Manual de usuario

Introducción

Este programa ha sido desarrollado con el fin de permitirles a los usuarios una visualización gráfica de la relación que existe entre varios tweets que tratan sobre un mismo tema.

Dicha relación se basa en pedirles a los usuarios una frase o palabra de su elección y posteriormente buscar todos los tweets relacionados con ese tema en particular. Una vez que se tienen todos los tweets, estos se proyectan de manera gráfica, mostrando no sólo la relación, sino también si el tweet es positivo, es decir involucra sentimientos/palabras positivas, negativo, es decir involucra sentimientos/palabras negativas, o si es neutro.

Así mismo, se cuenta con cuatro diccionarios de palabras positivas y de palabras negativas, dos en español y dos en inglés, los cuales el usuario puede editar mediante el programa.

Índice

Conceptos

Twitter es una red social que permite a los usuarios escribir pequeñas publicaciones llamadas tweets. Los usuarios de Twitter pueden compartir tweets y seguir los tweets de otros usuarios a través de diferentes dispositivos y plataformas.

Un *tweet* es una pequeña publicación de no más de 140 caracteres, aunque estos pueden llegar a variar dependiendo de Twitter, que le permite al usuario compartir no sólo una pequeña frase o texto, sino también le deja compartir fotos o preguntas. Además el usuario tiene la opción de compartir su ubicación, esto a su elección.

Grafo, en ciencias de la computación, es un conjunto de objetos enlazados entre sí. Cada objeto es llamado vértice o nodo, y a la unión que existe entre cada uno se le llama arista o arco. Los grafos permiten las relaciones que existen entre varios objetos que interactúan entre sí.

Subgrafo es un grafo que es parte de un grafo más grande. Como su nombre lo indica, es un grafo que sale a partir de otro grafo, perteneciendo siempre todo el subgrafo a ese otro grafo más grande. Aunque no necesariamente tiene que ser más pequeño, ya que el subgrafo puede llegar a abarcar a todo el grafo.

Graph Stream es una librería utilizada en el programa desarrollado la cual permite la visualización dinámica de grafos.

Utilización

Una vez abierto el programa, este tiene dos funciones principales más una función para cerrar el programa.

La primera función se llama *Ver Tweets*. Si se le da click a esta función, esta le pregunta al usuario acerca de un tema que desee buscar. Se puede ingresar una palabra o varias, y al hacer click aceptar o presionar la tecla de Enter, el programa buscará todos los tweets que se relacionen con las palabras ingresadas.

Una vez que ha buscado los tweets, éstos se van a desplegar en una pantalla llamada *Resultados de búsqueda*. Esta pantalla tiene seis opciones, se tiene la opción de *Regresar*, la cual te envía de vuelta al menú principal, y la opción de *Ver Grafos*, la cual muestra de manera gráfica, a través de Graph Stream, la relación que existe entre los tweets (nodos) y muestra los sentimientos el tweets, es decir, si el tweet es alegre, si es negativo o si sólo es neutro. También está la opción de *Ver Subgrafo*, la cual permite al usuario elegir un tweet (nodo) y a partir de él ver todos los que se relacionan con el mismo.

Ver Grafo Triste, *Ver Grafo Feliz* y *Ver Grafo Neutro*, le permiten ver al usuario los subgrafos todos los nodos que tienen que ver con sentimientos negativos, sentimientos positivos o que son neutros, respectivamente.

Se logra saber cuáles son los sentimientos contenidos en el tweet a través de los diccionarios que posee el programa, ya que gracias a ellos, se van comparando las palabras que contiene el tweet contra las palabras que tienen nuestros diccionarios de

palabras positivas y negativas. Por lo cual se deduce si el tweet es positivo, negativo o neutro.

La segunda función del menú principal se llama *Diccionarios*. Si se le da click a esta función se muestra una pantalla llamada Diccionario la cual del lado izquierdo tiene un menú para seleccionar los diferentes diccionarios.

