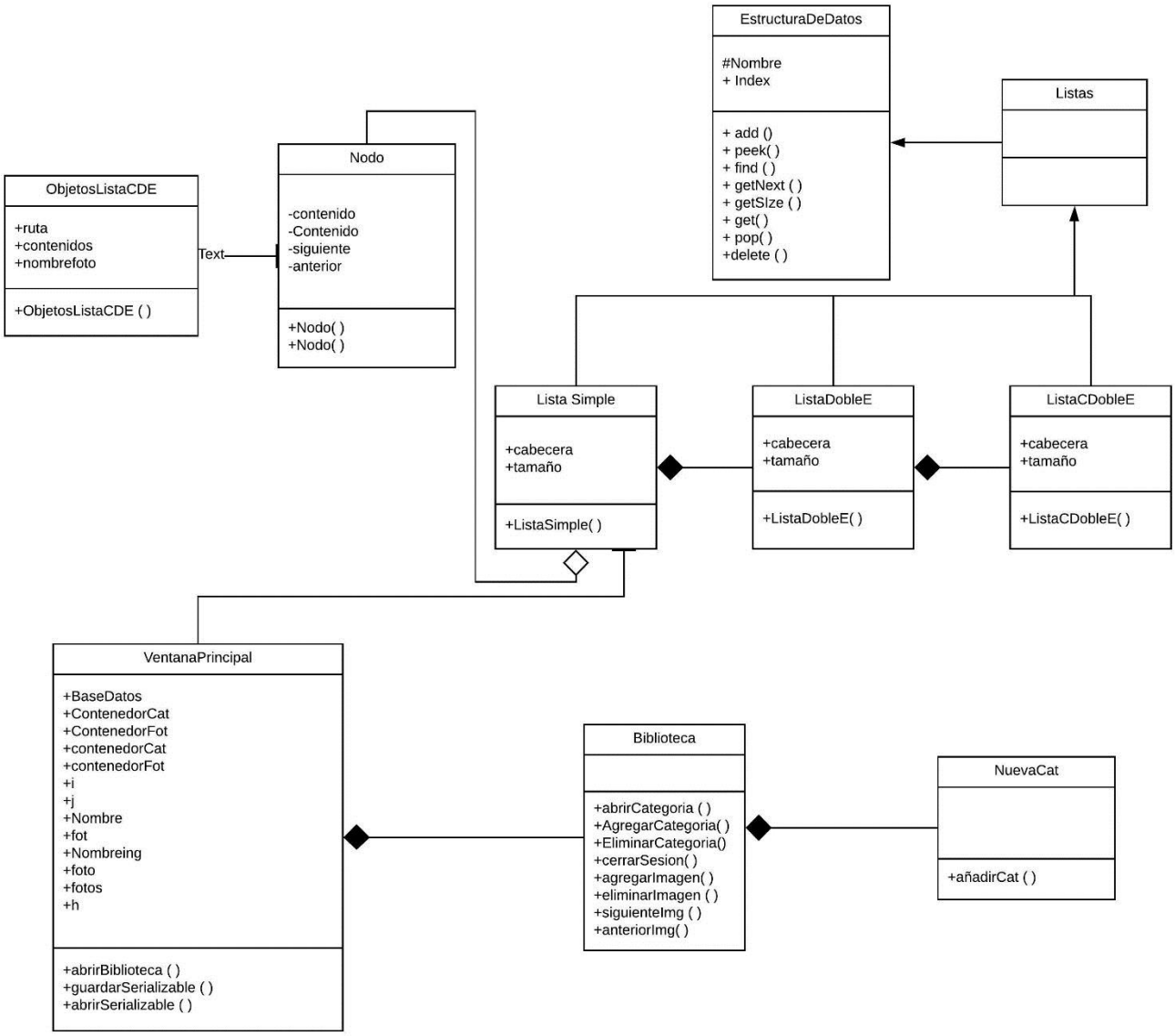
PARALLEL LIGHTROOM

MANUAL TÉCNICO

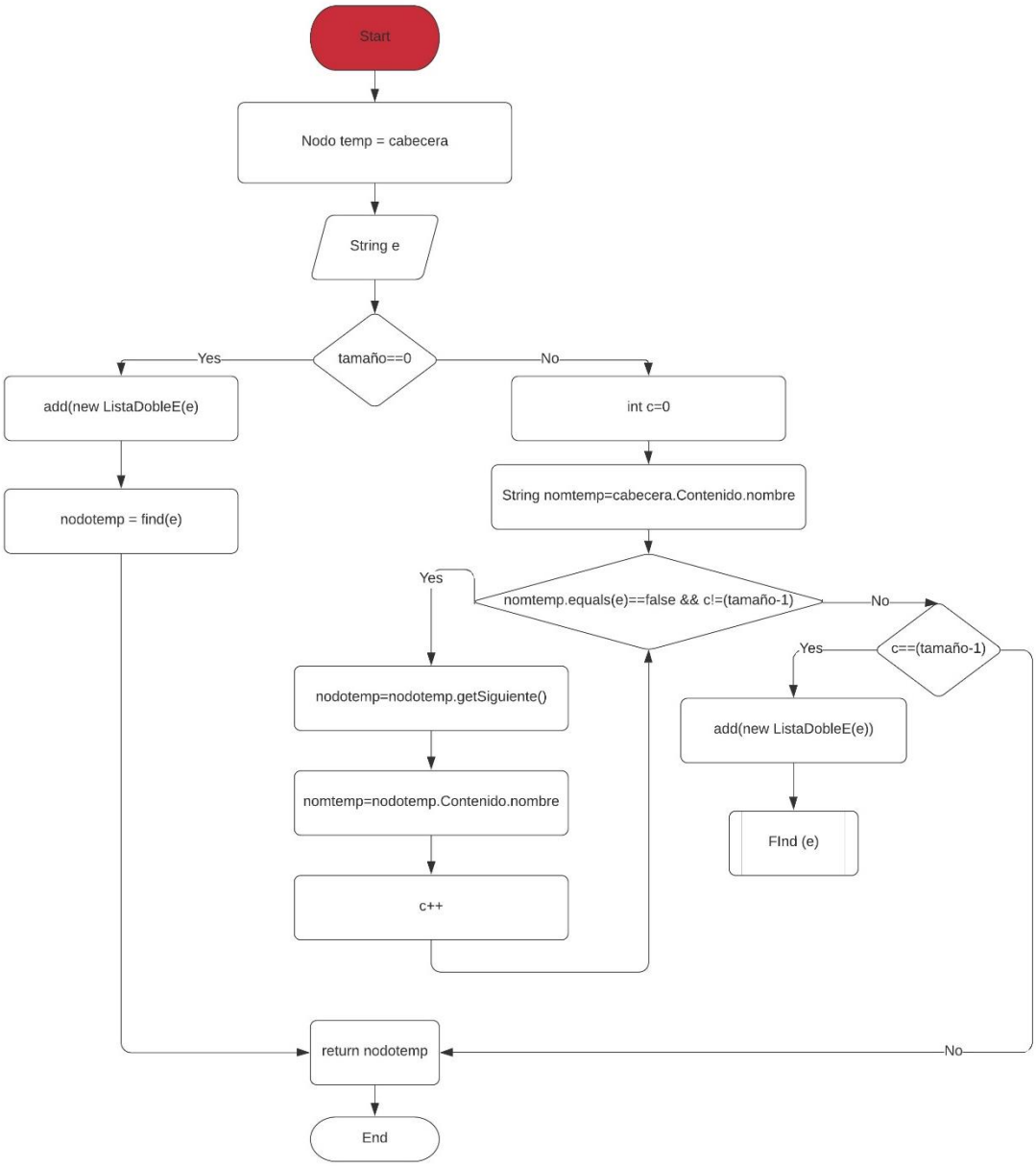
