|  |
| --- |
|  |
| Prueba de Caja Blanca |
| ***“Gestión de Ventas y Compras DIDEB”*** |
|  |
|  |
| **Integrantes:**  Sebastian Alvarez  Madely Betancourt  Luis Carvajal  Johanna Molina  Carlos Puco  **Fecha 2022-01-31** |

**ÍNDICE**

[CLIENTE 2](#_Toc96519729)

[1. Acción al presionar “Agregar” en “Agregar nuevo cliente” 2](#_Toc96519730)

[2. Acción al presionar “Modificar” en “Modificar cliente” 5](#_Toc96519731)

[3. Acción al presionar “Eliminar” en “Eliminar cliente” 7](#_Toc96519732)

[4. Acción al presionar “Buscar” en “Buscar cliente por nombre” 9](#_Toc96519733)

[5. Acción al presionar “Buscar” en “Buscar cliente por nombre de clínica” 11](#_Toc96519734)

[PROVEEDOR 12](#_Toc96519735)

[1. Agregar Proveedor 12](#_Toc96519736)

[2. Eliminar Proveedor 14](#_Toc96519737)

[3. Buscar Proveedor por nombre 15](#_Toc96519738)

[4. Buscar Proveedor por fabrica 17](#_Toc96519739)

[PRODUCTO 18](#_Toc96519740)

[Agregar producto 18](#_Toc96519741)

[Buscar Producto por Nombre 20](#_Toc96519742)

[Buscar Producto por rango de precio 21](#_Toc96519743)

[VENTA 26](#_Toc96519744)

[Agregar Venta 26](#_Toc96519745)

[Eliminar Venta 29](#_Toc96519746)

[Modificar Venta 30](#_Toc96519747)

[COMPRA 33](#_Toc96519748)

[Agregar compra 33](#_Toc96519749)

[Ver compra 35](#_Toc96519750)

[Eliminar compra 38](#_Toc96519751)

[Buscar compra por id 40](#_Toc96519752)

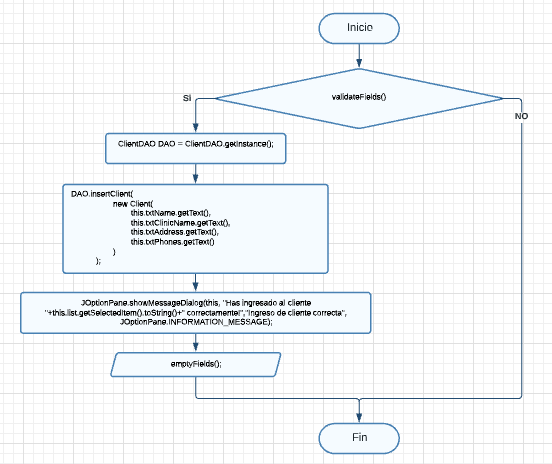
# CLIENTE

## Acción al presionar “Agregar” en “Agregar nuevo cliente”

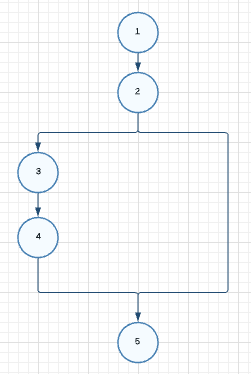
**CÓDIGO FUENTE**

|  |
| --- |
| if(validateFields()){  ClientDAO DAO = ClientDAO.getInstance();  DAO.insertClient(  new Client(  this.txtName.getText(),  this.txtClinicName.getText(),  this.txtAddress.getText(),  this.txtPhones.getText()  )  );    JOptionPane.showMessageDialog(this, "Has ingresado al cliente "+this.txtName.getText()+" correctamente!","Ingreso de cliente correcto", JOptionPane.INFORMATION\_MESSAGE);  emptyFields();  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5

**R2:** 1, 2, 5

**COMPLEJIDAD CICLOMÁTICA**

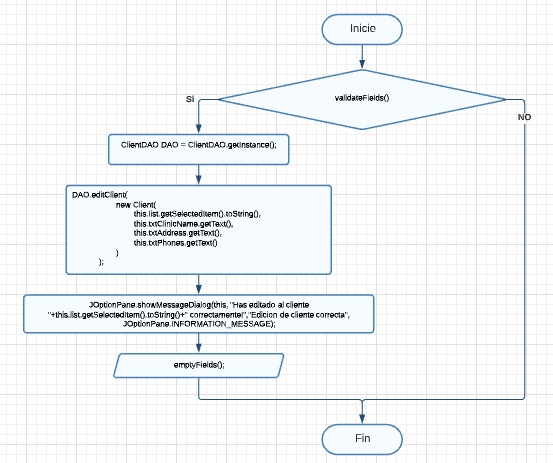
V(G)= 5 - 5 + 2 = 2

## Acción al presionar “Modificar” en “Modificar cliente”

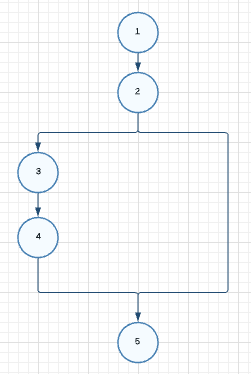
**CÓDIGO FUENTE**

|  |
| --- |
| if(validateFields()){  ClientDAO DAO = ClientDAO.getInstance();  DAO.editClient(  new Client(  this.list.getSelectedItem().toString(),  this.txtClinicName.getText(),  this.txtAddress.getText(),  this.txtPhones.getText()  )  );    JOptionPane.showMessageDialog(this, "Has editado al cliente "+this.list.getSelectedItem().toString()+" correctamente!","Edicion de cliente correcta", JOptionPane.INFORMATION\_MESSAGE);  emptyFields();  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5, 6

