

PRÁCTICA PSP01

Programación multiproceso

22/10/2022

CÉSAR BOUZAS SOTO

Contenido

1	Ejercicio 1) Se compone de 3 partes:.....	2
1.1	Implementa una aplicación que ordena un conjunto indeterminado de números que recibe a través de su entrada estándar; y muestra el resultado de la ordenación en su salida estándar. La aplicación se llamará 'ordenarNumeros'.....	2
1.2	Implementa una aplicación, llamada 'aleatorios', que genere al menos 40 números aleatorios (entre 0 y 100), y que los escriba en su salida estándar.	3
1.3	Realiza un pequeño manual (tipo "¿Cómo se hace?" o "HowTo"), utilizando un editor de textos (tipo word o writer) en el que indiques, con pequeñas explicaciones y capturas, cómo has probado la ejecución de las aplicaciones que has implementado en este ejercicio. Entre las pruebas que hayas realizado, debes incluir una prueba en la que utilizando el operador " " (tubería) redirijas la salida de la aplicación 'aleatorios' a la entrada de la aplicación 'ordenarNumeros'.....	4
1.3.1	Copiamos los jar en una carpeta.....	4
1.3.2	Entramos en la consola de sistema en la carpeta donde se encuentran los *.jar.	4
1.3.3	Ejecución de java -jar java -jar.....	4
2	Ejercicio 2) Se compone de 3 partes:.....	5
2.1	Implementa una aplicación que escriba en un fichero conjuntos de letras generadas de forma aleatoria. Escribiendo cada conjunto de letras en una línea distinta. El número de conjuntos de letras a generar por el proceso, también será dado por el usuario en el momento de su ejecución. Se Llamará Lenguaje.	5
2.2	Segunda parte: implementa una aplicación, llamada 'colaborar', que lance al menos 10 instancias de la aplicación "lenguaje". Haciendo que todas ellas, colaboren en generar un gran fichero de palabras. Cada instancia generará un número creciente de palabras de 10, 20, 30, ... Por supuesto, cada proceso seguirá escribiendo su palabra en una línea independiente de las otras. Es decir, si lanzamos 10 instancias de "lenguaje", al final, debemos tener en el fichero 10 + (Sumar.) 20 + 30 + ... + 100 = (Igual.) 550 líneas.	6
2.3	Realiza un pequeño manual (tipo "¿Cómo se hace?" o "HowTo"), utilizando un editor de textos (tipo word o writer) en el que indiques, con pequeñas explicaciones y capturas, cómo has probado la ejecución de las aplicaciones que has implementado en este ejercicio.	7

Reposito 1 Tubería

Repositorio2 Lenguaje

1 Ejercicio 1) Se compone de 3 partes:

1.1 Implementa una aplicación que ordena un conjunto indeterminado de números que recibe a través de su entrada estándar; y muestra el resultado de la ordenación en su salida estándar. La aplicación se llamará 'ordenarNumeros'.

Recibe uno por uno desde la entrada del sistema un serie de enteros , los ordena y los imprime entre corchetes y separados por comas.

Código Ordenar números.

```
package ordenarnumero;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

/**
 *
 * @author Cesar bouzas
 */
public class OrdenarNumero {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        ArrayList numbers = new ArrayList();
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextInt()) {
            numbers.add(sc.nextInt());
        }
        Collections.sort(numbers);
        for (int i = 0; i < numbers.size(); i++) {
            if (i == 0) {
                System.out.print("[");
            }
            if (i != (numbers.size() - 1)) {
                System.out.print(numbers.get(i) + ",");
            }
            if (i == numbers.size() - 1) {
                System.out.print(numbers.get(i) + "]" );
            }
        }
    }
}
```

Ilustración 1 Ordenar números

1.2 Implementa una aplicación, llamada 'aleatorios', que genere al menos 40 números aleatorios (entre 0 y 100), y que los escriba en su salida estándar.

Genera 40 números aleatorios entre 0 y 100.