Primero se selecciona el idioma, ya sea inglés o español. Una vez elegido el idioma, se selecciona Palaras Positivas o Palabras Negativas. Esto se hace con el fin de elegir entre uno de los cuatro tipos de diccionarios que se tienen.

Para cada uno de estos diccionarios hay tres funciones: Mostrar, Agregar y Borrar. En el caso de agregar y borrar, se debe introducir la palabra deseada por el usuario en el recuadro junto al botón Mostrar.

La función de *Mostrar* le muestra al usuario la lista completa de palabras que se tienen en el diccionario.

Por otro lado, *Agregar* permite añadir nuevas palabras que no contengan los diccionarios, hay que tener en cuenta que cualquier palabra que aparezca en los diccionarios de palabras positivas se tomará como positiva, y al comparar esas palabras con los tweets, se seguirán tomando como positivas; lo mismo para los diccionarios de palabras negativas.

Finalmente, *Borrar* permite al usuario borrar del diccionario seleccionado

El botón Regresar, al igual que en Ver Tweets, permite al usuario volver al menú principal.

El botón *Cerrar* que aparece al final del menú principal, al presionarlo es el que se encarga de cerrar el programa.

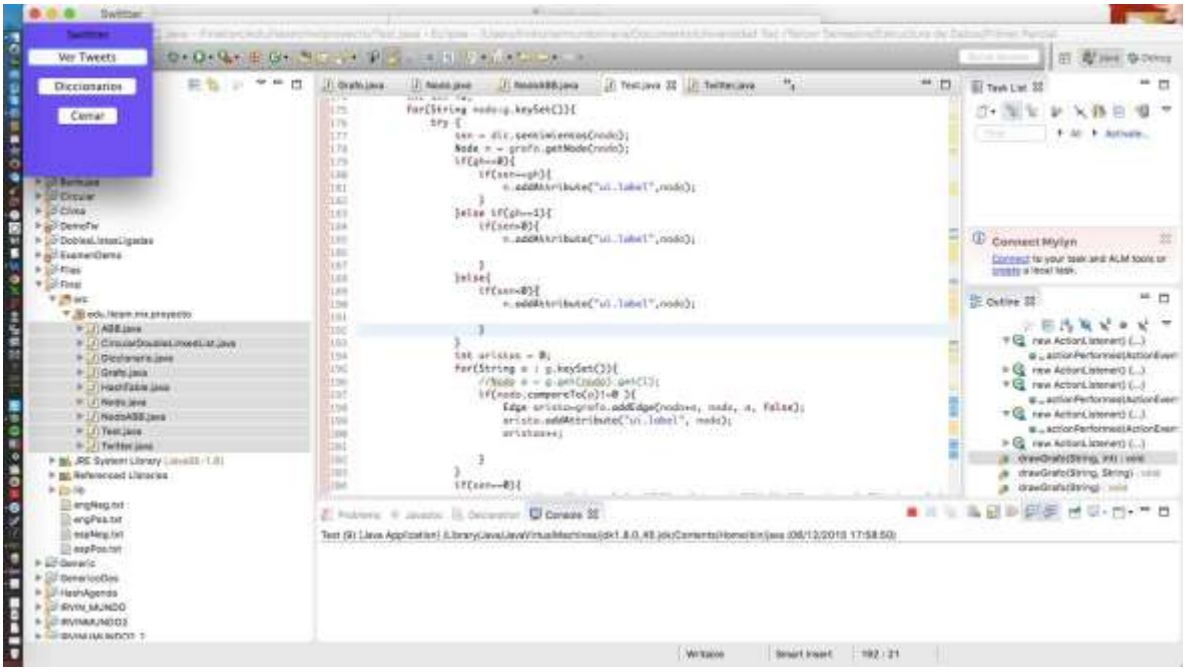
Pruebas

Primero corremos el programa. En el caso del usuario, se inicia con el .jar.