DIAGRAMA DE CLASES



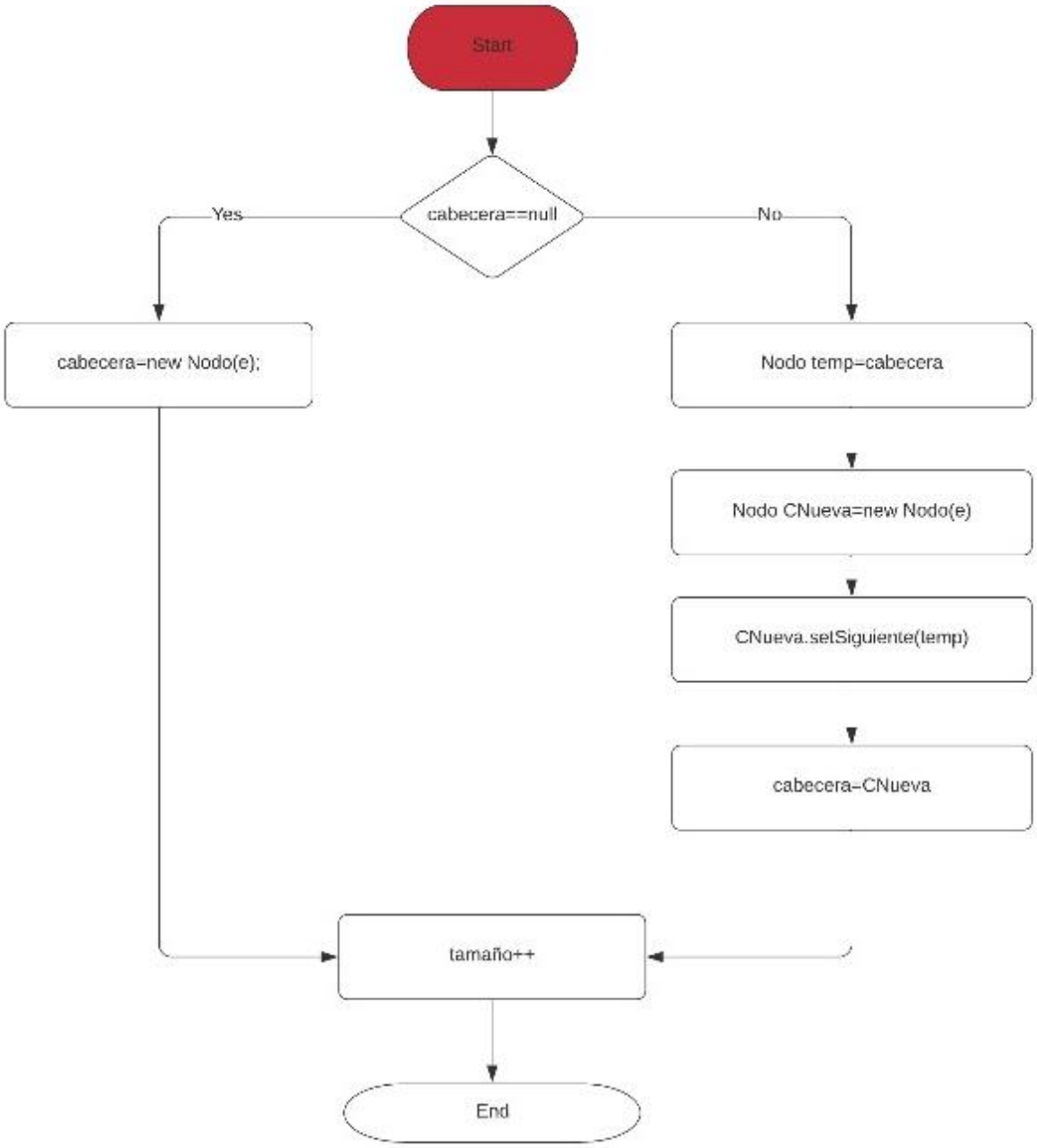
DIAGRAMAS DE FLUJO

Lista Simple

Metodo Find