[Código Aleatorio.](#)

```
package aleatorios;

/**
 * Esta clase crea aleatorios entre 0 y 40
 * @author: Cesar Bouzas Soto
 * @version: 221016
 */
public class Aleatorios {

    public static void main(String[] args) {
        for(int i=0;i<40;i++){
            System.out.println((int) (Math.random()*100));
        }
    }
}
```

Ilustración 2 Código Aleatorio

```
C:\Users\Usuario\Desktop\html_public\DAM\M
r
67
45
82
76
50
97
4
17
43
57
75
26
59
41
21
27
74
13
42
46
28
84
16
71
7
18
70
52
57
79
55
7
```

Ilustración 3 salida de 40 números aleatorios.

1.3 Realiza un pequeño manual (tipo "¿Cómo se hace?" o "HowTo"), utilizando un editor de textos (tipo word o writer) en el que indiques, con pequeñas explicaciones y capturas, cómo has probado la ejecución de las aplicaciones que has implementado en este ejercicio. Entre las pruebas que hayas realizado, debes incluir una prueba en la que utilizando el operador "|" (tubería) redirijas la salida de la aplicación 'aleatorios' a la entrada de la aplicación 'ordenarNumeros'.

1.3.1 Copiamos los jar en una carpeta.

Mediante Clean and build del IDE Netbeans generamos los Jar de cada clase, los colocamos en la misma carpeta.

[Repositorio de la carpeta con los *.jar.](#)

1.3.2 Entramos en la consola de sistema en la carpeta donde se encuentran los *.jar.

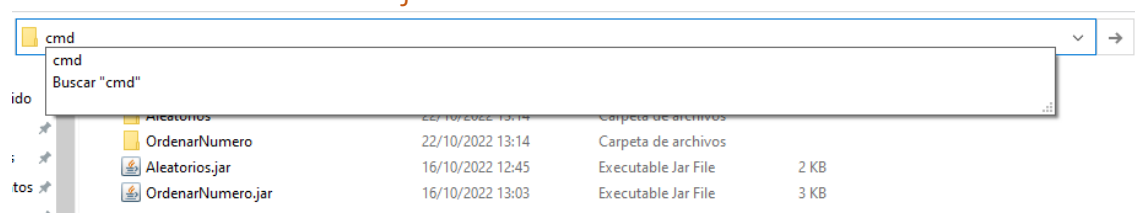


Ilustración 4 cambiamos la dirección por cmd.

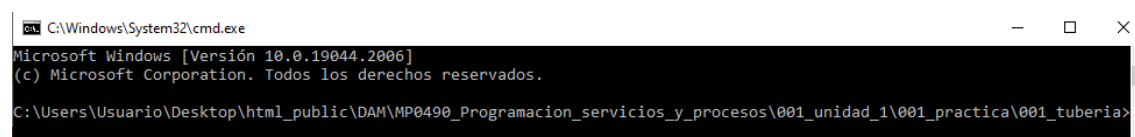


Ilustración 5 Consola del sistema.

1.3.3 Ejecución de java -jar | java -jar

Mediante la opción llamada tubería, los datos de salida de un procedimiento se introducen como entrada en el otro.

Si escribimos en línea de comandos `java -jar Aleatorios.jar | java -jar OrdenarNumero.jar` obtenemos 40 números aleatorios ordenados de menor a mayor.

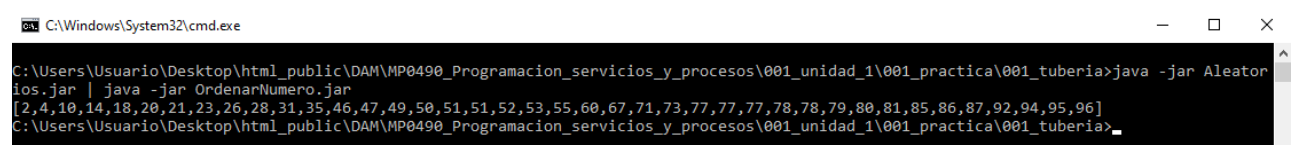


Ilustración 6 Salida de la tubería Aleatorio | Ordenar.

