Examen escrito de Programación I

Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas

12 de septiembre de 2011

- Disponer sobre la mesa en lugar visible un ***documento de identificación*** provisto de fotografía.
- Escribir ***nombre y dos apellidos*** en cada una de las hojas de papel que haya sobre la mesa.
- Comenzar a resolver cada una de las partes del examen ***en una hoja diferente*** para facilitar su corrección por profesores diferentes.
- El tiempo total previsto para realizar el examen es de **tres horas**.
- No está permitido consultar libros ni apuntes, excepto los dos documentos facilitados por los profesores de la asignatura: *Un breve resumen del lenguaje Java* y *Documentación de las clases Java utilizadas en la asignatura*.

Problema 1.º (3.5 puntos)

Un determinado ayuntamiento almacena información relativa a infracciones de tráfico con retirada de puntos cometidas en su municipio en un fichero de texto donde cada línea representa una infracción y responde a la siguiente sintaxis:

```
<infracción> ::= <fecha><sep><matrícula><sep><nif><sep><puntos>
<fecha> ::= <día>"/"<mes>"/"<año>
<matrícula> ::= <numMatrícula><letras>
<nif> ::= <dni>"-"<letra>
<puntos> ::= literal_entero
<sep> ::= "␣"{"␣"}
<día> ::= literal_entero
<mes> ::= literal_entero
<año> ::= literal_entero
<numMatrícula> ::= literal_entero
<dni> ::= literal_entero
<letras> ::= <letra><letra><letra>
<letra> ::= "A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"J"|"K"|"L"
          |"M"|"N"|"P"|"R"|"S"|"T"|"V"|"W"|"X"|"Y"|"Z"
```

Se pide diseñar una clase que tenga un método `main()` que, al ser ejecutado, pregunte de forma reiterada al operador por un NIF. El programa analizará el contenido de un fichero

denominado «infracciones.txt», que cumple con la sintaxis indicada previamente, e informará del total de puntos que ha perdido el conductor identificado por el NIF introducido por el operador. El programa terminará cuando se introduzca como NIF una cadena vacía.

A modo de ejemplo, si el contenido del fichero «infracciones.txt» fuese el del siguiente cuadro, el comportamiento del programa solicitado debería ser como el que se muestra en el cuadro subsiguiente.

```
14/10/2006  1470KJB  30225059-M  3
8/7/2008    2623CKD  99186985-T  2
22/9/2010   2179DCD  59667282-T  2
5/3/2007    1953JCZ   9411830-C  6
18/9/2006   6665FDG  30225059-M  6
30/9/2010   7617HJS  19791790-Y  6
8/11/2010   1470KJL  30225059-M  3
12/12/2009  1213JKG  83558336-J  4
```

```
Escriba un NIF (ENTRAR para acabar): 30225059-M
El conductor 30225059-M ha perdido 12 puntos

Escriba un NIF (ENTRAR para acabar): 12345678-A
El conductor 12345678-A no ha perdido puntos

Escriba un NIF (ENTRAR para acabar):
```

Problema 2.º (2.5 puntos)

La secuencia de Kolakoski es una secuencia infinita compuesta por bloques integrados por unos o por doses. Un bloque puede estar compuesto por un único dígito (1 o 2) o dos dígitos iguales (11 o 22). El dígito utilizado en un bloque tiene que ser distinto al dígito utilizado en los bloques inmediatamente anterior y posterior. Cada uno de los dígitos de la secuencia informa de la longitud de uno de los bloques posteriores. Así, el n-ésimo dígito de la secuencia (que puede ser un uno o un dos) informa de la longitud del $(n+1)$-ésimo bloque de la secuencia (las instrucciones para la construcción de la secuencia que siguen a continuación aclaran este punto).

La secuencia se construye del siguiente modo:

- Por definición, el primer bloque de la secuencia está formado por el dígito **1**
- Este 1 indica que el siguiente bloque (el segundo de la secuencia) tiene longitud 1. Como hay que alternar unos y doses, el siguiente bloque está formado por el dígito 2. Los dos primeros bloques definen la subsecuencia **12**
- El segundo dígito, que es un 2, indica que el tercer bloque tiene longitud 2. Como hay que alternar unos y doses, el tercer bloque está formado por los dígitos 11. Los tres

primeros bloques de la secuencia totalizan cuatro dígitos: **1211**

- El tercer dígito, un 1, quiere decir que el cuarto bloque tiene longitud 1. Por lo tanto, los cuatro primeros bloques de la secuencia son **12112**
- El cuarto dígito, que es un 1, indica que el quinto bloque tiene longitud 1. Los cinco primeros bloques de la secuencia están formados por los dígitos **121121**
- El quinto dígito, que es un 2, indica que el sexto bloque tiene longitud 2. Los seis primeros bloques de la secuencia son **12112122**
- Sucesivamente y de este mismo modo se siguen añadiendo bloques a la secuencia. Por ejemplo, los 47 primeros bloques de la secuencia están formados por los 70 dígitos siguientes:
  12112122122112112212112122112112122122112122121121122122112122122112122122112112

Se pide completar el siguiente método, añadiendo el código Java correspondiente:

```
/**
 * Pre:  longitud > 0
 * Post: devuelve una tabla de enteros con los primeros [longitud]
 *       dígitos de la secuencia de Kolakoski
 */
public static int[] kolakoski(int longitud)
```

Problema 3.º (4 puntos)

Es frecuente que muchos tipos de fichero (ofimáticos, fotográficos y audiovisuales, por ejemplo) incluyan como parte de su contenido metadatos o información relativa a la propia información del fichero, como, por ejemplo, su título, autor o fecha de creación.

Vamos a suponer un determinado tipo de fichero **binario** que incluye al principio de su contenido una cabecera con metadatos acerca del autor, título, año de creación y existencia de derechos de autor sobre su contenido. Esta información se encuentra codificada, como ya se ha dicho, al principio del fichero, a través de secuencias de *bytes* dependientes del metadato concreto (autor, título, año, derechos de autor) y que se explican a continuación:

Autor: el carácter (`char`) 'A', seguido de un dato de tipo `int` que representa la longitud de la cadena de caracteres correspondiente al nombre del autor, seguido de la secuencia de caracteres que forman propiamente el nombre del autor.

Título: una secuencia como la anterior, aunque encabezada por el carácter 'T'(carácter 'T', dato de tipo `int` que representa la longitud de la cadena de caracteres correspondiente al título y secuencia de caracteres del título propiamente dicho).

Año: el carácter 'Y' seguido de un dato de tipo `int` que representa el año de creación.

Derechos de autor: el carácter 'R', seguido de un dato de tipo **`boolean`** que tiene el valor cierto (**`true`**) si el fichero está sujeto a restricciones de uso debido a la existencia de derechos de autor sobre el mismo, o falso (**`false`**), en caso contrario.

En los ficheros con los que vamos a trabajar, cada uno de los cuatro metadatos anteriores aparecen exactamente una vez, aunque el orden en el que pueden aparecer no está determinado (es decir, puede ser cualquiera).

Por ejemplo, el fichero binario correspondiente a este enunciado de examen podría tener una cabecera como la siguiente:

```
<'T', 32, "Examen escrito de Programación I", 'A', 28, "Profesores
de Programación I", 'Y', 2011, 'R', false>
```

donde, por claridad, no se han escrito las secuencias de *bytes* concretas, sino los valores literales de los datos correspondientes, y donde las secuencias de caracteres se han escrito como cadenas de caracteres. Dicho ejemplo indicaría, por este orden, que el título del documento es la cadena de 32 caracteres "Examen escrito de Programación I", el autor es la cadena de 28 caracteres "Profesores de Programación I", el año del documento es el 2011 y que no está sujeto a restricciones relativas a derechos de autor. Los distintos metadatos podrían haber aparecido en otro orden, pero siempre habrá un único elemento correspondiente al título, otro al autor, otro al año y otro a los derechos de autor.

Se pide la construcción de una clase denominada `Metadatos` con los siguientes métodos:

- Un método constructor que permita crear un objeto de la clase. Este método constructor tendrá como parámetro el nombre de un fichero que contenga metadatos como los descritos y, por lo tanto, el nombre del fichero quedará asociado al objeto. El constructor leerá los primeros *bytes* del fichero, obteniendo los metadatos correspondientes al título, autor, año y derechos y los almacenará en atributos adecuados, que también deben definirse.
- Un método denominado `titulo` que devuelva la referencia a una cadena de caracteres (`String`) que corresponde con el título definido en los metadatos del fichero asociado al objeto.
- Un método denominado `autor` que devuelva la referencia a una cadena de caracteres (`String`) que corresponde con el autor definido en los metadatos del fichero asociado al objeto.
- Un método denominado `agno` que devuelva un dato de tipo **`int`** que corresponde con el año definido en los metadatos del fichero asociado al objeto.
- Un método denominado `protegido` que devuelva un dato de tipo **`boolean`** que indica si, de acuerdo con lo especificado en los metadatos del fichero asociado al objeto, existen restricciones de uso debido a la existencia de derechos de autor sobre el mismo.

