

## P1.-

Hay ciertas palabras que se escriben igual en distintos idiomas, aunque ocasionalmente tengan significados distintos. Por ejemplo, la palabra "hotel" se escribe igual y significa lo mismo en español, inglés y francés. Por su parte, la palabra "budget" se escribe igual, aunque tiene distintos significados en inglés y francés. Al respecto:

**A) (Ponderación:25%)** Escriba un método de encabezamiento String buscar(String X,String Y) que entregue el significado de la palabra X en el diccionario grabado en el archivo de texto de nombre Y.

### Notas

1. Cada línea del archivo contiene una palabra, un punto, y, a continuación, el significado
2. El archivo está ordenado ascendentemente por palabra (y no presenta repeticiones)
3. Si la palabra no existe se debe entregar el valor null

**B) (Ponderación:75%)**

Escriba un método que escriba en la pantalla los significados de una palabra en los idiomas español, inglés y francés respectivamente, en la forma indicada en los siguientes ejemplos:

significados("hotel");	significados("budget");	significados("hola");
español=...	español=no existe	español=...
inglés=...	inglés=...	inglés=no existe
francés=...	francés=...	francés=no existe

### Notas

1. Los diccionarios están grabados en los archivos de nombres "español", "inglés" y "francés"
2. La búsqueda en los 3 diccionarios debe realizarse concurrentemente en threads independientes.

### SOL:

**A)**

```
1 public static String buscar (String X, String Y) throws IOException{
2     String linea, palabra, significado=" ";
3     boolean encontrado = false;
4     BufferedReader bl=new BufferedReader(new FileReader(Y));
5     while((linea=bl.readLine())!=null){
6         palabra = linea.substring(0,linea.indexOf(" "));
7         significado = linea.substring(linea.indexOf(" ")+1);
8         if(palabra.equals(X)){
9             encontrado = true;
10            break;
11        }
12    }
13    bl.close();
```

```

14         if(encontrado)
15             return significado;
16         else
17             return null;
18     }

```

**B)**

```

1  public static void busca_threads(String palabra){
2      String s1, s2, s3;
3      System.out.println("Significados:\""+palabra+"\"");
4      buscador esp = new buscador(palabra, "español");
5      buscador ing = new buscador(palabra, "ingles");
6      buscador fra = new buscador(palabra, "frances");
7      esp.start();
8      ing.start();
9      fra.start();
10     try{
11         esp.join();
12         ing.join();
13         fra.join();
14     }catch(InterruptedException e){
15     }
16     s1 = esp.significado();
17     s2 = ing.significado();
18     s3 = fra.significado();
19     if(s1==null)
20         System.out.println("español=no existe");
21     else
22         System.out.println("español="+s1);
23     if(s2==null)
24         System.out.println("ingles=no existe");
25     else
26         System.out.println("ingles="+s2);
27
28     if(s3==null)
29         System.out.println("frances=no existe");
30     else
31         System.out.println("frances="+s3);
32 }

```

```

1  import java.io.*;
2  class buscador extends Thread{
3      String palabra, diccionario, significado;
4      public buscador(String X, String Y){
5          palabra = X;
6          diccionario = Y;
7      }
8      public void run(){
9          try{
10             significado = buscar(palabra, diccionario);

```

```

11         }catch(IOException e){
12         }
13     }
14     public static String buscar (String X, String Y) throws IOException{
15         String linea, palabra, significado=" ";
16         boolean encontrado = false;
17         BufferedReader b1=new BufferedReader(new FileReader(Y));
18         while((linea=b1.readLine())!=null){
19             palabra = linea.substring(0,linea.indexOf(" "));
20             significado = linea.substring(linea.indexOf(" ")+1);
21             if(palabra.equals(X)){
22                 encontrado = true;
23                 break;
24             }
25         }
26         b1.close();
27         if(encontrado)
28             return significado;
29         else
30             return null;
31     }
32     public String significado(){
33         return significado;
34     }
35 }

```

## P2.-

### A) (Ponderación:25%)

Escriba un método de encabezamiento void resta(String X,String Y,String Z) que realice la operación  $Z=X-Y$  entre los conjuntos que están grabados en los archivos de texto de nombres X e Y, dejando el resultado en el archivo de nombre Z.

#### Notas.

1. Los archivos no están ordenados y cada línea contiene el nombre de una persona.
2. El archivo de nombre Z debe contener las personas de X que no están en Y

**B) (Ponderación:75%)** Los alumnos de una universidad pueden cursar las asignaturas A,B,C, D y E en un semestre. Escriba un programa que grabe en el archivo de texto "soloC" la lista de los alumnos que sólo están cursando la asignatura C. Las listas de los nombres de los alumnos de cada curso están grabadas en los archivos "A", "B", "C", "D" y "E".

Para conseguir el resultado, el programa debe usar threads que produzcan concurrentemente los resultados de C-A, C-B, C-D y C-E. A continuación, realice concurrentemente  $C-A \cap C-B$  y  $C-D \cap C-E$ . Finalmente interseque los resultados de las intersecciones anteriores.

Nota. Suponga que dispone también del método inter(X,Y,Z) que realiza  $Z=X \cap Y$ .

### SOL:

#### A)

```
1 public static void resta(String X, String Y, String Z) throws IOException{
2     boolean escribir;
3     String lineax, lineay;
4     BufferedReader bx = new BufferedReader(new FileReader(X));
5     BufferedReader by;
6     PrintWriter pz = new PrintWriter(new FileWriter(Z));
7     while((lineax=bx.readLine())!=null){
8         escribir = true;
9         by = new BufferedReader(new FileReader(Y));
10        while((lineay=by.readLine())!=null){
11            if(lineax.equals(lineay)){
12                escribir = false;
13                break;
14            }
15        }
16        by.close();
17        if(escribir)
18            pz.println(lineax);
19        }
20        bx.close();
21        pz.close();
22    }
23 }
```

#### B)

```

1 public static void solo_c(){
2     restador ca = new restador ("C", "A", "CA");
3     restador cb = new restador ("C", "B", "CB");
4     restador cd = new restador ("C", "D", "CD");
5     restador ce = new restador ("C", "E", "CE");
6     ca.start();
7     cb.start();
8     cd.start();
9     ce.start();
10    try{
11        ca.join();
12        cb.join();
13        cd.join();
14        ce.join();
15    }catch(InterruptedException e){
16    }
17    inter("CA", "CB", "CA_inter_CB");
18    inter("CD", "CE", "CD_inter_CE");
19    inter("CA_inter_CB", "CD_inter_CE", "soloC")
20    }
21 }

```

```

1 import java.io.*;
2 class restador extends Thread{
3     String X, Y, Z;
4     public restador(String X, String Y, String Z){
5         this.X = X;
6         this.Y = Y;
7         this.Z = Z;
8     }
9     public void run(){
10        try{
11            resta(X, Y, Z);
12        }catch(IOException e){
13        }
14    }
15    public static void resta(String X, String Y, String Z) throws
IOException{
16        boolean escribir;
17        String lineax, lineay;
18        BufferedReader bx = new BufferedReader(new FileReader(X));
19        BufferedReader by;
20        PrintWriter pz = new PrintWriter(new FileWriter(Z));
21        while((lineax=bx.readLine())!=null){
22            escribir = true;
23            by = new BufferedReader(new FileReader(Y));
24            while((lineay=by.readLine())!=null){
25                if(lineax.equals(lineay)){
26                    escribir = false;
27                    break;
28                }

```

```
29         }
30         by.close();
31         if(escibir)
32             pz.println(lineax);
33     }
34     bx.close();
35     pz.close();
36 }
37 }
```