UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICA

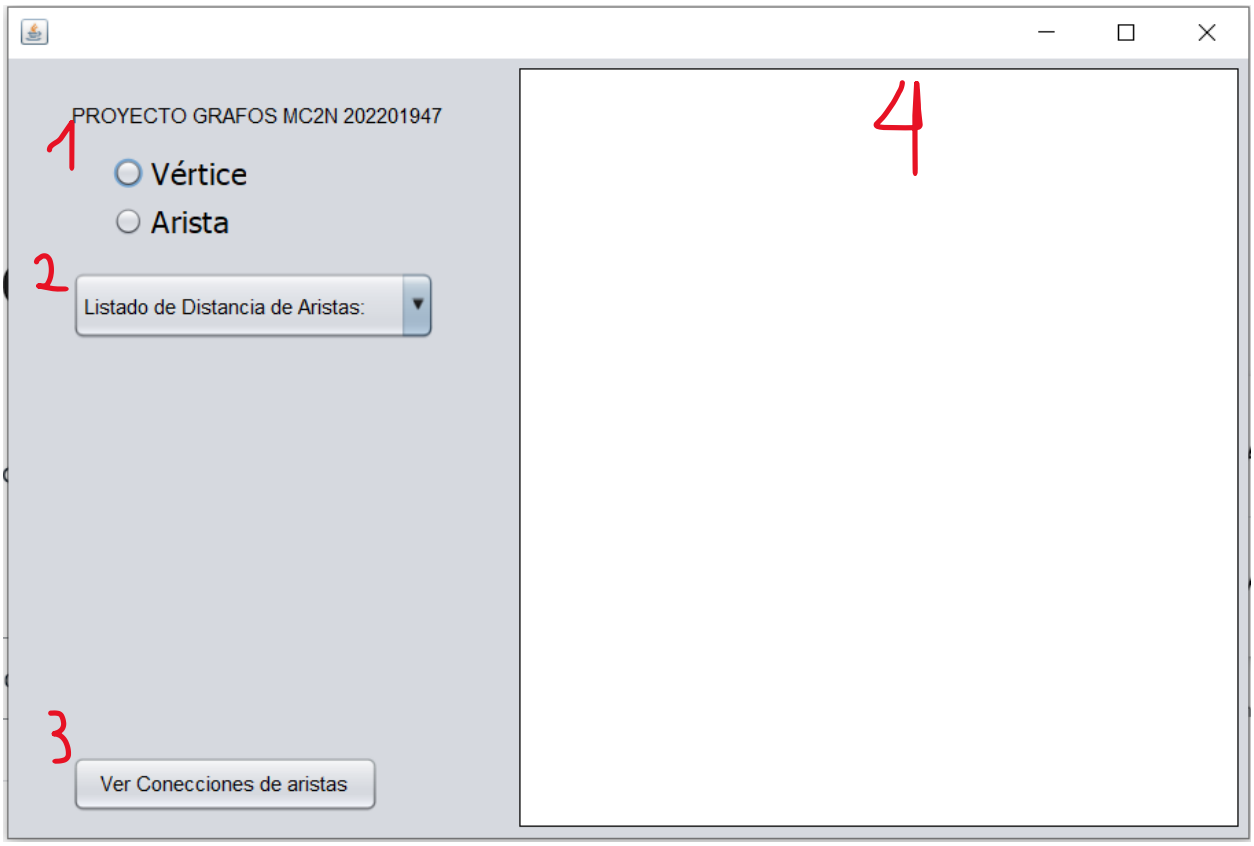
MATEMÁTICA PARA COMPUTACIÓN 2
Ing. Mario López
Aux. Diego Camey
Sección N

Nombre	Registro Académico
PABLO ANDRES RODRIGUEZ LIMA	202201947

Actividad	Correlativo	Fecha
PROYECTO FINAL	1	13/ MARZO / 2023

DESCRIPCIÓN DE CALIFICACIÓN	
Presentación (20)	
Ejercicios (80)	
TOTAL (100)	