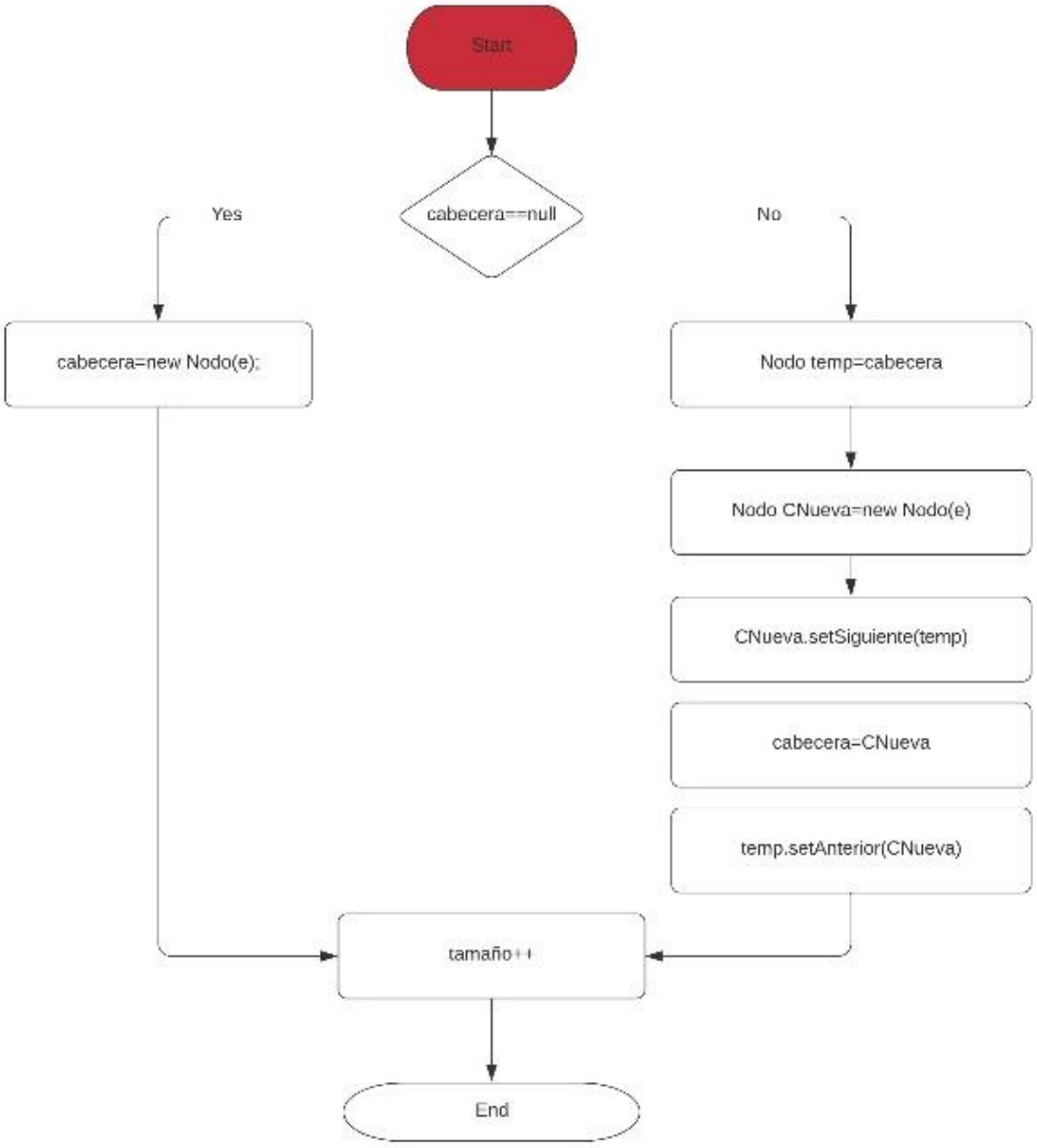


Metodo Add

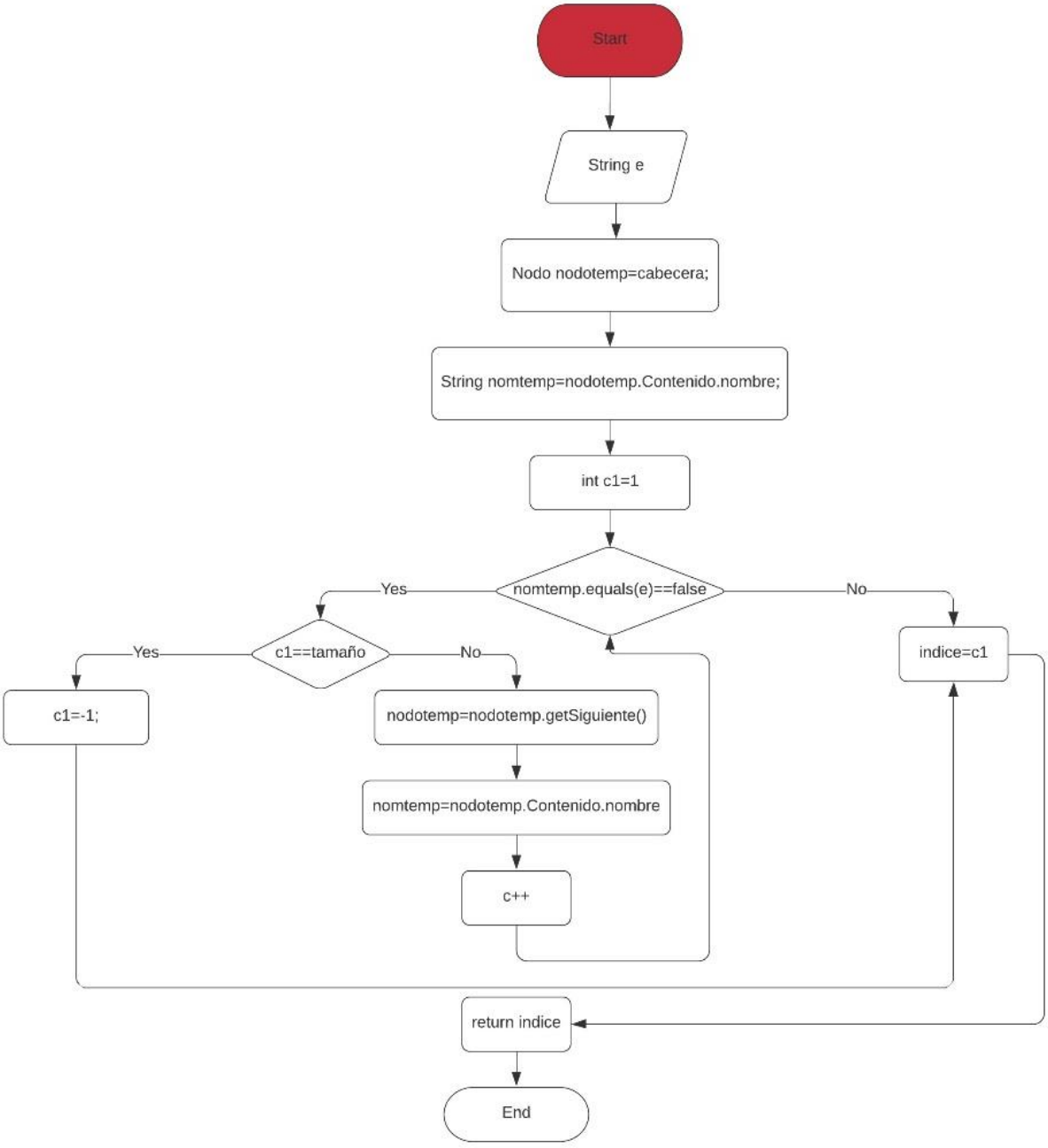

Start
cabecera==null
Yes
No
cabecera=new Nodo(e);
Nodo temp=cabecera
Nodo CNueva=new Nodo(e)
CNueva.setSiguiente(temp)
cabecera=CNueva
tamaño++
End

