Teoría de la Información y Codificación

Práctica 1: Codificación económica

José A. Montenegro Montes

23 de octubre de 2011

1. Enunciado

Realice una clase **Fuente** que implemente los siguientes métodos:

getSimbolos(n): Devuelve n símbolos de un Alfabeto según una probabilidad dada.

getCodificacion(): Devuelve un objeto codificación, que incluye el codebook, con respecto al alfabeto binario utilizando regla de Shannon-Fano.

getOptimalCodificacion(): Devuelve un objeto codificación, que incluye el codebook, con respecto al alfabeto binario utilizando regla de Huffman.

getSimbolosCodificados(n): Devuelve dos listas. Primera lista con n símbolos (idem primer caso), Segunda con codificación de los n símbolos según regla de Shannon-Fano.

getSimbolosCodificadosOptimos(n): Devuelve dos listas. Primera lista con n símbolos (idem primer caso), Segunda con codificación de los n símbolos según Regla de Huffman.

Una vez implementadas, verificar las clases con los ejercicios de las transparencias del Tema 3.

Los métodos requeridos no incluyen ni los constructores de la clase como así funciones auxiliares que deben ser implementadas para llevar a cabo los métodos anteriormente descritos. Entre ellos a modo de ejemplo se encuentran los métodos:

Entropía

Longitud media (L)

Regla de Shannon-Fano

Regla Huffman

Constructor (Alfabeto A) : Alfabeto con todos los símbolos equiprobables.

Constructor (Alfabeto A, Probabilidad p) : Alfabeto con distribución de probabilidad p.

setDistribucion(p) : Establece una nueva distribución de probabilidad.

setAlfabetoCodificacion (A_d) : Establece el Alfabeto destino de la codificación, por defecto Binario.

2. Conclusiones

Esta práctica pretende establecer la clase Fuente donde el alumno deberá aplicar los conceptos estudiados en este tema:

- Concepto de Fuente
- Entropía e Incertidumbre
- Longitud Media de Palabra(L)
- Codificación Óptima

3. Código

Esta sección muestra un ejemplo de la clase Fuente y código auxiliar para su creación.

Clase Fuente

```
1  package practica1;
2
3  public class Fuente {
4
5          private Alfabeto alfabeto;
6          private List probabilidades = new ArrayList();
7
8          public Fuente(Alfabeto alfabeto, List probabilidades) {
9          }
10
11         public Fuente(Alfabeto alfabeto) {
12         }
13
14         /**
15          * Metodo que genera la lista de sinbolos
16          * del alfabeto segun sus probabilidades
17          *
```

```
18          * @param numeroSimbolos
19          * @return
20          */
21         public List getSimbolos(int numeroSimbolos){
22         }
23
24         /**
25          * Metodo encargado de obtener un codebook binario
26          * segun Shanon
27          *
28          * @return
29          */
30         public Codebook getCodificacion(){
31         }
32
33         /**
34          * Metodo encargado de obtener un codebook al
35          * alfabeto binario segun Huffman
36          *
37          * @param alfabetoDestino
38          * @return
39          */
40         public Codebook getOptimalCodificacion(){
41
42         }
43
44         /**
45          * Metodo encargado de devolver una lista de
46          * numSimbolos del alfabeto origen
47          * y sus correspondientes codificaciones binarias según Shannon-Fano.
48          *
49          * @param numSimbolos
50          * @return
51          */
52         public SimbolosCodificados getSimbolosCodificados(int numSimbolos){
53         }
54
55         /**
56          * Metodo encargado de devolver una lista de
57          * numSimbolos del alfabeto origen
58          * y sus correspondientes codificaciones binarias en el
59          * alfabeto destino segun hufman
60          *
61          * @param numSimbolos
62          * @return
63          */
64         public SimbolosCodificados getSimbolosCodificadosOptimos(int numSimbolos){
65         }
66
67
```

```
68          //======================metodos auxiliares==================
69
70
71          //====================get and set=========================
72
73          public Alfabeto getAlfabeto() {
74                  return alfabeto;
75          }
76
77          public void setAlfabeto(Alfabeto alfabeto) {
78          }
79
80          public void setAlfabetoCodificacion(Alfabeto alfabeto) {
81          }
82
83          public List getProbabilidades() {
84                  return probabilidades;
85          }
86
87
88          public void setProbabilidades(List probabilidades) {
89          }
90
91
92  }
```

Clase CodeBook

```
1   package Practica1;
2
3   import java.util.ArrayList;
4   import java.util.List;
5   import java.util.TreeMap;
6
7   /**
8    * Un CodeBook representa un conjunto de símbolos de un alfabeto
9    * junto con su codificación en otro alfabeto.
10   *          s1 -> c(s1)
11   *          s2 -> c(s2)
12   *          ...
13   *          sn -> c(sn)
14
15   *
16   */
17  public class CodeBook {
18
19          /**
20           * Los elementos se almacenan ordenados lexicograficamente.
```

```
21          */
22         private TreeMap<Character, String> codebook;
23 
24         /**
25          * Crea un codebook vacío.
26          */
27         public CodeBook() {
28                 this.codebook = new TreeMap<Character, String>();
29         }
30 
31         /**
32          * Añade un elemento al codebook.
33          *
34          * @param c         Símbolo.
35          * @param s         Símbolo codificado.
36          */
37         public void add(char c, String s) {
38                 this.codebook.put(c, s);
39         }
40 
41         /**
42          * Elimina un elemento del codebook.
43          *
44          * @param c         Símbolo a eliminar.
45          */
46         public void erase(char c) {
47                 this.codebook.remove(c);
48         }
49 
50         /**
51          * Obtiene el código de un símbolo.
52          *
53          * @param c         Símbolo.
54          * @return         Símbolo codificado.
55          */
56         public String get(char c) {
57                 return this.codebook.get(c);
58         }
59 
60         /**
61          *
62          * @return         Todos los códigos del codebook.
63          */
64         public List<String> codes() {
65                 return new ArrayList<String>(this.codebook.values());
66         }
67 
68         /**
69          *
70          * @return         Todos los símbolos del codebook.
```

```
71          */
72         public List<Character> simbols() {
73                 return new ArrayList<Character>(this.codebook.keySet());
74         }
75
76         /**
77          * Tamaño del codebook.
78          *
79          * @return        Número de elementos del codebook.
80          */
81         public int size() {
82                 return codebook.size();
83         }
84
85         @Override
86         public String toString() {
87                 String res = new String();
88                 for (char c : this.codebook.keySet()) {
89                         res = res.concat(c + " -> " + get(c) + '\n');
90                 }
91                 return res;
92         }
93 }
```

Clase SimbolosCodificados

```
1  package practica1;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.List;
6
7  /**
8   *
9   * Clase encargada de encapsular una
10  * lista de simbolos y sus codificaciones
11  *
12  */
13 public class SimbolosCodificados {
14
15         private List simbolos = new ArrayList();
16         private List simbolosCodificados = new ArrayList();
17
18
19         public SimbolosCodificados(List simbolos, List simbolosCodificados) {
20                 super();
21                 this.simbolos = simbolos;
22                 this.simbolosCodificados = simbolosCodificados;
```

```
23          }
24
25          //======================get and set=========================
26
27          public List getSimbolos() {
28                  return simbolos;
29          }
30
31          public void setSimbolos(List simbolos) {
32                  this.simbolos = simbolos;
33          }
34
35          public List getSimbolosCodificados() {
36                  return simbolosCodificados;
37          }
38
39          public void setSimbolosCodificados(List simbolosCodificados) {
40                  this.simbolosCodificados = simbolosCodificados;
41          }
42
43
44  }
```