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2016

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

Se desea realizar una parte del software perteneciente a una empresa de autobuses. Para ello se dispone de la clase Autobús ya realizada, cuyos objetos almacenan los datos de un autobús concreto. Se desea crear la clase EmpresaAutobuses. Los diagramas de estas clases se muestran aquí:



Los métodos de la clase Autobús hacen lo siguiente:

- *Constructor*: Construye el autobús a partir de la matrícula y el número de plazas, que se pasan como parámetros.
- getMatrícula(): Retorna la matrícula del autobús.
- getPlazas(): Retorna el número de plazas del autobús.
- getKilómetros(): Retorna los kilómetros totales recorridos por el autobús.
- estáOcupado(): Retorna un booleano indicando si el autobús está ocupado o libre.
- nombreConductor(): Retorna el nombre del conductor o null si no hay ninguno asignado.
- iniciaServicio(): Inicia un servicio, asignando el nombre del conductor y dejando el autobús ocupado.
- finalizaServicio(): Finaliza un servicio sumando los kilómetros recorridos y dejando el autobús libre y el nombre del conductor a null.

La clase EmpresaAutobuses dispondrá de un atributo que es el nombre de la empresa y otro que es un ArrayList de objetos de la clase Autobús y que contiene la lista de los autobuses pertenecientes a la empresa. Sus métodos deberán hacer lo siguiente:

- *Constructor*: Asigna el nombre de la empresa que se pasa como parámetro al atributo nombreEmpresa y crea la lista vacía.

- añadeAutobús(): Añade un nuevo autobús a la lista con la matrícula y plazas indicadas, siempre que no exista un autobús en la lista con la misma matrícula; retorna true si se ha podido añadir o false en caso de que ya hubiese un autobús con la misma matrícula.
- listadoOcupados(): Hace un listado en pantalla de los autobuses ocupados. Se empieza con un encabezamiento que muestra el nombre de la empresa, y el número total de autobuses. Luego se muestra una línea que explica los datos que vienen a continuación. Por último se muestran los autobuses ocupados, uno por línea, con estos datos: matrícula, kilómetros y nombre del conductor.
- contrata(): Contrata un autobús con el número de plazas mínimo indicado. Para seleccionar el autobús de entre los que están en la lista se miran solo los que estén libres y con un número de plazas >= plazasMínimas. De estos autobuses se selecciona el que tiene el menor número de plazas y, entre los que cumplen este requisito, el que tiene menor número de kilómetros. Con el autobús seleccionado se inicia el servicio usando como nombre del conductor el que se pasa como parámetro. Retorna el autobús seleccionado o null si no hay ninguno libre con suficientes plazas. Para este método se seguirá el siguiente pseudocódigo:

```
método contrata(entero plazasMínimas, String nombreConductor)
                    retorna Autobús
   Autobús seleccionado=null;
   // Recorrer todos los autobuses
   para cada Autobús actual en lista hacer
      si actual está libre y tiene un número de plazas>=plazasMínimas entonces
         // el autobús actual está libre y con el número de plazas correcto
         si seleccionado=null entonces
            // Aún no hay ningún autobús seleccionado; elegimos el actual
            seleccionado=actual
         si no
            // si actual tiene menos plazas que el seleccionado, lo elegimos
            si número de plazas de actual<número de plazas de seleccionado
            entonces
               seleccionado=actual
            si no
               // si el autobús actual tiene las mismas plazas que el seleccionado
               // y su número de kilómetros es menor, lo elegimos
               si numero de plazas de actual = número de plazas de seleccionado y
                  kilómetros de actual<kilómetros de seleccionado
               entonces
                  seleccionado=actual
               fin si
            fin si
         fin si
      fin si
   fin para
   si seleccionado!=null entonces
      // se ha seleccionado autobús y se inicia el servicio
      seleccionado.iniciaServicio(nombreConductor)
   fin si
   retorna seleccionado
fin método
```

- nombresConductores(): Retorna un array con los nombres de los conductores de los autobuses cuyas matrículas se pasan en el parámetro. Los conductores aparecerán en el mismo orden que las respectivas matrículas. Es decir, si una matrícula está en la casilla

"i" del array, su conductor estará también en la casilla "i" del array retornado. Si alguno de los autobuses no tiene conductor por estar libre se pone en la casilla correspondiente del array retornado el String "autobús libre" en lugar del nombre del conductor. Si alguna de las matrículas no se encuentra en la lista se pone en la casilla correspondiente del array retornado el String "matrícula no encontrada:", concatenado a la matrícula no encontrada.

Para hacer este método se recomienda recorrer el array de matrículas según sus índices. Para cada índice "i" buscar la matrícula en la lista de autobuses. Si se encuentra, poner en la casilla "i" del resultado el nombre del conductor o "autobús libre" según el autobús esté ocupado o libre. Si no se encuentra, poner en la casilla "i" del resultado "matrícula no encontrada:" y la matrícula no encontrada.

Valoración:

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

2) nombresConductores: 1.5 puntos

3) Resto de los métodos: 1 punto cada uno