2 Ejercicio 2) Se compone de 3 partes:

2.1 Implementa una aplicación que escriba en un fichero conjuntos de letras generadas de forma aleatoria. Escribiendo cada conjunto de letras en una línea distinta. El número de conjuntos de letras a generar por el proceso, también será dado por el usuario en el momento de su ejecución. Se Llamará Lenguaje.

[Repositorio de Lenguaje.class](#)

Genera y guarda en archivo ,un numero n de palabras de máximo 8 letras (alfabéticas) indicado por consola.

```

1  /*
2  public static void main(String[] args) {
3
4      int n = Integer.parseInt(args[0]);
5      String title="*****PROCESO DE *** PALABRAS*****";
6      System.out.println(title);
7      File f = createFile(args[1]);
8      writeRAF(f,title,n);
9  }
10
11  public static File createFile(String r){
12      File f = new File(r);
13      String sName = System.getProperty("os.name").toLowerCase();
14
15      // if (sName.contains("win")) {
16      // if (f.exists()) {
17      //     f = new File(r.toLowerCase().replace("/", "\\"));
18      // } else {
19      //     f = new File("c:\\\\FicherosDeLenguaje.txt");
20      // }
21
22      if (!f.exists()) {
23          try {
24              f.createNewFile();
25              System.out.println("Fichero creado : "+f.getCanonicalPath());
26          } catch (IOException e) {
27              System.out.println("Error de entrada salida createFile 1");
28          }
29      } else {
30          try {
31              System.out.println("El fichero "+f.getCanonicalPath()+" ya existe. ");
32          } catch (IOException e) {
33              System.out.println("Error de entrada salida createFile 2");
34          }
35      }
36      return f;
37  }
38
39  public static String createWord() {
40      int n = 1 + (int) (Math.random() * 8);
41      String word = "";
42      for (int i = 0; i < n; i++) {
43          word += (randomChar());
44      }
45      return word;
46  }
47
48  public static void writeRAF (File f,String title ,int n) {
49      FileOutputStream fos=null;
50      String s;
51
52      try {
53          RandomAccessFile rAF = new RandomAccessFile(f, "rwd");
54
55      }
56      {
57          fos = rAF.getChannel().lock();
58          rAF.seek(f.length());
59          rAF.writeChars(title+"\n");
60          for (int i = 0; i < n; i++) {
61              s = "Proceso[" + (i + 1) + "/" + n + "] palabra ----> " + createWord();
62              rAF.writeBytes(s+"\n");
63              System.out.println(s);
64          }
65          fos.release();
66          fos = null;
67      } catch (IOException e) {
68          System.out.println("Error de entrada/salida(writeRAF)");
69          e.printStackTrace();
70      } finally {
71      }
72  }
73
74  public static char randomChar() {
75      int randomInt=(int) (Math.random()*(122-97));
76      //System.out.println(randomInt+" ----> "+(char)(randomInt));
77      return (char)randomInt;
78  }
79  }

```

Ilustración 7 Código lenguaje.class

2.2 Segunda parte: implementa una aplicación, llamada 'colaborar', que lance al menos 10 instancias de la aplicación "lenguaje". Haciendo que todas ellas, colaboren en generar un gran fichero de palabras. Cada instancia generará un número creciente de palabras de 10, 20, 30, ... Por supuesto, cada proceso seguirá escribiendo su palabra en una línea independiente de las otras. Es decir, si lanzamos 10 instancias de "lenguaje", al final, debemos tener en el fichero $10 + (\text{Sumar.}) 20 + 30 + \dots + 100 = (\text{Igual.}) 550$ líneas.

[Repositorio Colaborar.class](#)

```
package vista;

import java.io.File;
import java.io.IOException;
public class Colaborar {

    public static void main(String[] args) {
        Process newProcess=null;
        File f=new File("archivo.txt");
        try{
            for (int i = 10; i <=100; i=i+10){
                String comand="java -jar Lenguaje.jar "+i+" "+f.getName();
                newProcess = Runtime.getRuntime().exec(comand);
                System.out.println(comand+" ----->Lanzado");
            }
        }catch (SecurityException ex){
            System.err.println("Ha ocurrido un error de Seguridad."+
                "No se ha podido crear el proceso por falta de permisos.");
        }catch (IOException ex){
            System.err.println("Ha ocurrido un error, Entrada/Salida.");
        }catch(Exception ex){
            System.out.println("Error genérico");
        }finally{

        }
    }
}
```

