Examen de Introducción al Software
(Ingeniería Informática)

Septiembre 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 retorna otro array que solo contiene los elementos del array original no negativos. Observar que habrá que recorrer el array original dos veces. Una para contar el número de elementos no negativos y así poder crear el array a retornar de ese tamaño. En el segundo recorrido se rellena el array a retornar con los valores no negativos.

2) Escribir una clase Java con los métodos indicados en el pseudocódigo que aparece abajo, que permite jugar al conocido juego del *MasterMind* contra el computador. El usuario intentará averiguar el código de 4 cifras:

```
clase MasterMind
  // atributos:
  // código a adivinar
  texto codigo = ""
  // conjunto de símbolos válidos para el juego
  texto[] digitos = {"0","1","2","3","4","5","6","7","8","9"}

  // constructor que elige un código de 4 cifras no repetidas
  constructor MasterMind()
    texto candidato
    para i desde 1 hasta 4
       // repite hasta que la cifra encontrada no esté repetida
       hacer
          entero indice=número aleatorio entre 0 y 9
          candidato = digitos[indice]
       mientras codigo contiene a candidato
       codigo = codigo + candidato
    fin para
  fin constructor

 // comprueba cuántos aciertos y coincidencias hay en la propuesta
 método comprobar (texto propuesta) retorna texto
    entero aciertos = 0
    entero coincidencias = 0

    // recorremos la propuesta y verificamos sus caracteres
    para i desde 0 hasta 3
       si propuesta.caracterEnPos(i) = codigo.caracterEnPos(i) entonces
         aciertos++
       si no, si codigo contiene a propuesta.substring(i,i+1)  entonces
         coincidencias = coincidencias + 1
       fin si
    fin para
    retorna "Tu propuesta "+propuesta+" tiene"+aciertos+
         "aciertos y "+coincidencias+"coincidencias."
  fin método
fin clase
```

Nota 1: Para obtener un número aleatorio entre 0 y 9 se puede usar el método `Math.random()` que nos da un número real en [0.0 ... 1.0), multiplicarlo por 10 y convertirlo a entero truncando la parte fraccionaria

Nota 2: el método booleano `contains(s)` de la clase `String` permite saber si el string sobre el que se aplica contiene al string `s`. Por ejemplo, `s1.contains(s2)` retorna `true` si `s1` contiene a `s2` o `false` en caso contrario.

3) Escribir el código Java de un método *recursivo* que permite eliminar un texto almacenado en el parámetro `borrar` de un texto más grande llamado `completo`. El método retornará el texto obtenido. Su cabecera será:

```
public static String borra (String completo, String borrar)
```

Llamaremos `n` a la longitud del string `borrar`.

El caso directo se da cuando el texto `completo` tiene una longitud menor que `n`. En este caso se retorna el texto `completo`.

En el caso recursivo:

- Si los primeros `n` caracteres de `completo` son iguales a `borrar` se retorna el valor retornado por `borra()` usando como primer parámetro los caracteres de `completo` excepto los `n` primeros, y como segundo parámetro el texto `borrar`.
- Si no son iguales, se retorna el primer carácter de `completo` concatenado al valor retornado por `borra()` usando como primer parámetro el texto `completo` excepto su primer carácter, y como segundo parámetro el texto `borrar`.

4) 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`: `=A$1*B$1`. ¿Qué fórmula aparecerá en la celda `E2` si copiamos allí esta fórmula? ¿Y si la fórmula original fuese `=$A$1*$B1` ?

b. Disponemos de una hoja de cálculo con valores entre las filas 1 a 50 en las columnas `A` y `B`. Indicar los pasos que darías para que en la fila 51 aparezca el número de casillas no vacías de la columna 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 cuyas celdas contengan textos que pongan "APROBADO" si la media de las casillas no vacías de la fila es superior o igual a 5, o "SUSPENSO" en caso contrario.

d. Indica los motivos por los que al diseñar una base de datos es preciso pensar en los criterios de búsqueda que se van a usar.

e. Indica los motivos por los que es conveniente establecer relaciones entre tablas de 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)