Lista Doblemente Enlazada

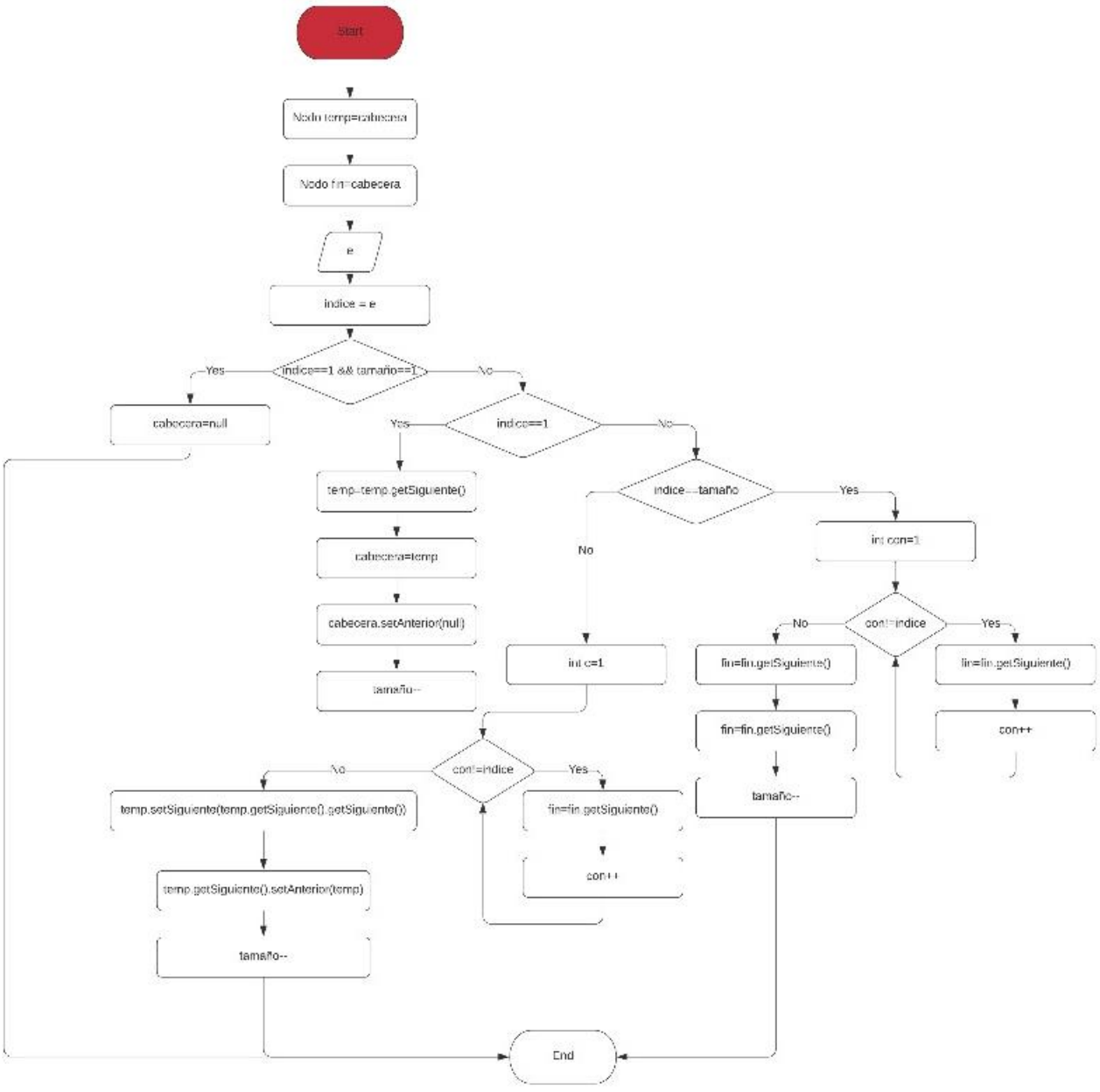
Metodo Add



Metodo find

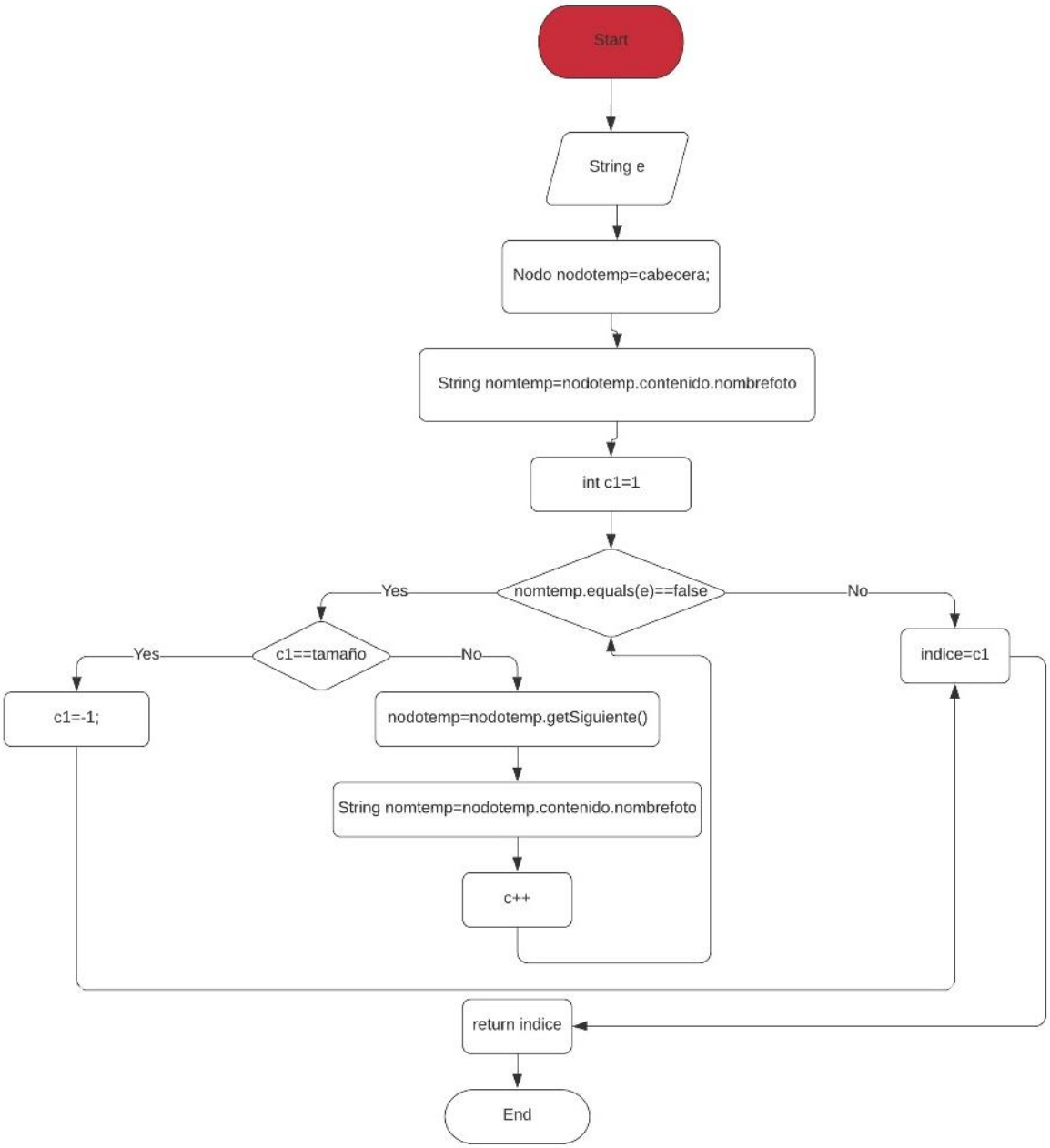

Start
String e
Nodo nodotemp=cabecera;
String nomtemp=nodotemp.Contenido.nombre;
int c1=1
nomtemp.equals(e)==false
Yes
No
c1==tamaño
Yes
No
indice=c1
c1=-1;
nodotemp=nodotemp.getSiguiente()
nomtemp=nodotemp.Contenido.nombre
c++
return indice
End

Metodo delete



Lista Circular Doblemente Enlazada

Metodo find



EXPLICACIÓN DE PROCESOS

Lista Simple

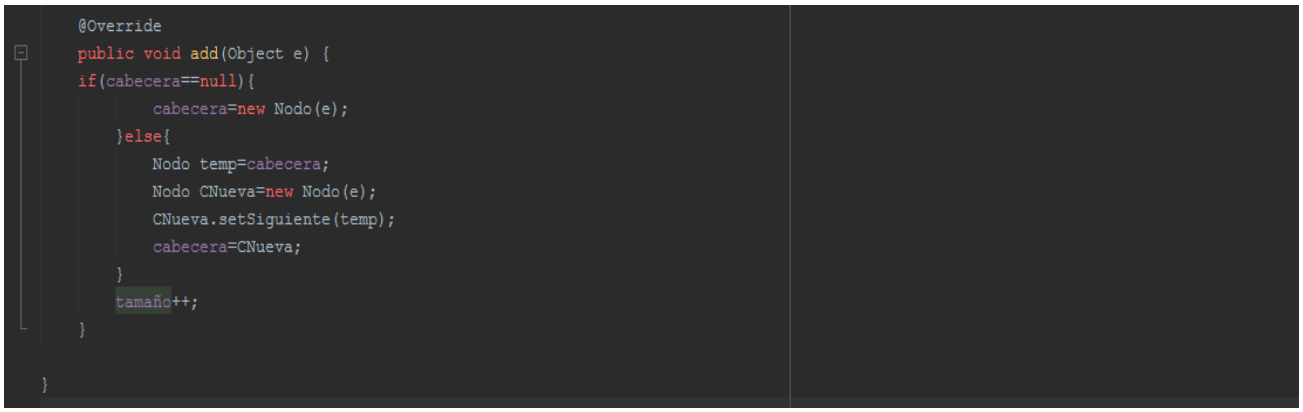
Metodo Find

El metodo se encarga de buscar un texto dentro de toda la lista. Para ello toma la cabecera de la lista y la guarda en un puntero. Si no hay nada en la lista (tamaño 0), el objeto toma la cadena del argumento y con eso crea una lista doble enlazada usando la cadena como nombre; ejecutando de nuevo el metodo buscar para regresar el nodo que contiene la lista. De no ser asi, toma el nombre de la cabecera y lo asigna a una cadena. Comienza un ciclo hasta que encuentre el nombre y si no lo encuentra, crea la lista con el argumento y ejecuta recursivamente el metodo. Cuando encuentre el nodo, de una u otra forma lo regresa.

```
//Para acceder a la lista contenida tomar este metodo y sacarla con el metodo getContenido del nodo
public Object find(Object e) {

    Nodo nodotemp=cabecera;

    if(tamaño==0){
        add(new ListaDobleE((String) e));
        nodotemp = (Nodo) find(e);
    }else{
        int c=0; //Contador
        String nomtemp=cabecera.Contenido.nombre;
        while(nomtemp.equals(e)==false && c!=(tamaño-1)){
            nodotemp=nodotemp.getSiguiente();
            nomtemp=nodotemp.Contenido.nombre;
            c++;
        }

        if(c==(tamaño-1)){
            add(new ListaDobleE((String) e));
            nodotemp=(Nodo) find(e);
        }

    }
    return nodotemp;

}
```