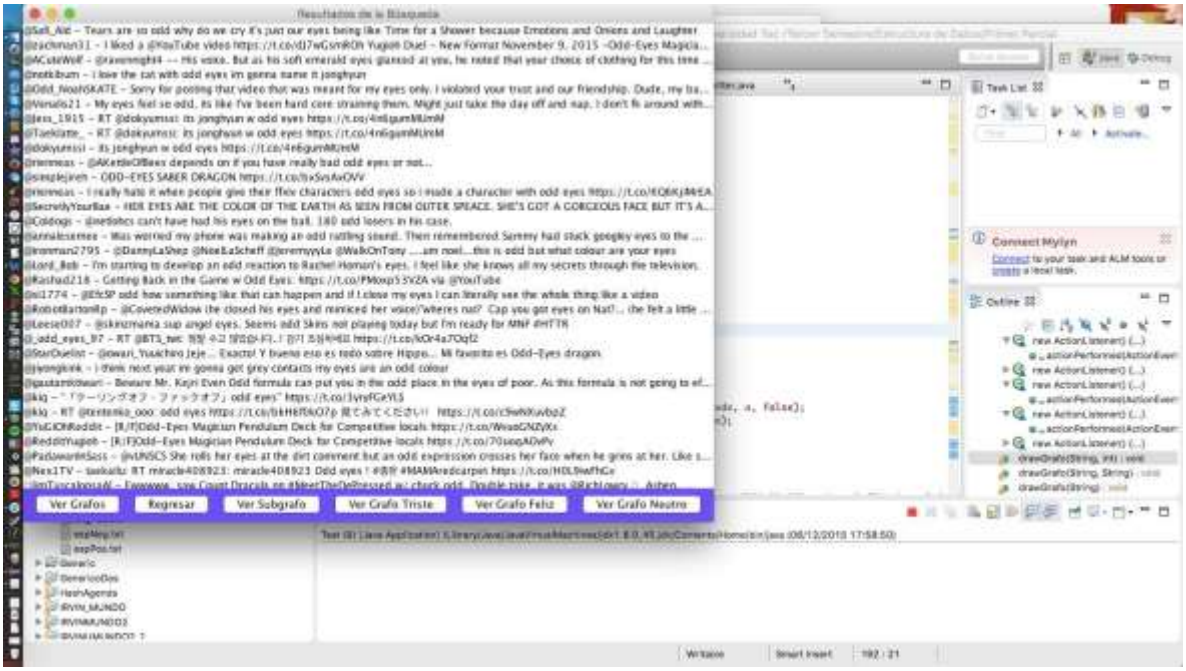
En la esquina superior izquierda se muestra el menú principal del programa.

Tiene sus tres funciones, ordenadas de arriba hacia abajo, Ver Tweets, Diccionarios y Cerrar.