Solución al problema 1.º

```
package examen2;

import java.io.*;
import java.util.*;

public class puntosTrafico {

   /**
    * Pre:  Existe un fichero de texto denominado "infracciones.txt" que
    *       recoge infracciones de tráfico cometidas en un determinado
    *       ayuntamiento que responde a la sintaxis presentada en el
    *       enunciado del examen.
    * Post: Ha preguntado de forma reiterada al operador por un NIF y, en
    *       función del contenido del fichero "infracciones.txt" ha
    *       informado del total de puntos que ha perdido el conductor
    *       identificado por dicho NIF. El programa ha terminado cuando
    *       se ha introducido como NIF una cadena vacía.
    */
   public static void main(String[] args) {
      Scanner teclado = new Scanner(System.in);
      System.out.print("Escriba␣un␣NIF␣(ENTRAR␣para␣acabar):␣");
      String nif = teclado.nextLine();

      while (nif.length()>0) {
         int puntos = puntosPerdidos(nif);
         if (puntos > 0) {
            System.out.println("El␣conductor␣" + nif + "␣ha␣perdido␣"
                     + puntos + "␣puntos");
         }
         else {
            System.out.println("El␣conductor␣" + nif
                     + "␣no␣ha␣perdido␣puntos");
         }

         System.out.println();
         System.out.print("Escriba␣un␣NIF␣(ENTRAR␣para␣acabar):␣");
         nif = teclado.nextLine();
      }
   }
```

```
   /**
    * Pre:  Existe un fichero de texto denominado "infracciones.txt" que
    *       recoge infracciones de tráfico cometidas en un determinado
    *       ayuntamiento que responde a la sintaxis presentada en el
    *       enunciado del examen.
    * Post: Ha devuelto el número de puntos que, según el contenido del
    *       fichero "infracciones.txt" ha perdido el conductor
    *       identificado por el valor del parámetro [nifBuscado].
    */
   private static int puntosPerdidos(String nifBuscado) {
      Scanner s;
      try {
         s = new Scanner(new File("infracciones.txt"));
         int puntos = 0;

         // Recorrido del fichero
         while (s.hasNextLine()) {
            // Recorrido de una línea del fichero
            Scanner linea = new Scanner(s.nextLine());
            linea.next();                    // se ignora la fecha
            linea.next();                    // se ignora la matrícula
            String nifLinea = linea.next();  // NIF
            if (nifLinea.equalsIgnoreCase(nifBuscado)) {
               puntos += linea.nextInt();    // Suma de los puntos
            }
         }

         s.close();
         return puntos;

      } catch (FileNotFoundException e) {
         System.out.println("El␣fichero␣no␣existe:␣" + e.getMessage());
         return 0;
      }
   }
}
```

Solución al problema 2.º

```
/**
* Pre:  longitud > 0
* Post: devuelve una tabla de enteros con los primeros [longitud]
*       dígitos de la secuencia de Kolakoski
*/
public static int[] kolakoski(int longitud) {
  int[] resultado = new int[longitud];

  /* Declaraciones e inicializaciones */
  // valor del dígito o dígitos que hay que añadir a la secuencia
  int siguienteDigito = 1;

  // Por la definición del primer bloque de la secuencia
  resultado[0] = siguienteDigito;

  // Número de dígitos obtenidos hasta el momento
  int numDigitos = 1;

  // Número de bloques obtenidos hasta el momento
  int numBloques = 1;

  while (numDigitos < longitud) {
     /* Cálculo del siguiente dígito o dígitos a añadir:
      * si 1, es 2; si 2, es 1 */
     siguienteDigito = 3 - siguienteDigito;

     // Se añade un dígito del nuevo bloque
     resultado[numDigitos] = siguienteDigito;

     /* Si la longitud del nuevo bloque es 2 y "cabe" en la tabla
      * se añade el segundo dígito del bloque */
     if ((resultado[numBloques-1] == 2) && (numDigitos + 1 < longitud)) {
        resultado[numDigitos+1] = siguienteDigito;
     }

     /* Se actualiza el número de bloques y dígitos de la secuencia
      * obtenidos */
     numDigitos += resultado[numBloques-1];
     numBloques++;
  }

  return resultado;
}
```