Método Add

El método add se encarga de agregar un nuevo nodo a la lista. Primero evalúa si la cabecera esta vacía para agregar el nodo. De no ser así, crea una variable temporal en donde guarda la cabecera. Crea un nuevo nodo con el objeto del argumento y le asigna el nodo siguiente el valor de temp. La cabecera pasa siendo esta nueva cabecera creada. Independientemente de lo que haya escogido, aumenta en uno el tamaño de la lista.



```
    @Override
    public void add(Object e) {
    if(cabecera==null){
            cabecera=new Nodo(e);
        }else{
            Nodo temp=cabecera;
            Nodo CNueva=new Nodo(e);
            CNueva.setSiguiente(temp);
            cabecera=CNueva;
        }
        tamaño++;
    }

}
```

Lista Doblemente Enlazada

Método Add

El método add se encarga de agregar un nuevo nodo a la lista. Primero evalúa si la cabecera está vacía para agregar el nodo. De no ser así, crea una variable temporal en donde guarda la cabecera. Crea un nuevo nodo con el objeto del argumento y le asigna el nodo siguiente el valor de temp. Así mismo, el método se encarga de enlazar este nodo temp con la cabecera. La cabecera pasa siendo esta nueva cabecera creada. Independientemente de lo que haya escogido, aumenta en uno el tamaño de la lista.

```
@Override
public void add(Object e) {
    if(cabecera==null){
        cabecera=new Nodo(e);
    }else{
        Nodo temp=cabecera;
        Nodo CNueva=new Nodo(e);
        CNueva.setSiguiente(temp);
        temp.setAnterior(CNueva);
        cabecera=CNueva;
    }
    tamaño++;
}
```

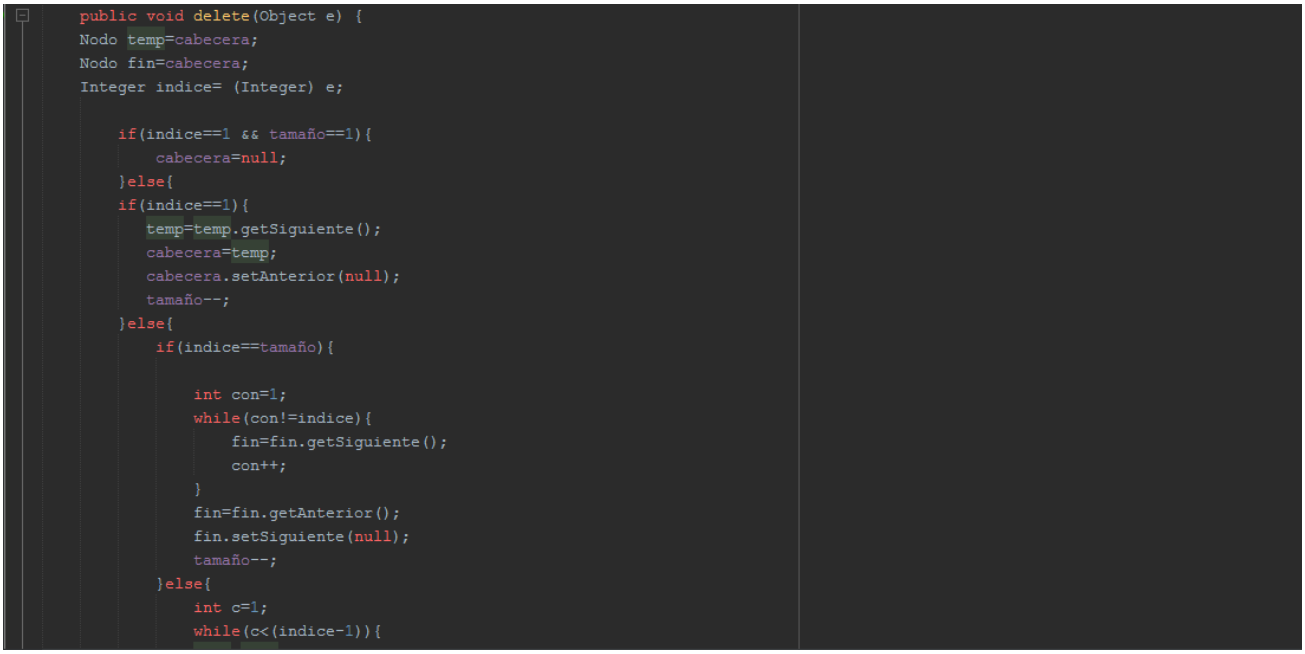
Método Find

El método find se encarga de encontrar un objeto con la cadena que se le pasa como argumento. Crea un apuntador con la cabecera de la lista y un string con el nombre de la cabecera. Comienza a realizar la búsqueda del nodo comparando el nombre de la lista con el nombre de la cabecera. Cuando lo encuentre, regresa la posición.