Septiembre 2016

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

Un reciente juego para teléfonos móviles ha tenido un rápido crecimiento a nivel mundial y se hace necesario almacenar la información del uso y crecimiento de este juego para poder dimensionar adecuadamente los servidores necesarios en los diferentes países donde se ofrece. Se desea realizar una parte del software necesario para hacer este análisis. Para ello se dispone de la clase `Pais` ya realizada, cuyos objetos almacenan los datos del juego en un pais concreto. Se desea crear la clase `Mundo` que contendrá los datos del juego a escala mundial. Los diagramas de estas clases se muestran aquí:

Pais
+static final int MAX_DIAS=15 ...
+ Pais(String nombre int numInstalaciones) +String getNombre() +int getNumInstalaciones() +actualiza (int incrementoInstalaciones, int actividadDelDia +double getActividadMedia() +double getActividadMediaPonderada() +double getTendencia()

Mundo
-ArrayList<Pais> lista
+ Mundo () +boolean add(Pais p) +double actividadMedia(boolean ponderada) +boolean actualiza(String nombrePais int incrementoInstalaciones, int actividadDelDia) +listarMayorTendencia(double tendenciaMin)

Cada objeto de la clase `Pais` almacena el nombre del país, el número total de instalaciones del juego y datos sobre la actividad de los jugadores medidos en minutos al día, para un número de días máximo igual a la constante `MAX_DIAS`. Los métodos de la clase `Pais` hacen lo siguiente:

- *Constructor*: Construye el país con su nombre y numero de instalaciones pero sin datos de actividad.
- `getNombre()`: Retorna el nombre del pais.
- `getNumInstalaciones()`: Retorna el número total de instalaciones del juego en ese país.
- `actualiza()`: Actualiza los datos del juego añadiendo las instalaciones y actividad media de un dia. Para ello se suma el incremento de instalaciones al numero total de instalaciones y se añade a los datos registrados la actividad del dia (en minutos), descartándose la actividad mas antigua si el número de días registrados es igual al máximo.
- `getActividadMedia()`: Retorna la actividad media medida en minutos al dia para los últimos `MAX_DIAS`.
- `getActividadMediaPonderada()`: Retorna la actividad media ponderada medida en minutos al dia para los últimos `MAX_DIAS`, dando mayor peso a los últimos días y menos a los días anteriores.

- getTendencia(): Retorna la tendencia de los últimos MAX_DIAS, que es el crecimiento de la actividad diaria en ese periodo, en minutos por dia

La clase Mundo almacena una lista de objetos de la clase Pais, conteniendo datos del uso del juego, con el objetivo de poder estimar las necesidades mundiales de servidores. Sus métodos deberán hacer lo siguiente:

- *Constructor*: Crea la lista vacía.
- add(): Ánade un nuevo país a la lista, siempre que no exista; si se ha añadido correctamente se retorna true, pero si ya existía ese país se retorna false.
- actividadMedia(): Retorna la media de las actividades medias de cada país de la lista, en minutos al dia. Si ponderada vale true se usan las actividades medias ponderadas y si no las actividades medias normales. Si la lista está vacía se retorna Double.NaN.
- actualiza(): Actualiza los datos del país cuyo nombre se indica, con el incremento del número de instalaciones y la actividad del día (minutos) indicados. Si se hace correctamente se retorna true. En cambio, si el país con ese nombre no existe se retorna false.
- listarMayorTendencia(): Muestra en pantalla un listado de los países cuya tendencia de crecimiento de actividad es superior al parámetro tendenciaMin expresado en minutos al dia. Se pone primero una línea de encabezamiento con el texto "Listado de países con tendencia de crecimiento superior a " seguido del valor del parámetro tendenciaMin. A continuación una segunda línea de encabezamiento que explique los datos que vendrán a continuación. Finalmente se pone un país por línea, con el nombre del país, número de instalaciones, actividad media ponderada y tendencia.

Nota: Puesto que los métodos add y actualiza deben buscar en la lista un país por su nombre, se valorará hacer esta búsqueda en un único método privado (1 punto).

Finalmente se pide escribir un programa principal de prueba que utilice todos los métodos de la clase Mundo mostrando los resultados de la prueba en pantalla. Los métodos que retornan booleanos deben invocarse tanto en la situación en la que retornan false como en la que retornan true, mostrándose estos valores retornados en pantalla.

Valoración:

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

2) Método privado de búsqueda: 1 punto

3) Métodos add y actualiza: 0.5 puntos cada uno

4) Resto de los métodos: 0.75 puntos cada uno

5) Programa principal: 1 punto