Elementos gráficos

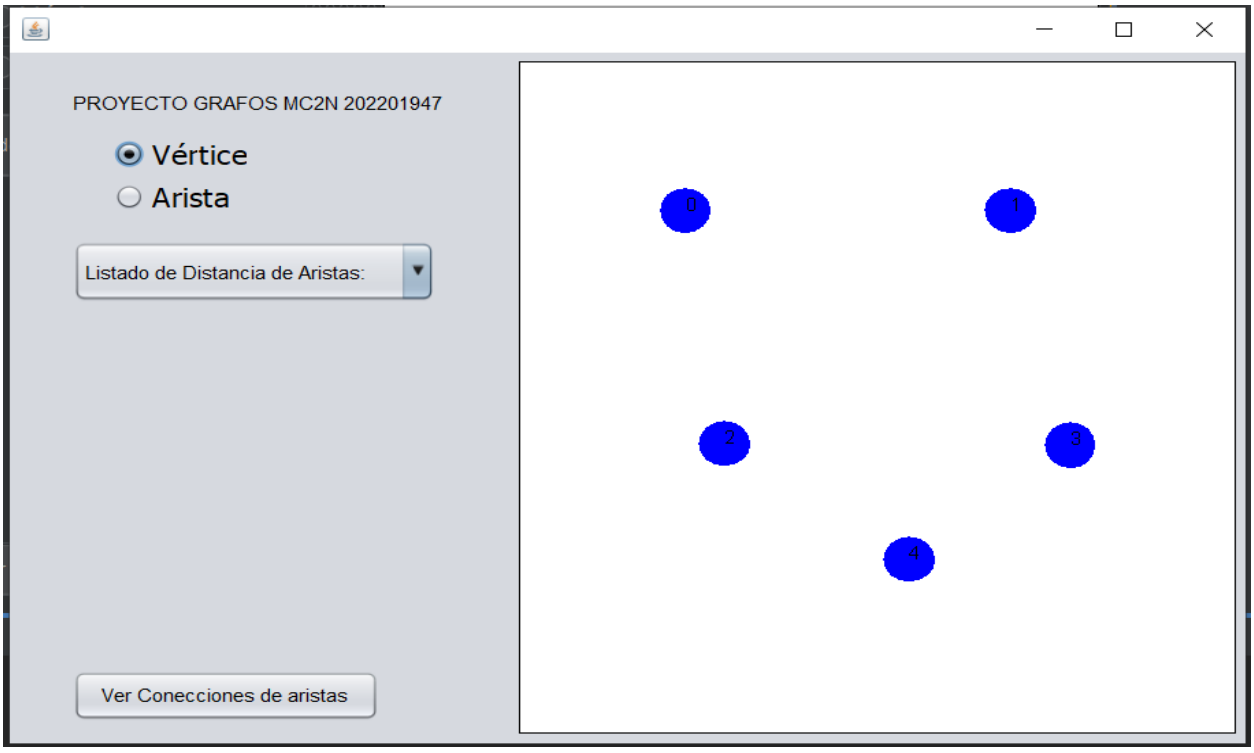


1. Es el apartado donde se selecciona que se quiere dibujar, es llamado radiobutton permite hacer la selección de un solo elemento que se quiera dibujar.

2. Como parte para agregar opcional existe un listado donde muestra la distancia que hay entre un vértice y otro o bien el peso de la arista esta es medida en pixeles ya que es la que utiliza el programa.

3. en consola de sistema se muestra de manera escrita de manera “teórica” como van unidos los vértices o que vértices tienen relación.