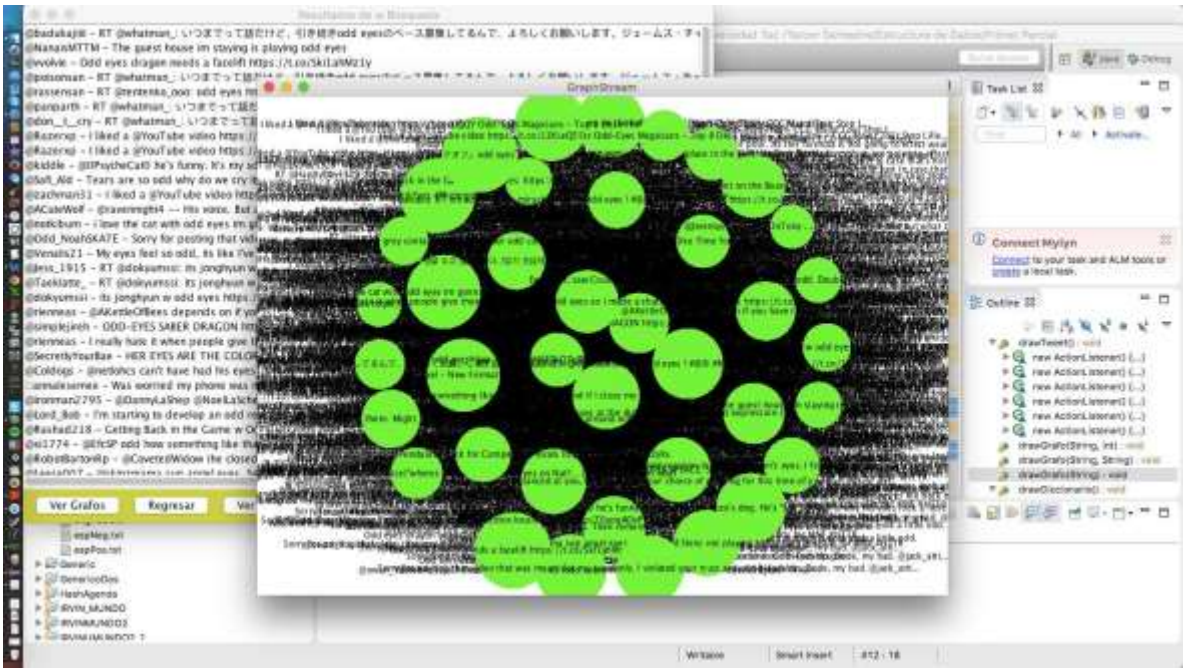
El color de fondo del menú principal es aleatorio, por lo cual, cada vez que se corra el programa, el color será diferente.



Al darle click a Ver Tweets, este le pregunta al usuario qué desea buscar, para posteriormente mostrar todos los tweets que se relacionan con lo que se ingresó, como se muestra a continuación.