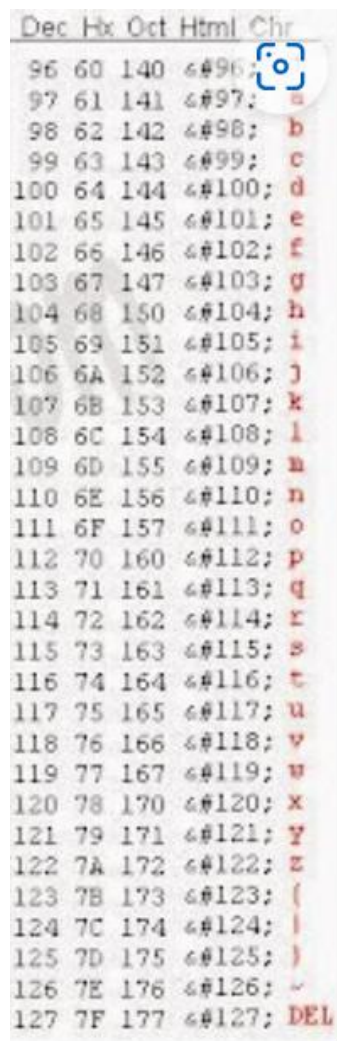
Ilustración 8 Código de colaborar.class

2.3 Realiza un pequeño manual (tipo "¿Cómo se hace?" o "HowTo"), utilizando un editor de textos (tipo word o writer) en el que indiques, con pequeñas explicaciones y capturas, cómo has probado la ejecución de las aplicaciones que has implementado en este ejercicio.

Por su lado lenguaje crea palabras compuestas por un numero aleatorio de caracteres de 1 a 8 , comprendidos dentro del código ascii comprendidos entre el 96->a y el 122->z.

```
public static char randomChar(){
    int randomInt=(int) (97+Math.random()*(122-97));
    //System.out.println(randomInt+" ----> "+(char) (randomInt));
    return (char)randomInt;
}
```

Ilustración 9 generación de ascii aleatorio entre a-z



Dec	Hx	Oct	Html	Chr
96	60	140	`	a
97	61	141	a	b
98	62	142	b	c
99	63	143	c	d
100	64	144	d	e
101	65	145	e	f
102	66	146	f	g
103	67	147	g	h
104	68	150	h	i
105	69	151	i	j
106	6A	152	j	k
107	6B	153	k	l
108	6C	154	l	m
109	6D	155	m	n
110	6E	156	n	o
111	6F	157	o	p
112	70	160	p	q
113	71	161	q	r
114	72	162	r	s
115	73	163	s	t
116	74	164	t	u
117	75	165	u	v
118	76	166	v	w
119	77	167	w	x
120	78	170	x	y
121	79	171	y	z
122	7A	172	z	{
123	7B	173	{	
124	7C	174	|	}
125	7D	175	}	~
126	7E	176	~	DEL
127	7F	177		

Ilustración 10 tabla ascii

Cada uno de los procesos imprime un titulo donde n es el número de palabras a generar que recibe como primer argumento.

```
int n = Integer.parseInt(args[0]);
String title="*****PROCESO DE "+n+" PALABRAS*****";
```

