Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2011

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

1) Escribir en *Java* el siguiente algoritmo descrito en pseudocódigo, que calcula los datos de una regresión lineal a partir de dos arrays de puntos `x[]` e `y[]` del mismo tamaño y los almacena en los atributos reales `pendiente` e `intersección`, pertenecientes a la misma clase que el método.

```
método regresionLineal (array de reales x, array de reales y)
   real sumX=0, sumY=0, sumXY=0, sumX2=0
   entero n=0
   para i desde 0 hasta longitud de x-1 hacer
      sumX=sumX+x[i]
      sumY=sumY+y[i]
      sumXY=sumXY+x[i]*y[i]
      sumX2=sumX2+x[i]*x[i]
      n=n+1;
   fin para
   pendiente=(n*sumXY-sumX*sumY)/(n*sumX2-sumX*sumX)
   interseccion=(sumY-pendiente*sumX)/n
fin método
```

La regresión lineal es una técnica que permite obtener la pendiente y punto de intersección de la recta que más se aproxima a la función $y=f(x)$, definiéndose esta función como un conjunto de parejas (x,y).

2) Se desea escribir un *método Java* para discriminar un código postal que se pasa como parámetro en forma de String de 5 caracteres, de modo que se retorne un carácter según la siguiente tabla:

Códigos Postales	Comunidad	Carácter a retornar
28xxx	Madrid	'M'
08xxx o 25xxx	Cataluña	'C'
resto de códigos	Resto	'R'

Nota: xxx representa cualquier combinación de tres cifras

La cabecera del método será:

```
public static char zonaPostal (String codigoPostal)
```

3) Escribir el *pseudocódigo* de un algoritmo iterativo que calcula la suma del siguiente desarrollo en serie de la función e^x:

$$\sum_{i=0}^{n} \frac{x^i}{i!}$$

Para hacer más eficiente este algoritmo, se debe crear una variable para el numerador del desarrollo (x^i) y otra variable para el denominador ($i!$). Observar que de una iteración a la siguiente, el numerador debe multiplicarse por x, pues $x^{i+1}=x^i*x$; el denominador debe multiplicarse por el siguiente valor de i, pues $(i+1)!=i!*(i+1)$.

4) Un palíndromo (del griego *palin dromein*, volver a ir hacia atrás) es una palabra, número o frase que se lee igual hacia adelante que hacia atrás. Escribir un *método recursivo* en Java con la cabecera mostrada abajo, que retorne un booleano indicando si el String `s` es un palíndromo o no. El caso directo se da cuando la longitud de `s` es 0 o 1, en cuyo caso hay que retornar `true`. En el caso recursivo retornar esta expresión lógica:

(primer carácter de s == último carácter de s) &&
palindromo(substring con todos los caracteres de s excepto el primero y el último)

Cabecera del método:

```
public boolean palindromo(String s)
```

5) Indicar *razonadamente* si las siguientes afirmaciones son correctas o no. Utilizar un máximo de 3 líneas para cada respuesta:

a. La única forma de guardar datos en una base de datos es mediante un software de gestión de bases de datos.

b. La principal ventaja de una base de datos es que podemos guardar grandes cantidades de información.

c. En una hoja de cálculo las fórmulas de una casilla pueden usar datos de otras casillas de otras hojas.

d. El lenguaje SQL se usa para obtener datos de una hoja de cálculo.

e. Los datos de una base de datos no se pueden leer y escribir desde páginas Web.

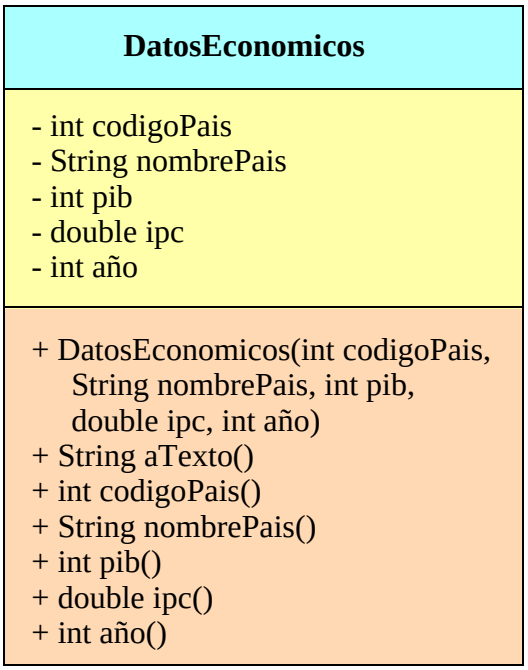
Nota: en esta cuestión, las respuestas correctas suman 0.2 puntos y las incorrectas restan 0.1 puntos. Se valora la precisión de la respuesta.