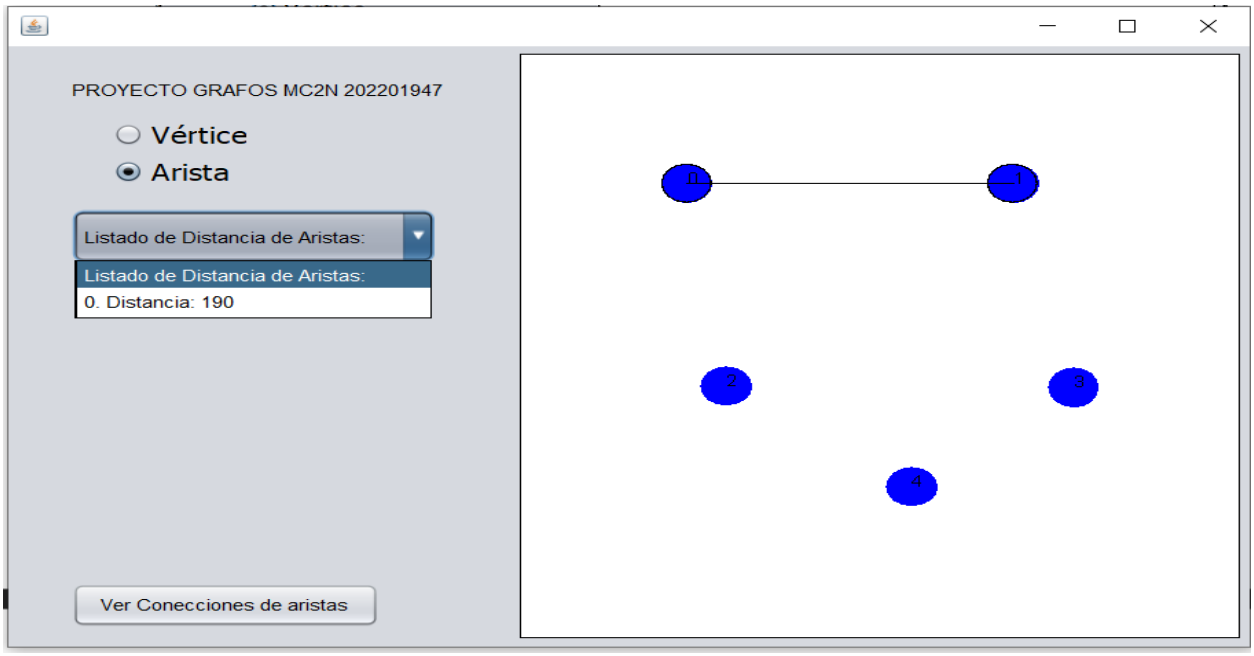
4. un panel en blanco que es el área de dibujo con lo que ahí se puede diseñar el grafo.



Al seleccionar el apartado de vértice nos habilita una opción con la que con el simple de dar un clic sobre el panel en blanco nos dibuja un punto y esto nos da libertad de dibujar el grafico que deseemos.

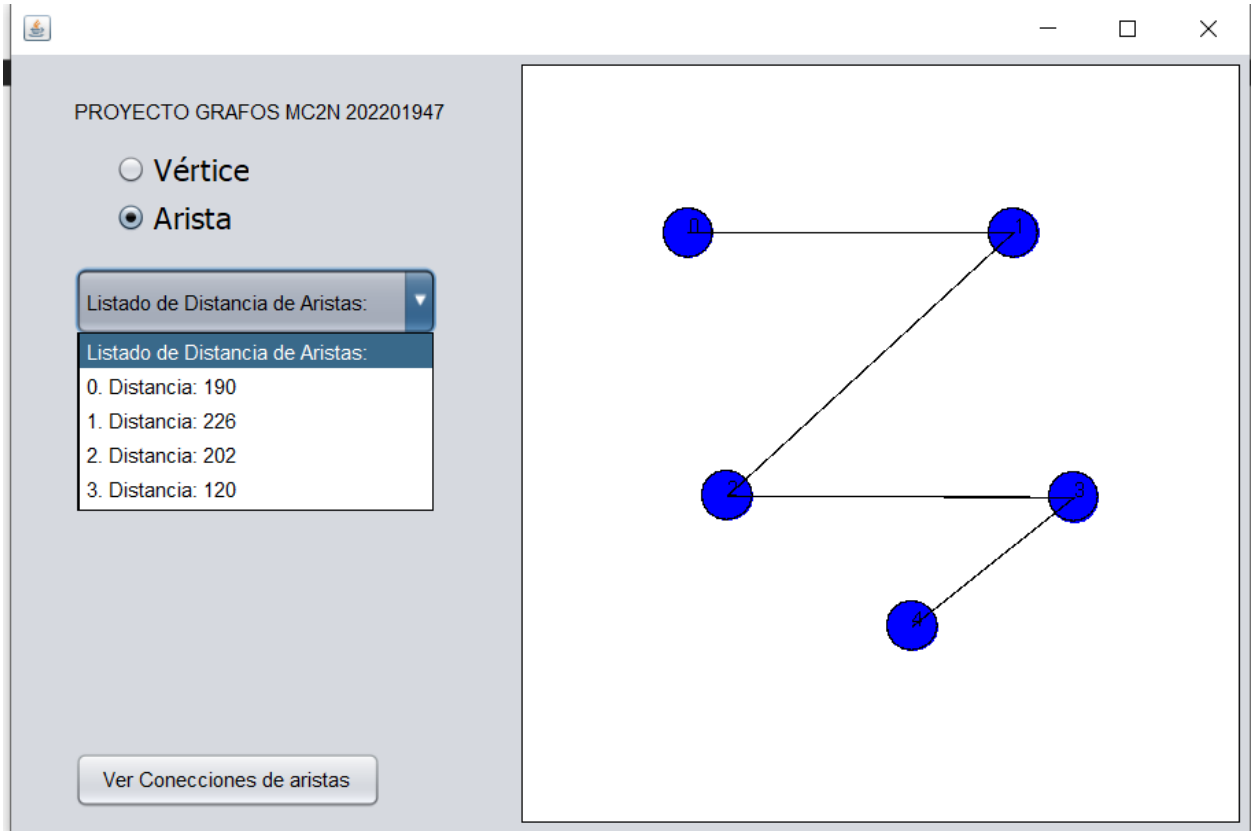
En la imagen de abajo se muestra el código de los atributos que posee cada punto, como lo es su nombre, posición en el eje x también sobre el eje y agregado el color que es para ver el vertice de mejor manera.

```
public class Vertice {

    int name;
    int posx, posy;
    Color color;

    public Vertice(int name, int posx, int posy, Color color) {
        this.name = name;
        this.posx = posx;
        this.posy = posy;
        this.color = color;
    }
```

