Tabla de contenido

Diccionario / Cifrador [Documentación]

Hemos desarrollado este proyecto con sumo cuidado y dedicación, esforzándonos por desarrollar al máximo nuestros conocimientos hasta el momento. Tal vez aún hay cientos de errores que faltan por corregir, pero hemos dedicado un porcentaje altamente significativo en la tarea de buscarlos y arreglarlos.
Esperamos que si te animas a revisar toda esta documentación también te animes a tomar el proyecto, probarlo, analizarlo, descargarlo y hasta editarlo.

Elementos necesarios

Para poder editar el proyecto (y "testear") es necesario tener instalada la última versión de NetBeans y JDK.

Código Fuente

El código fuente del proyecto se puede encontrar en **Github**. *Si quieres, puedes probar y mejorar nuestra versión.*

Créditos

Este proyecto fue desarrollado por:

- Wilson Tovar (@wilsontov)
- Dolcey Mendoza (@dolceymendozajr)

Así mismo agradecemos a William García (@wgarcia1309) por su ayuda en encontrar uno que otro bug en el programa.

Información General

Árbol de Archivos

El proyecto que usted abrirá contiene la siguiente estructura:

```
/
├──nbproject/
├──src/
|   ├──lab_final_dolcey_wilson/
|       ├──Ayuda.form
|       ├──Ayuda.java
|       ├──Binario.java
|       ├──Cifrado.form
|       ├──Cifrado.java
|       ├──Diccionario.form
|       ├──Diccionario.java
|       ├──EncriptarVigenere.java
|       ├──FuncionesPropias.java
|       ├──MetodoEncript.java
|       ├──ayuda.jpg
|       └──change.png
├──Laboratorio_Final _IST2089_201630.pdf
├──README.md
├──build.xml
└──manifest.mf
```

`Laboratorio_Final _IST2089_201630.pdf` es un archivo PDF con toda la información correspondiente al laboratorio final. *Recomendamos su lectura para mayor comprensión del proyecto*

Clases del Proyecto

Las clases se encuentran en la carpeta `lab_final_dolcey_wilson`, y son las siguientes:

- Ayuda
- Binario
- Cifrado
- Diccionario
- EncriptarVigenere
- FuncionesPropias
- MetodoEncript

Ayuda.java

La clase `Ayuda` es una clase `JFrame Form`. Esta contiene los nombres de los desarrolladores y un enlace hacia el proyecto en **Github**, donde se pueden encontrar los enlaces de la documentación.

Binario.java

La clase `Binario` convierte un `String` a binario.

Variables globales

Estos dos vectores son usados para la conversión de letras con tildes y la ñ. El vector `n_extendidas` contiene el código binario correspondiente a cada letra tildada y las ñ.

```
static int[] n_extendidas = {
        11100001, 11101001, 11101101,
        11110011, 11111010, 11000001,
        11001001, 11001101, 11010011,
        11011010, 11110001, 11010001};
```

```
static char[] extendidas = {
        'á', 'é', 'í',
        'ó', 'ú', 'Á',
        'É', 'Í', 'Ó',
        'Ú', 'ñ', 'Ñ'};
```

Funciones

String Convertir

String Descifrar

Convertir

Esta función retorna el valor binario del `String` ingresado. Es la única subrutinaa de la clase.

```
public static String Convertir(String palabra) {
        String text_convertido = "";
        String temp = "";

        for (int i = 0; i < palabra.length(); i++) {
            char letra = palabra.charAt(i);
            int ascii = (int) letra;
            temp = "";

            for (int j = 0; j < 8; j++) {
                temp += (ascii % 2) + "";
                ascii = ascii / 2;
            }

            for (int j = 7; j >=0; j--)
                text_convertido += temp.substring(j, j+1);
            text_convertido += " ";
        }

        return text_convertido;
    }
```