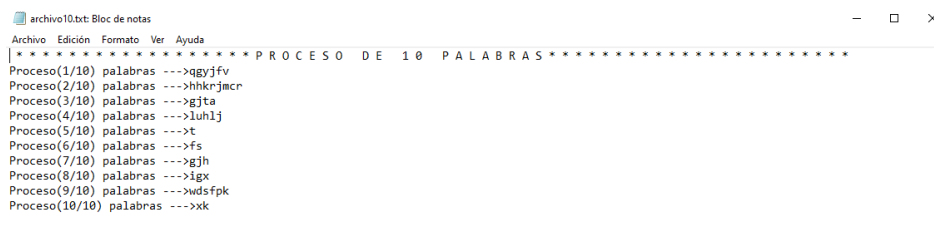
Ilustración 11 Generación de titulo

Por si solo Lenguaje jar crea una cantidad de palabras que le pasamos como primer parámetro y como segundo el archivo donde las debe guardar: java -jar lenguaje.jar 10 archivo10.txt



```
C:\Windows\System32\cmd.exe
C:\Users\Usuario\Desktop\html_public\DAM\MP0490_Programacion_servicios_y_procesos\001_unidad_1\001_practica\002_colaborar>java -jar Lenguaje.jar 10 archivo10.txt
*****PROCESO DE 10 PALABRAS*****
Fichero creado :C:\Users\Usuario\Desktop\html_public\DAM\MP0490_Programacion_servicios_y_procesos\001_unidad_1\001_practica\002_colaborar\archivo10.txt
Proceso(1/10) palabras --->qgyjfv
Proceso(2/10) palabras --->hkhkrjmc
Proceso(3/10) palabras --->gjta
Proceso(4/10) palabras --->luhlj
Proceso(5/10) palabras --->t
Proceso(6/10) palabras --->fs
Proceso(7/10) palabras --->gjh
Proceso(8/10) palabras --->igx
Proceso(9/10) palabras --->wdsf
Proceso(10/10) palabras --->xk
C:\Users\Usuario\Desktop\html_public\DAM\MP0490_Programacion_servicios_y_procesos\001_unidad_1\001_practica\002_colaborar>
```

Ilustración 12 java -jar lenguaje.jar 10 archivo10.txt

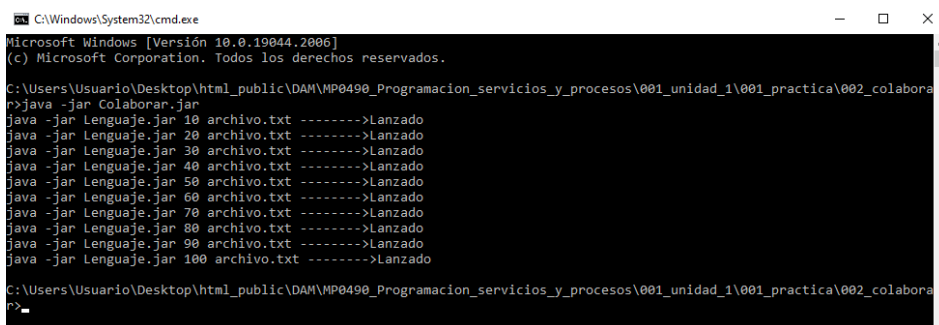


```
archivo10.txt: Bloc de notas
*****PROCESO DE 10 PALABRAS*****
Proceso(1/10) palabras --->qgyjfv
Proceso(2/10) palabras --->hkhkrjmc
Proceso(3/10) palabras --->gjta
Proceso(4/10) palabras --->luhlj
Proceso(5/10) palabras --->t
Proceso(6/10) palabras --->fs
Proceso(7/10) palabras --->gjh
Proceso(8/10) palabras --->igx
Proceso(9/10) palabras --->wdsf
Proceso(10/10) palabras --->xk
```

Ilustración 13 contenido del archivo

Se indica con el titulo cual es el proceso que toma el control de la escritura , como proceso esta bloqueado debe empezar hasta terminar , no se garantiza el orden pues esto lo decide el gestor de procesos del SO , encargado de determinar el momento que cada proceso entra en el CPU.

La aplicación colaborar lo único que hace es llamar 10 veces a Lenguaje.jar introduciéndole como parámetro 10,20,,30...100 lo que hace que tengamos en el archivo.txt 550 líneas como además he decidido introducir un titulo debemos tener 550+10=560 líneas .