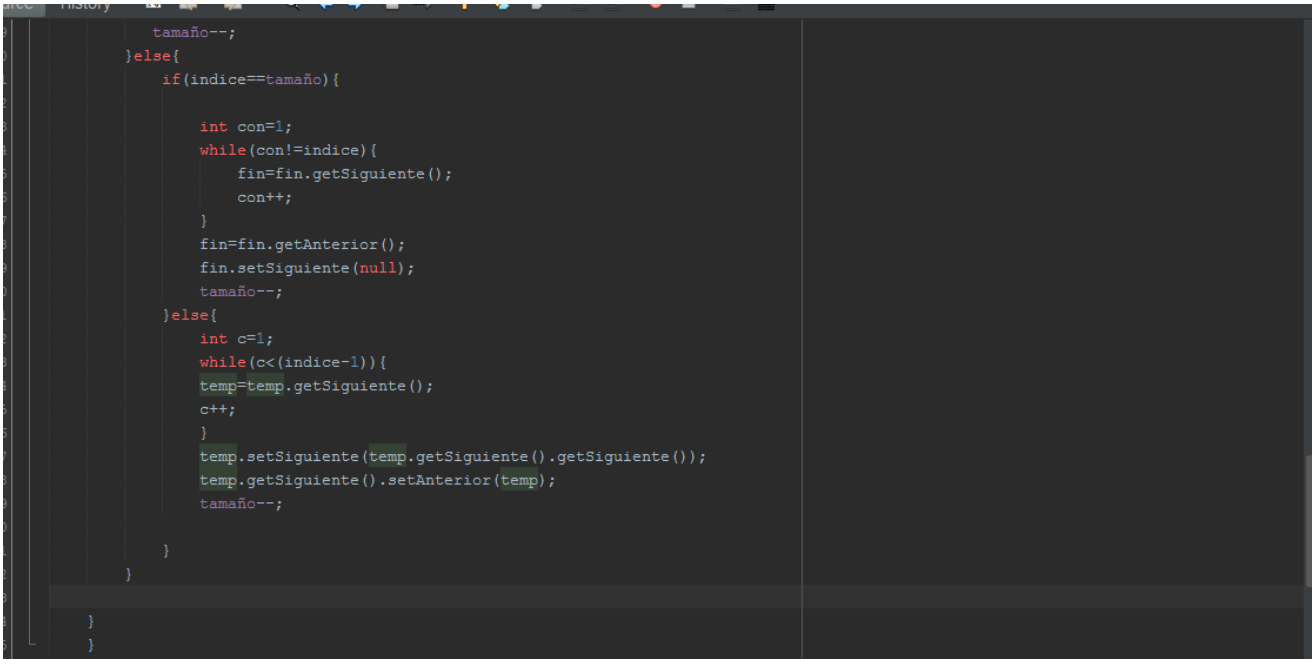
```
}

@Override
public Object find(Object e) {
    int indice;
    Nodo nodotemp=cabecera;
    String nomtemp=nodotemp.Contenido.nombre;
    int c1=1;
    while(nomtemp.equals((String)e)==false){
            if (c1==tamaño){
                c1=-1;
                break;
            }
            nodotemp=nodotemp.getSiguiente();
            nomtemp=nodotemp.Contenido.nombre;
            c1++;

        }
    indice=c1;
    return indice;
}
```

Método delete

El método delete se encarga de eliminar cualquier nodo de la lista. Primero verifica que el tamaño sea uno para vaciar la lista. Si el índice es uno, elimina la cabecera y lo conecta con el nodo siguiente. Si es igual al tamaño se encarga de borrar la ultima posición y si no es asi, llega al elemento anterior del objeto a borrar y lo conecta con el siguiente al que se quiere eliminar.



```
public void delete(Object e) {
Nodo temp=cabecera;
Nodo fin=cabecera;
Integer indice= (Integer) e;

    if(indice==1 && tamaño==1){
        cabecera=null;
    }else{
    if(indice==1){
       temp=temp.getSiguiente();
       cabecera=temp;
       cabecera.setAnterior(null);
       tamaño--;
    }else{
        if(indice==tamaño){

            int con=1;
            while(con!=indice){
                fin=fin.getSiguiente();
                con++;
            }
            fin=fin.getAnterior();
            fin.setSiguiente(null);
            tamaño--;
        }else{
            int c=1;
            while(c<(indice-1)){
```



```
            tamaño--;
        }else{
            if(indice==tamaño){

                int con=1;
                while(con!=indice){
                    fin=fin.getSiguiente();
                    con++;
                }
                fin=fin.getAnterior();
                fin.setSiguiente(null);
                tamaño--;
            }else{
                int c=1;
                while(c<(indice-1)){
                temp=temp.getSiguiente();
                c++;
                }
                temp.setSiguiente(temp.getSiguiente().getSiguiente());
                temp.getSiguiente().setAnterior(temp);
                tamaño--;

            }
        }

    }
    }
```

Lista Circular Doblemente Enlazada

Método Find

El método find se encarga de encontrar un objeto con la cadena que se le pasa como argumento. Crea un apuntador con la cabecera de la lista y un string con el nombre de la foto que hay en el objeto especial almacenado en la lista. Comienza a realizar la búsqueda del nodo comparando el nombre de la foto con el nombre de la cabecera. Cuando lo encuentre, regresa la posición.

```
@Override
public Object find(Object e) {
    int indice;
    Nodo nodotemp=cabecera;
    String nomtemp=nodotemp.contenido.nombrefoto;
    int c1=1;
    while(nomtemp.equals((String)e)==false){
            if (c1==tamaño){
                c1=-1;
                break;
            }
            nodotemp=nodotemp.getSiguiente();
            nomtemp=nodotemp.contenido.nombrefoto;
            c1++;

        }
    indice=c1;
    return indice;

}
```