Esta linea de código selecciona la i-ésima letra de la palaba: `char letra = palabra.charAt(i);` Y luego, con la ayuda de `(int)`, tomamos el código `ascii` de la i-ésima letra: `int ascii = (int) letra;`

Conversión a base 2

Convertir un número a base 2 es muy fácil. Se realizan divisiones enteras hasta que el cociente sea menor que la base (2). Esta operación la realizamos en el `for` anidado:

```
for (int j = 0; j < 8; j++) {
  temp += (ascii % 2) + "";
  ascii = ascii / 2;
}
```

El límite del `for` es 8 porque todo número de base 10 menor que 255 al ser convertido a base 10 será un numero de 8 dígitos (octeto).

Primero tomamos el valor numérico de la i-ésima letra que anteriormente habíamos obtenido, aplicamos mod 2 y lo convertimos a cadena de caracteres: `temp += (ascii % 2) + "";` Así mismo debemos ir dividiendo el `ascii` entre la base.

Descifrar

Esta función permite tomar una cadena conformada por números binarios y obtener las letras a la que corresponden, usando el código ascii de la misma.

Para convertir un número binario base diez, se deben hacer una serie de operaciones matemáticas simples.

Ejemplo: `número decimal: 10001101`
Se toma cada el número de la i-ésima posición y se multiplica por la base (2) elevada a la `i` y se suman estas multiplicaciones. Sería así:
`1 * (2^7) + 0 * (2^6) + 0 * (2^5) + 0 * (2^4) + 1 * (2^3) + 1 * (2^2) + 0 * (2^1) + 1 * (2^0) = 128 + 0 + 0 + 0 + 8 + 4 + 0 + 1 = 133`

Teniendo esto claro podemos entender facilmente la función.

```
public static String Descifrar(String palabra) {
        String[] frase = palabra.split(" ");
        String pal_final = "";
        boolean sw;
        String temp = "";

        for (int i = 0; i < frase.length; i++) {
            sw = true;
            int j = 0;
            while (sw && j < extendidas.length) {
                if (Integer.parseInt(frase[i]) == n_extendidas[j
]) {
                    sw = false;
                    temp = extendidas[j] + "";
                } j++;
            }

            if (sw) {
                int sum = 0;
                int n = 7;
                for (int k = 0; k < frase[i].length(); k++) {
                   sum += (Integer.parseInt(frase[i].substring(k
, k + 1)) * (Math.pow(2, n)));
                   n--;
                }
                pal_final += (char) sum;
            } else
                pal_final += temp;
        }

        return pal_final;
    }
```

Usamos un ciclo `for` para recorrer toda la palabra, letra a letra.

Primero verificamos si la letra de la i-ésima posición es una vocal tildada o es la `ñ`.

```
while (sw && j < extendidas.length) {
                if (Integer.parseInt(frase[i]) == n_extendidas[j
]) {
                    sw = false;
                    temp = extendidas[j] + "";
                } j++;
}
```

Cifrado.java

Esta es un clase `JFrame Panel` , por medio de la cual se muestra cada una de las palabras que fueron traducidas en tres métodos de cifrado siguiendo el siguiente esquema:

```
Propio
   └──Vigenère
     └──Binario
```

Variables de la clase

static String[] **palabras**;
static String **pass**;
static int **i** = 0; // Índice que controla la posición de los vectores a mostrar
static String[] **binarios**;

Funciones & Subrutinas

PasarDatos

btn_SiguienteActionPerformed

PasarDatos

Esta subrutina pasa dos datos específicos de la clase `Diccionario` a la clase `Cifrado:` clave y el vector de palabras traducidas.

```
public static void PasarDatos(String[] pals, String clave) {
  i = 0;
  palabras = pals;
  pass = clave;
}
```