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.2006]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\Usuario\Desktop\html_public\DAM\MP0490_Programacion_servicios_y_procesos\001_unidad_1\001_practica\002_colaborar>java -jar Colaborar.jar
java -jar Lenguaje.jar 10 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 20 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 30 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 40 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 50 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 60 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 70 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 80 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 90 archivo.txt ----->Lanzado
java -jar Lenguaje.jar 100 archivo.txt ----->Lanzado
C:\Users\Usuario\Desktop\html_public\DAM\MP0490_Programacion_servicios_y_procesos\001_unidad_1\001_practica\002_colaborar>
```


Ilustración 14 Ejecución java -jar Colaborar.jar

PRÁCTICA 1 Programación multiproceso – ALUMNO: CESAR BOUZAS SOTO

```
archivo.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
***** PROCESO DE 10 PALABRAS *****
Proceso(1/10) palabras --->rvvllaou
Proceso(2/10) palabras --->b
Proceso(3/10) palabras --->wjemad
Proceso(4/10) palabras --->fwqcp
Proceso(5/10) palabras --->cxlyqd
Proceso(6/10) palabras --->ddvhyx
Proceso(7/10) palabras --->hkaupvj
Proceso(8/10) palabras --->y
Proceso(9/10) palabras --->cbaofb
Proceso(10/10) palabras --->k
***** PROCESO DE 60 PALABRAS *****
Proceso(1/60) palabras --->ygxftfid
Proceso(2/60) palabras --->mnjneny
Proceso(3/60) palabras --->rp
Proceso(4/60) palabras --->uea
Proceso(5/60) palabras --->bq
Proceso(6/60) palabras --->msvnoi
Proceso(7/60) palabras --->vmh
Proceso(8/60) palabras --->xxbprll
Proceso(9/60) palabras --->wmyohjx
Proceso(10/60) palabras --->svftvel
Proceso(11/60) palabras --->rgqqeefs
Proceso(12/60) palabras --->iahunfnf
Proceso(13/60) palabras --->ag
Proceso(14/60) palabras --->kxa
Proceso(15/60) palabras --->gpwhr
Proceso(16/60) palabras --->lhnn
```

Ilustración 15 contenido de archivo.txt

```
archivo.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Proceso(26/30) palabras --->ttka
Proceso(27/30) palabras --->ihmbghel
Proceso(28/30) palabras --->mhqvhdtd
Proceso(29/30) palabras --->gqvudpy
Proceso(30/30) palabras --->hueowu
***** PROCESO DE 20 PALABRAS *****
Proceso(1/20) palabras --->tcwk
Proceso(2/20) palabras --->ojj
Proceso(3/20) palabras --->tam
Proceso(4/20) palabras --->vxfi
Proceso(5/20) palabras --->ftdwt
Proceso(6/20) palabras --->nkp
Proceso(7/20) palabras --->do
Proceso(8/20) palabras --->lecc
Proceso(9/20) palabras --->wvgsva
Proceso(10/20) palabras --->gp
Proceso(11/20) palabras --->nxnags
Proceso(12/20) palabras --->utlqc
Proceso(13/20) palabras --->ps
Proceso(14/20) palabras --->qjsc
Proceso(15/20) palabras --->wavywgvf
Proceso(16/20) palabras --->qrhbwxfy
Proceso(17/20) palabras --->hkxk
Proceso(18/20) palabras --->yk
Proceso(19/20) palabras --->a
Proceso(20/20) palabras --->rvfqmksb
```



Línea 560, columna 37 100% UNIX (LF) UTF-8

Ilustración 16 Comprobación del total de líneas 550 palabras + 10 títulos

Ilustración 1 Ordenar números.....	2
Ilustración 2 Código Aleatorio.....	3
Ilustración 3 salida de 40 números aleatorios.	3
Ilustración 4 cambiamos la dirección por cmd.	4
Ilustración 5 Consola del sistema.....	4
Ilustración 6 Salida de la tubería Aleatorio Ordenar.....	4
Ilustración 7 Código lenguaje.class	5
Ilustración 8 Código de colaborar.class	6
Ilustración 9 generación de ascii aleatorio entre a-z	7
Ilustración 10 tabla ascii	7
Ilustración 11 Generación de titulo	8
Ilustración 12 java -jar lenguaje.jar 10 archivo10.txt.....	8
Ilustración 13 contenido del archivo	8
Ilustración 14 Ejecución java -jar Colaborar.jar	8
Ilustración 15 contenido de archivo.txt	9
Ilustración 16 Comprobación del total de líneas 550 palabras + 10 títulos.....	9