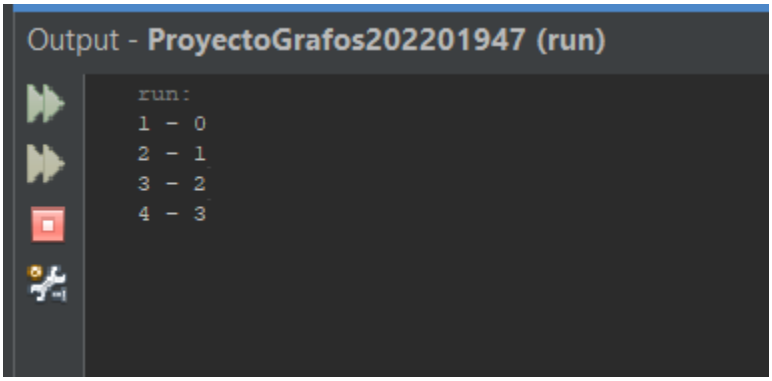


Al seleccionar el botón de arista, nos habilita una opción la cual se encarga de seleccionar el vértice y así también seleccionar otro como imagen de como armar un vértice, y sobre la pestaña opcional se muestra la distancia de la arista en cuestión. En los atributos de la arista se encuentra, la posición de los vértices que sirve para medir la distancia y también para saber cuál es el que inicia y cuál es el que termina.

```java
public class Arista {
    int nodoinicial;
    int nodofinal;
    int x1, y1, x2, y2;
    int dist;

    public Arista(int nodoinicial, int nodofinal, int x1, int y1, int x2, int y2, int dist) {
        this.nodoinicial = nodoinicial;
        this.nodofinal = nodofinal;
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
        this.dist = dist;
    }
```



Al terminar de graficar el grafo con las opciones dadas se selecciona el botón el cual nos permite ver el nombre de la arista y ver a cual esta conectado, esto nos puede dar una idea sobre el papel de como esta conformado el grafo.



Aspectos de Programación

```
ArrayList<Vertice> nodos;
ArrayList<Arista> arcos;
int cantNodos = 0;
Vertice nodoinicial = null;
Vertice nodofinal = null;
int tamNodos = 30;
int[][] matriz;
int a=0;
```

Variables y listas iniciales las cuales nos van a ayudar a guardar la información de nuestro grafo.

```
public Ventana() {
    initComponents();
    nodos = new ArrayList<Vertice>();
    arcos = new ArrayList<Arista>();
    Insertar(jPanel1.getGraphics());
}
```