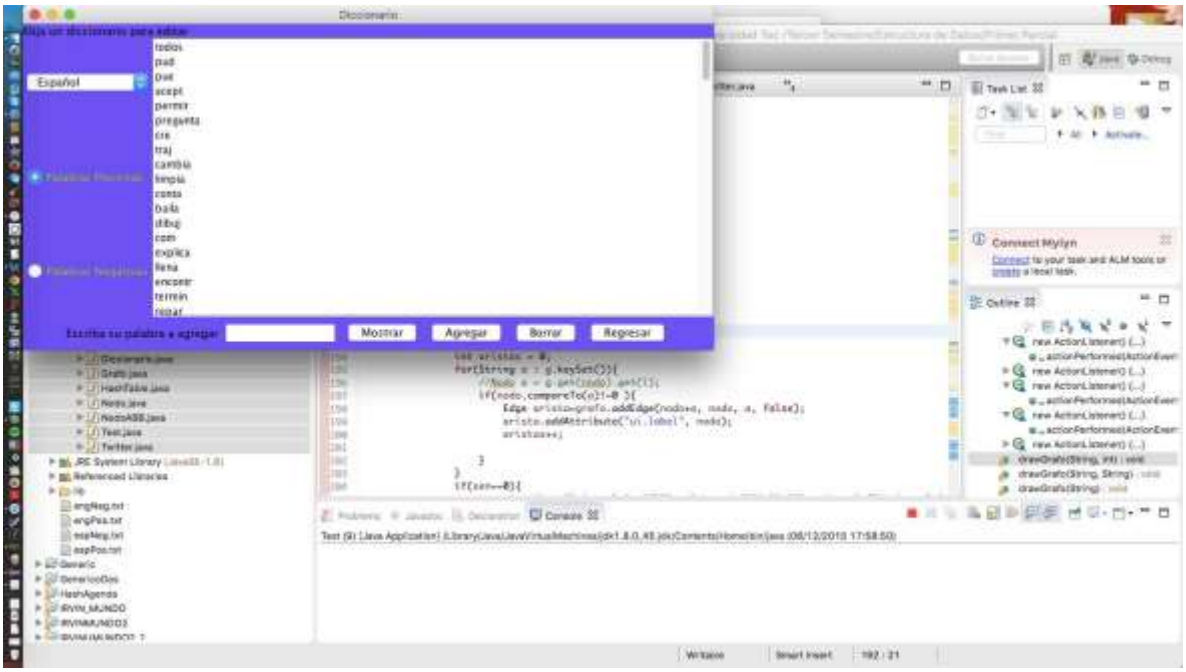


En la pantalla de Resultados de Búsqueda existe la opción de Ver Grafos, Ver Subgrafo, Ver Grafo Triste, Ver Grafo Feliz y Ver Grafo Neutro. Cualquiera de estos botones muestra un grafo en el cual están ya sea todos o parte de los tweets mostrados, aunque cada uno tiene su propio grafo ya que no se le está pidiendo enseñar lo mismo. Un ejemplo de un grafo es el siguiente:



Si se le da click al botón Diccionarios del menú principal se despliega la pantalla Diccionarios en donde se pueden agregar palabras a los diccionarios, se pueden borrar las palabras o simplemente desplegar el listado de palabras que contiene cada diccionario.

Aquí se muestra como se despliega el diccionario en español de palabras positivas.



Algoritmos

Grafo

```java
package edu.itesm.mx.proyecto;

import java.util.*;

public class Grafo extends HashMap<String, CircularDoubleLinkedList<String>>{
    boolean[] rec;
    CircularDoubleLinkedList<Nodo> pila;
    int costo;

    public Grafo(){
        super();
    }

    public boolean BFS(Grafo g, CircularDoubleLinkedList<Nodo> c) {
        boolean recorrido = false;
        //linked list de test
        //Nuevo linked list utilizada como cola
        CircularDoubleLinkedList<Nodo> cola = new CircularDoubleLinkedList();
        rec = new boolean[c.size()];
        //recorrer c/nodo del grafo
        //mejor utilizar el tweet para agregar a sus nodos adyacentes
        //después se van a pedir los adyacentes de ese para recorrerlos
        int i=0;
        for(String nodo : g.keySet()){
            if(i==0){
                Nodo node = new Nodo(nodo);
                cola.addLast(node);
            }
            //No visitado
            //g.getNode(nodo).addAttribute("ui.style", "fill-color: rgb(128,128,128);size: "+(5*aristas)+";");
            rec[i]=false;
            i++;
        }
        //nodo 1 ya recorrido pero no se marca sino hasta el sig. for
        i=0;
        while(!cola.isEmpty()){
            // extraer el nodo i de la cola y explorar todos sus nodos adyacentes
            for(String nodo : g.keySet()){
```

```java
    public boolean DFS(Grafo g, CircularDoubleLinkedList<Nodo> c){
        boolean recorrido = false;
        pila = new CircularDoubleLinkedList();
        rec = new boolean[c.size()];
        int i=0;
        for(String nodo : g.keySet()){
            rec[i]=false;
            //g.getNode(nodo).addAttribute("ui.style", "fill-color: rgb(128,128,128);size: "+(5*aristas)+";");
            i++;
        }
        i=0;
        costo=0;
        //pila podría ser la sublista de los tweets adyacentes al que se envía
        for(String nodo : g.keySet()){
            Nodo node = new Nodo(nodo);
            pila.addLast(node);
            if(rec[i]==false){
                DFS(costo, i, pila);
            }
            i++;
        }
        recorrido=true;
        return recorrido;
    }
    private void DFS(int costo, int i, CircularDoubleLinkedList<Nodo> pila){
        costo+=1;
        //Agregar el costo a la arista

        //se va a checar para cada tweet vecino
        //cambiar for para que haga eso
        while(!pila.isEmpty()){
            //ver si nodos adyacentes al nodo enviado han sido recorridos
            if(rec[i]==false){
                //g.getNode(nodo).setAttribute("ui.style", "fill-color: rgb(255, 69, 0);size: "+(5*aristas)+";");
                rec[i]=true;
                pila.removeLast();
                DFS(costo, i, pila);
            }
```