btn_SiguienteActionPerformed

Esta subrutina es un evento. Se ejecuta cuando se presiona el boton `Sig. Palabra` .

```
private void btn_SiguienteActionPerformed(java.awt.event.ActionE
vent evt) {
        if (i == 0)
            txt_Clave.setText("Clave: " + pass);
        if (i < palabras.length) {
            btn_Siguiente.setText("Sig. Palabra");

            txt_Palabra.setText(palabras[i]);
            txt_Propio.setText(MetodoEncript.encrip(palabras[i])
);
            txt_Propio_Vigenere.setText(EncriptarVigenere.Encrip
t(txt_Propio.getText(), pass));
            txt_Propio_Vigenere_BInario.setText(Binario.Converti
r(txt_Propio_Vigenere.getText()));
        } else {
            JOptionPane.showMessageDialog(this, "No hay más pala
bras por mostrar", "ERROR", JOptionPane.ERROR_MESSAGE);
            btn_Siguiente.setEnabled(false);
        }
        i++;
    }
```

Diccionario.java

Clase principal del proyecto. *Aquí transcurre la marte de la magia.*

Funciones & Subrutina

EncriptarVigenere.java

Variable globales

```
public static char dicc[] = new char[32];
public static char pal1[] = new char[100];
public static char pal2[] = new char[100];
public static char clave[] = new char[100];
```

Funciones & Subrutinas

Esta clase contiene dos funciones: `Encript` & `Descencript` (String), y una subrutina, `GenerarVect` .

String Encript

GenerarVect

```
static void GenerarVect(String texto, String pass) {
        int letra = (int) 'a';
        for (int i = 0; i < 27; i++) {
            if (i < 14)
                dicc[i] = (char) (letra);
            else if (i == 14)
                dicc[i] = 'ñ';
            else if (i > 14)
                dicc[i] = (char) (letra - 1);
            letra++;
        }

        dicc[27] = 'á';
        dicc[28] = 'é';
        dicc[29] = 'í';
        dicc[30] = 'ó';
        dicc[31] = 'ú';

        int pos = 0, j;
        for (j = 0; j < pass.length(); j++)
            clave[j] = pass.charAt(j);
        int pos_fin = j;

        for (int i = 0; i < texto.length(); i++) {
            char temp = texto.charAt(i);
            if (pos == pos_fin)
                pos = 0;

            pal1[i] = temp;
            pal2[i] = clave[pos];
            pos++;
        }
    }
```

String Descencript

Función que descifra un `String`, usando la clave.

```
static String Desencript(String texto, String pass) {
        String desencriptado = "";
        GenerarVect(texto, pass);

        int xi = 0, ci = 0;

        for (int i = 0; i < texto.length(); i++) {
            int j = 0;
            boolean sw = true;

            do {
                if (Character.toString(dicc[j]).equalsIgnoreCase
(Character.toString(pal2[i]))) {
                    ci = j;
                    sw = false;
                } else
                    j++;
            } while (sw && j < 27);

            j = 0;
            sw = true;
            do {
                if (Character.toString(dicc[j]).equalsIgnoreCase
(Character.toString(pal1[i]))) {
                    xi = j;
                    sw = false;
                } else
                    j++;
            } while (sw && j < 27);

            if (xi - ci >= 0) {
                int a = (xi - ci) % 27;
                desencriptado = desencriptado + dicc[a];
            } else if (xi - ci < 0) {
```

```
                int b = (xi - ci + 27) % 27;
                desencriptado = desencriptado + dicc[b];
            }
        }

        return desencriptado;
}
```

FuncionesPropias.java

Hemos sustituido un método de Java llamado `charAt()`, así mismo hemos creado una función que cuenta un carácter dentro de una cadena.

Funciones

char CharAtt

int ContarElemEpeci

CharAtt

```
public static char CharAtt(String word, int i) {
        if (i > -1 && i <= word.length() - 1) {
            char[] chara = word.toCharArray();
            return chara[i];
        } else
            return '?';
}
```

La posición del carácter que el usuario ingresó es válida si ésta es mayor que -1 y no excede el tamaño de la cadena. Si esto se cumple se genera un vector, `char[] chara`, con los carácteres de la cadena `word`. Luego retorna el carácter de la i-ésima posición. Si la condición inicial no se cumple, se retorna `?`, indicando que algo salió mal.