Opciones de nuestra ventana donde se ve la ejecución del programa la cual nos ayuda a tener ya listo los receptores de información.

```
private void Insertar(Graphics g){
    jPanel1.addMouseListener(new MouseListener(){
        @Override
        public void mouseClicked(MouseEvent e) {
            if(jRadioButton1.isSelected()){
                g.setColor(Color.BLUE);
                g.fillOval(e.getX() - (tamNodos/2), e.getY() - (tamNodos/2), tamNodos, tamNodos);
                nodos.add(new Vertice(cantNodos, e.getX()- (tamNodos/2), e.getY()- (tamNodos/2), Color.WHITE));
                g.setColor(Color.BLACK);
                g.drawString(Integer.toString(cantNodos), e.getX(), e.getY());
                cantNodos++;
            }else{
                if(nodoinicial == null){
                    nodoinicial = BuscarNodo(e.getX(), e.getY());
                    if(nodoinicial != null){
                        seleccionarNodo(nodoinicial, g, Color.red);
                    }
                }else{
                    nodofinal = BuscarNodo(e.getX(), e.getY());
                    if(nodofinal!=null){
                        seleccionarNodo(nodofinal, g, Color.red);
                        if(nodofinal.name != nodoinicial.name){
                            g.setColor(Color.black);
                            g.drawLine(nodoinicial.posx + (tamNodos/2), nodoinicial.posy + (tamNodos/2),
                                    nodofinal.posx + (tamNodos/2), nodofinal.posy + (tamNodos/2));
                            int distancia = distancia(nodoinicial.posx + (tamNodos/2),
                                    nodofinal.posx + (tamNodos/2), nodoinicial.posy + (tamNodos/2), nodofinal.posy + (tamNodos/2));
                            arcos.add(new Arista(nodoinicial.name, nodofinal.name,
                                    nodoinicial.posx + (tamNodos/2), nodoinicial.posy + (tamNodos/2),
                                    nodofinal.posx + (tamNodos/2), nodofinal.posy + (tamNodos/2), distancia));

                            jComboBox1.addItem( a +". Distancia: " + distancia );
                            a++;
                            //System.out.println("distancia: " + distancia);
```

A grandes rasgos, se establece que pasa cuando se da clic en este caso es de la creación del vértice lo cual indica que tiene 30 pixeles de tamaño, que es un círculo, que es color azul y que el nombre saga del centro para evitar confusiones con otros elementos, además que hace los cálculos de la distancia para mostrarlos en el bloque anidado.

```
private int distancia(int x1, int x2, int y1, int y2){
    double dist = Math.sqrt(Math.pow(x2-x1, 2.0) + Math.pow(y2-y1, 2.0));
    return (int)dist;
}

private void seleccionarNodo(Vertice nodo, Graphics g, Color color){
    g.setColor(color);
    g.drawOval(nodo.posx, nodo.posy, tamNodos-1, tamNodos-1);
}

private Vertice BuscarNodo(int coordx, int coordy){
    Vertice nodoReturn = null;
    for (Vertice nodo : nodos) {
        if(coordx >= nodo.posx && coordx <= nodo.posx + tamNodos &&
                coordy >= nodo.posy && coordy <= nodo.posy + tamNodos){
            nodoReturn = nodo;
            break;
        }
    }
    return nodoReturn;
```

Un método el cual es llamado para hacer la operación de distancia, y para la arista le crea un contorno que indica que el nodo es seleccionado y también el ver que pasa si no se selecciona un vértice receptor el cual deshabilita la selección.

```java
private void calcularMatriz(){
    matriz = new int[nodos.size()][nodos.size()];
    for (Arista arco : arcos) {
        matriz[arco.nodoinicial][arco.nodofinal] = arco.dist;
        matriz[arco.nodofinal][arco.nodoinicial] = arco.dist;
    }
}

private void Prim(int[][] matriz){
    boolean vector[]  =new boolean[nodos.size()];
    vector[0] = true;
    while(faltaAlguno(vector)){
        int min = menor(matriz, vector);
        vector[min] = true;
    }
}

private int menor(int[][] matriz, boolean[] vector){
    int menor = Integer.MAX_VALUE;
    int fila = -1, col=-1;
    for (int i = 0; i < matriz.length; i++) {
        if(vector[i]){
            for (int j = 0; j < matriz.length; j++) {
                if(matriz[j][i]!=0 && vector[j]==false && matriz[j][i]<=menor){
                    menor = matriz[j][i];
                    fila = j;
                    col = i;
                }
            }
        }
    }

    System.out.println("" + fila + " - " + col);
    return fila;
}
```

Por último, es los datos de la matriz, como lo guardamos en dos listas es pertinente decir que si las unimos se trata de una matriz lo que hace es ver los nombres de los vértices y en base a las aristas crear una relación (que muestra en consola) y muestra el resultado.