Examen de Introducción al Software
(Ingeniería Informática)

Febrero 2011

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

Se dispone de una clase ya realizada llamada `DatosEconomicos` que permite almacenar los datos macroeconómicos de un país en un año determinado. El diagrama de clases se muestra en la figura[1]. El significado de los atributos es:



- `codigoPais`: codigo numérico que identifica el país
- `nombrePais`: nombre del país
- `pib`: producto interior bruto, en millones de euros
- `ipc`: índice de precios al consumo, en porcentaje
- `año`: año correspondiente a los datos almacenados

Y los métodos hacen lo siguiente:

- constructor: copia los parámetros en los atributos
- `aTexto`: retorna un texto con los datos económicos
- `codigoPais`, `nombrePais`, `pib`, `ipc`, `año`: métodos observadores que retornan el atributo del mismo nombre.

Se pide implementar en Java la clase `HistoriaEconomica` que sirve para guardar los datos históricos de varios países a lo largo de varios años y que dispone de métodos para obtener información a partir de esta historia.

La clase tiene una lista variable de objetos de la clase `DatosEconomicos`, almacenados en un atributo de tipo `ArrayList<DatosEconomicos>` llamado `lista` (*nota*: no añadir más atributos). Además dispone de los siguientes métodos, que funcionan según sus comentarios de documentación:

1. "+" indica que el método es público y "-" indica que el atributo es privado

```java
public class HistoriaEconomica
{
    // Atributo
    ...

    /**
     * Constructor que crea la lista vacía
     */
    public HistoriaEconomica()...

    /**
     *  Añade el objeto d, que contiene datos económicos, a la lista.
     *  Si el año contenido en d es menor que el año del
     *  último elemento de la lista no lo añade y pone
     *  un mensaje de error en pantalla
     */
    public void añade (DatosEconomicos d)...

   /**
     * Retorna el número de objetos con datos económicos almacenados
     * para el país cuyo nombre se indica
     */
    public int numeroDatos(String pais) ...

   /**
     * Retorna un booleano que indica si hay en la lista algún ipc
     * superior al porcentaje indicado, o no
     */
    public boolean hayIpcSuperior(double porcentaje)...

    /**
     * Busca el año del primer máximo local del IPC del país cuyo
     * código se indica.
     * Un máximo local se da cuando encontramos, para un país
     * concreto, un valor para un año que es superior al del año
     * anterior y también al del año posterior.
     *Retorna -1 si no encuentra ningún máximo para ese país
     */
    public int añoIpcMaximoLocal(int codigo) ...

   /**
     * Retorna el código del país para el que se da el IPC más grande
     * entre los años inicial y final indicados
     */
    public int paisIpcMaximo(int añoInicial, int añoFinal) ...

}
```

Para el método `numeroDatos`() observar que hay que crear un contador con valor incial cero. Mediante un recorrido completo de la lista, se incrementa el contador cada vez que encontramos un objeto cuyo nombre de país coincide con `pais`. Recordar que el nombre del país se obtiene con el método `nombrePais`().

Para el método `hayIpcSuperior`(), observar que se puede usar un algoritmo de búsqueda en el que la condición de búsqueda es que el IPC sea mayor que `porcentaje`. Recordar que el IPC de un objeto de la lista se obtiene con el método `ipc`().

Para el método `paisIpcMaximo`() se recomienda calcular el máximo mediante un recorrido de la lista completa, pero trabajando sólo con los datos cuyo año pertenezca al intervalo [`añoInicial`,`añoFinal`]. Además, cada vez que se encuentre un nuevo máximo, anotar en una variable el índice de la casilla de la lista, para poder retornar el código del país cuando finalice el recorrido. Recordar que el año y el código de país de un objeto de la lista se obtiene con los métodos `año`() y `codigoPais`(), respectivamente.

Para el método `añoIPCMaximoLocal` usar el siguiente pseudocódigo:

```
método añoIpcMaximoLocal (entero codigo) retorna entero
   entero anterior=-1 // casilla anterior a la que vamos a comprobar
   entero actual=-1  // casilla que vamos a comprobar ahora;
                     // la posterior es la i
   para i desde 0 hasta tamaño de lista-1 hacer
      DatosEconomicos d= casilla i de lista
      si d.codigoPais()==codigo entonces
         si anterior!=-1 && actual!=-1 entonces
            si ipc de la casilla (anterior) de lista < ipc de la casilla (actual) de lista &&
               ipc de la casilla (i) de lista < ipc de la casilla (actual) de lista
            entonces
               // hemos encontrado un máximo local
               retorna año de la casilla (actual) de lista
            fin si
         fin si
         // actualiza los índices anterior y actual
         anterior=actual;
         actual=i
      fin si
   fin para
   // si llegamos aquí no hemos encontrado ningún máximo local
   retorna -1
fin método
```

Valoración:

1) Constructor, atributo y `añade`: 1 punto

2) Resto de los métodos: 1 punto cada uno.