Test

```
public static void main (String args[]){
    System.setProperty("org.graphstream.ui.renderer", "org.graphstream.ui.j2dviewer.J2DGraphRenderer");
    new Test();
}
public Test(){
    drawPrincipal();
}

public void drawTweet(){
    tweets=new JFrame();
    tweets.getContentPane().setBackground(color);
    tweets.getContentPane().setForeground(color2);
    tweets.setSize(800,600);
    tweets.setTitle("Resultados de la Búsqueda");
    tweets.setLayout(new BorderLayout());
    tweets.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    Twitter t=new Twitter();
    DefaultListModel l=new DefaultListModel();
    s=JOptionPane.showInputDialog(principal, "Escriba el tema que desea ver: ");
    l=t.consultaTwitter(s,l);
    lista = new JList(l);
    tweets.add(lista, BorderLayout.CENTER);
    bGrafos=new JButton("Ver Grafos");
    JPanel z=new JPanel();
    z.add(bGrafos);
    z.add(bregresar);
    tweets.add(z, BorderLayout.SOUTH);
    bregresar.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            tweets.dispose();
            drawPrincipal();
            principal.setVisible(true);
        }
    });
    bGrafos.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            drawGrafo(s);
        }
```

```
public void drawGrafo(String str){
    t = new Twitter();
    CircularDoubleLinkedList <String>c =  t.consultaTwitter(str);
    int tamano;
    grafos = new JFrame();
    grafos.setSize(800, 600);
    grafos.getContentPane().setBackground(color);
    grafos.getContentPane().setForeground(color2);
    Grafo g = new Grafo();
    tamano = c.size();
    System.out.println(c.size());
    for(int w = 0; w < c.size(); w++){
        g.put(c.get(w), c);
    }
    c = new CircularDoubleLinkedList();
    int comp = 0;
    String uno="", dos="";
    Graph grafo = new SingleGraph("Grafo de tweets");
    for(String nodo : g.keySet()){
        grafo.addNode(nodo);
        String[] n = nodo.split(" ");
        for(String x : g.keySet()){
            int aa = 0;
            String[] e = x.split(" ");
            for(String v : n){
                for(String w : e){
                    if(w.equals(v)){
                        aa++;
                    }
                }
            }
            if(comp<aa && x!=nodo){
                uno = nodo;
                dos = x;
                comp=aa;
            }
        }
    }
```

```
Diccionario dic = new Diccionario();
int sen =0, nodoMayor=0;
String tweetMayor=new String();
for(String nodo:g.keySet()){
    try {
        sen = dic.sentimientos(nodo);
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    Node n = grafo.getNode(nodo);
    n.addAttribute("ui.label",nodo);
    int aristas = 0;
    for(String a : g.keySet()){
        if(nodo.compareTo(a)!=0 ){
            try{
                Edge arista=grafo.addEdge(nodo+a, nodo, a, false);
                arista.addAttribute("ui.label", nodo);
                aristas++;
                if(sen==0){
                    n.setAttribute("ui.style", "fill-color: rgb(0,25
                }else if(sen<0){
                    n.setAttribute("ui.style", "fill-color: rgb(255,
                }else if(sen>0){
                    n.setAttribute("ui.style", "fill-color: rgb(0,0,
                }
                if(aristas>nodoMayor){
                    tweetMayor = nodo;
                    nodoMayor=aristas;
                }
            }catch(Exception ex){
            }
        }
        aristas=0;
    }
}
System.out.println(tweetMayor);
grafo.display();
```