ContarElemEpeci

Esta función es sumamente fácil de entender. Simplemente toma una cadena (`word`), y busca un caracter (`elem`) en la cadena.

```
public static int ContarElemEpeci(String word, String elem) {
        int ve = 0;
        for (int i = 0; i < word.length(); i++) {
            if (word.substring(i, i +1).equals(elem))
                ve++;
        }
        return ve;
}
```

MetodoEncript.java

Esta clase es la encargada de cifrar y descifrar mediante un mecanismo que es muy sencillo y que consta de un diccionario de letras.

Mecanismo de cifrado

El mecanismo de cifrado que usamos es muy sencillo. Usa dos vectores: uno que llamaremos diccionario y el otro que contiene todas las letras del alfabeto español (incluyendo las vocales tildadas).

```
// DICCIONARIO
static char abc[] =
       {'n', 'H', 'D', 'a', 'e',
        'Q', 'F', 'b', 'w', 's',
        'v', 'O', 'y', 'R', 'S',
        'r', 'u', 'Z', 'o', 'ñ',
        'x', 'N', 'h', 'U', 'J',
        'W', 'i', 'p', 'I', 'c',
        'B', 'f'};
```

El segundo vector es generado con la subrutina GenerarVect.

Funciones & Subrutinas

GenerarVect

String desencrip

GenerarVect

Usamos esta función para generar un vector con todas las letras del alfabeto español, incluyendo las vocales tildadas, y otro vector con el número de la posición de cada una.

```
static void GenerarVect() {
        int letra = (int) 'a';
        for (int i = 0; i < 27; i++) {
            if (i < 14) {
                dicc[i] = (char) (letra);
            } else if (i == 14) {
                dicc[i] = 'ñ';
            } else if (i > 14) {
                dicc[i] = (char) (letra - 1);
            }
            letra++;
        }

        dicc[27] = 'á';
        dicc[28] = 'é';
        dicc[29] = 'í';
        dicc[30] = 'ó';
        dicc[31] = 'ú';

        for (int i = 0; i < num.length; i++)
            num[i] = (int) abc[i];
}
```

---

dicc[]

Inicializamos la variable `letra` con el codigo ascii de la primera letra, la a: `int letra = (int) 'a'`

Luego, generamos las demás letras, hasta la `z`.

```
for (int i = 0; i < 27; i++) {
     if (i < 14)
          dicc[i] = (char) (letra);
     else if (i == 14)
          dicc[i] = 'ñ';
     else if (i > 14)
          dicc[i] = (char) (letra - 1);
     letra++;
}
```

Tenemos un caso especial, la `ñ` :

```
else if (i == 14)
          dicc[i] = 'ñ';
```

Agregamos las vocales tildadas:

```
dicc[27] = 'á';
dicc[28] = 'é';
dicc[29] = 'í';
dicc[30] = 'ó';
dicc[31] = 'ú';
```

num[]

En el mismo ciclo generamos un segundo vector con las posiciones de cada letra, anteriormente generada.

```
for (int i = 0; i < num.length; i++)
     num[i] = (int) abc[i];
```

desencrip

Como su nombre lo indica, esta función descifra un texto cifrado (`String encriptada`) mediante nuestro método. Es muy sencillo entenderlo.

```
public static String desencrip(String encriptada) {
        GenerarVect();
        String desen = "";
        int ch = 0;

        for (int i = 0; i < encriptada.length(); i++) {
            for (int j = 0; j < num.length; j++) {
                if (num[j] == encriptada.charAt(i)) {
                    ch = dicc[j];
                }
            }
            desen += (char) ch;
        }

        return desen;
}
```

Primero, genera los vectores `dicc` y `num` : `GenerarVect();`

encrip