Solución al problema 3.º

```
package examen2;

import java.io.*;

/**
 * Clase destinada a leer y almacenar los metadatos (título, autor, año,
 * existencia de derechos de autor) de un fichero binario con la
 * estructura descrita en en el enunciado del examen.
 */
public class Metadatos {

   // Atributos para almacenar los metadatos que se leerán en el
   // constructor
   private String autor;
   private String titulo;
   private int agno;
   private boolean derechosAutor;
   private String nombreDelFichero;

   /**
    * Constante que indica el número total de metadatos
    */
   private static final int NUMERO_METADATOS = 4;

   /**
    * Pre:  El fichero cuyo nombre es [nombreFichero] es binario y tiene
    *       una cabecera con metadatos codificados del modo indicado en
    *       enunciado del examen.
    * Post: Ha inicializado los atributos de la clase con los metadatos
    *       leídos del fichero cuyo nombre es [nombreFichero]
    */
   public Metadatos(String nombreFichero) {
      nombreDelFichero = nombreFichero;
      leerMetadatos();
   }

   /**
    * Pre:  El fichero cuyo nombre es [nombreDelFichero] es binario y
    *       tiene una cabecera con metadatos codificados del modo
    *       indicado en enunciado del examen.
    * Post: Ha inicializado los atributos de la clase con los metadatos
    *       leídos del fichero cuyo nombre es [nombreDelFichero]
    */
   private void leerMetadatos() {
      try {
         ObjectInputStream f
            = new ObjectInputStream(
                     new FileInputStream(nombreDelFichero));

         // Lectura de los 4 metadatos
```

```
      for (int numMD=0; numMD<NUMERO_METADATOS; numMD++) {
         // [etiqueta] determina el metadato concreto
         char etiqueta = f.readChar();

         if (etiqueta == 'A') {
            autor = leerString(f);
         }
         else if(etiqueta == 'T') {
            titulo = leerString(f);
         }
         else if (etiqueta == 'Y') {
            agno = f.readInt();
         }
         else if (etiqueta == 'R') {
            derechosAutor = f.readBoolean();
         }
      }
      f.close();
   }
   catch (IOException ex) {
      System.out.println("Error al leer el fichero "
               + nombreDelFichero + ": " + ex.getMessage());
   }
}

/**
 * Pre:  Los primeros bytes pendientes de leer de [f] son un dato de
 *       tipo int que representa la longitud L de una cadena de
 *       caracteres C y una secuencia de caracteres que forman dicha
 *       cadena C.
 * Post: Ha devuelto la cadena de caracteres C de longitud L
 *       codificada en los primeros bytes pendientes de leer de [f],
 *       que han sido leídos.
 * @throws IOException Si alguna operación de entrada/salida es
 *         errónea
 */
private String leerString(ObjectInputStream f) throws IOException {
   // Lectura de la longitud
   int longitud = f.readInt();

   // Lectura de la secuencia de caracteres, concatenándolos en un
   // objeto String
   String cadena = "";
   for (int i=0; i<longitud; i++) {
      cadena += f.readChar();
   }
   return cadena;
}

/**
 * Post: Ha devuelto la referencia a la cadena de caracteres que
 *       corresponde con el título definido en los metadatos del
```

```
     *       fichero cuyo nombre se especificó al construir este objeto.
     */
    public String titulo() {
       return titulo;
    }

    /**
     * Post: Ha devuelto la referencia a la cadena de caracteres que
     *       corresponde con el nombre del autor definido en los metadatos
     *       del fichero cuyo nombre se especificó al construir este
     *       objeto.
     */
    public String autor() {
       return autor;
    }

    /**
     * Post: Ha devuelto el dato que corresponde con el año definido en
     *       los metadatos del fichero cuyo nombre se especificó al
     *       construir este objeto.
     */
    public int agno() {
       return agno;
    }

    /**
     * Post: ha devuelto el dato que indica si, de acuerdo con lo
     *       especificado en los metatados del fichero cuyo nombre se
     *       especificó al construir este objeto, existen restricciones de
     *       uso debido a la existencia de derechos de autor sobre el
     *       mismo.
     */
    public boolean protegido() {
       return derechosAutor;
    }
}
```