Diccionario

```
package edu.itesm.mx.proyecto;
import java.io.BufferedReader;

public class Diccionario {
    public String [] archivos={"engNeg.txt","engPos.txt","espNeg.txt","espPos.txt"};

    public DefaultListModel mostrar(DefaultListModel m, int i){
        BufferedReader lector = null;
        try{
            lector = new BufferedReader(new FileReader(archivos[i]));
            while(lector.ready()){
                m.addElement(lector.readLine());
            }
        }catch(Exception ex){
            ex.printStackTrace();
        }
        return m;
    }

    public  void agregar(String palabra, int i) throws IOException{
        BufferedReader lector = new BufferedReader(new FileReader(archivos[i]));
        CircularDoubleLinkedList<String> pa = new CircularDoubleLinkedList();
        while(lector.ready()){
            pa.addLast(lector.readLine());
        }
        pa.addLast(palabra);
        File f = new File(archivos[i]);
        FileWriter bw = new FileWriter(f,false);
        PrintWriter out = new PrintWriter(bw);
        out.write(pa.toString());
        bw.close();
    }
public  void borrar(String palabra,  int i) throws IOException{
    BufferedReader lector = new BufferedReader(new FileReader(archivos[i]));
    CircularDoubleLinkedList<String> pa=new CircularDoubleLinkedList();
    while(lector.ready()){
        if(!lector.readLine().equals(palabra)){
            pa.addLast(lector.readLine());
        }
    }
    File f = new File(archivos[i]);
    FileWriter bw = new FileWriter(f,false);
    PrintWriter out = new PrintWriter(bw);
    out.write(pa.toString());
    bw.close();
}

public int sentimientos(String tweet) throws IOException{
    int felicidad=0, tristeza=0;
    for(int i=0;i<archivos.length;i++){
        try {
            BufferedReader lector = new BufferedReader(new FileReader(archivos[i]));
            String line = "";
            while(lector.ready()){
                if( line != null && tweet != null){
                    if(line.contains(tweet)){
                        if(i%2==0){
                            tristeza++;
                        }else{
                            felicidad++;
                        }
                    }
                }
                line = lector.readLine();
            }
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

Twitter

```java
package edu.itesm.mx.proyecto;
import javax.swing.*;

public class Twitter<T> extends JFrame{
    private JButton consulta;
    private twitter4j.Twitter twitter;
    public Twitter(){
        super("Twitter");
    }
    public CircularDoubleLinkedList consultaTwitter(String con){
        configuraTwitter();
        CircularDoubleLinkedList c = new CircularDoubleLinkedList();
        try {
            Query query = new Query(con);
            QueryResult result;
            int i=0;
            result = twitter.search(query);
            List<Status> tweets = result.getTweets();
            for (Status tweet : tweets) {
                c.addLast(tweet.getText());
            }
            i++;
        } catch (Exception te) {
            te.printStackTrace();
            System.out.println("Failed to search tweets: " + te.getMessage());
        }
        return c;
    }
public DefaultListModel consultaTwitter(String con, DefaultListModel m){
    configuraTwitter();
    try {
        Query query = new Query(con);
        QueryResult result;
        do{
            result = twitter.search(query);

            List<Status> tweets = result.getTweets();

            for (Status tweet : tweets) {
                String cambio="@" + tweet.getUser().getScreenName() + " - " + tweet.getText();
                m.addElement(cambio);
            }

        } while((query = result.nextQuery()) != null);
    }catch (Exception te) {
        te.printStackTrace();
        System.out.println("Failed to search tweets: " + te.getMessage());
    }
    return m;
}

public void configuraTwitter(){
    ConfigurationBuilder cb = new ConfigurationBuilder();
    cb.setDebugEnabled(true)
    .setOAuthConsumerKey("1p5P7csT4JVytA8PJVKQs0FrC")
    .setOAuthConsumerSecret("8S40bMHM7VMUwAVKzsI4FJ6oR6uAyI3DG1j5KUCauOViz4ohP9")
    .setOAuthAccessToken("432589386-hyjl6HqdwovriVhT3km9vB3VwDlD4gwglzQsO7V6")
    .setOAuthAccessTokenSecret("R7XbArvVyI4skEud6HLT52r4g9aXr3FABsZNeMMI6tSuC");
    TwitterFactory tf = new TwitterFactory(cb.build());
    twitter = tf.getInstance();
}
```