**R2:** 1, 2, 6

**COMPLEJIDAD CICLOMÁTICA**

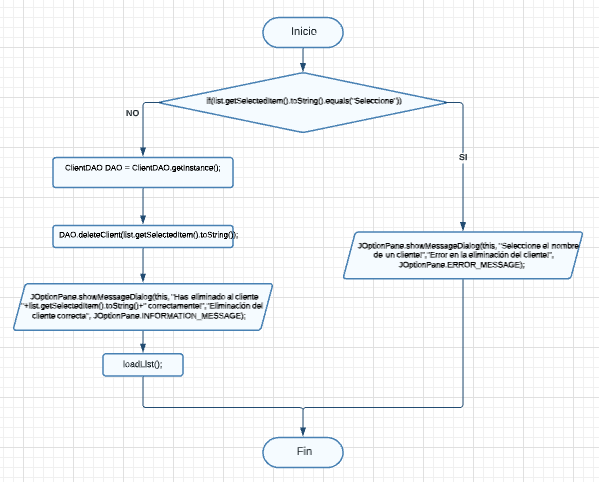
V(G)= 5 - 5 + 2 = 2

## Acción al presionar “Eliminar” en “Eliminar cliente”

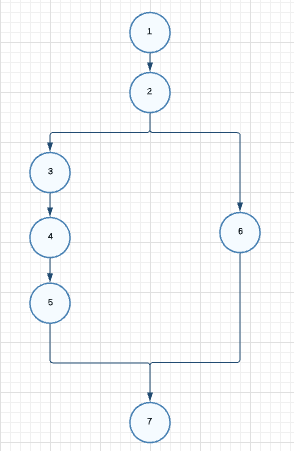
**CÓDIGO FUENTE**

|  |
| --- |
| if(list.getSelectedItem().toString().equals("Seleccione")){  JOptionPane.showMessageDialog(this, "Seleccione el nombre de un cliente!","Error en la eliminación del cliente!", JOptionPane.ERROR\_MESSAGE);  } else {  ClientDAO DAO = ClientDAO.getInstance();  DAO.deleteClient(list.getSelectedItem().toString());    JOptionPane.showMessageDialog(this, "Has eliminado al cliente "+list.getSelectedItem().toString()+" correctamente!","Eliminación del cliente correcta", JOptionPane.INFORMATION\_MESSAGE);  loadList();  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5, 7

**R2:** 1, 2, 6, 7

**COMPLEJIDAD CICLOMÁTICA**

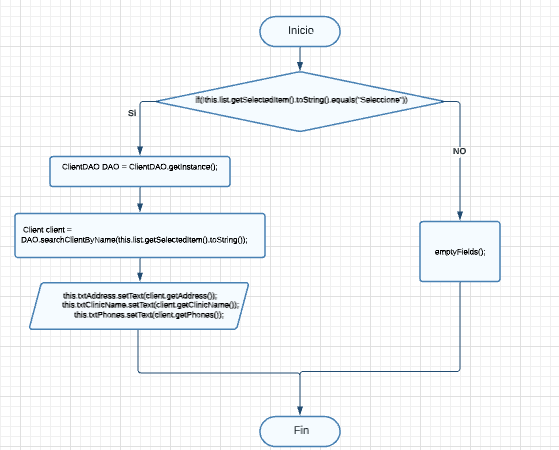
V(G)= 7 - 7 + 2 = 2

## Acción al presionar “Buscar” en “Buscar cliente por nombre”

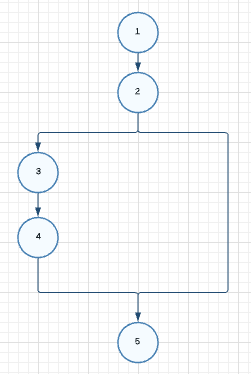
**CÓDIGO FUENTE**

|  |
| --- |
| if(!this.list.getSelectedItem().toString().equals("Seleccione")){  ClientDAO DAO = ClientDAO.getInstance();  Client client = DAO.searchClientByName(this.list.getSelectedItem().toString());  this.txtAddress.setText(client.getAddress());  this.txtClinicName.setText(client.getClinicName());  this.txtPhones.setText(client.getPhones());  } else {  emptyFields();  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5

**R2:** 1, 2, 5

**COMPLEJIDAD CICLOMÁTICA**

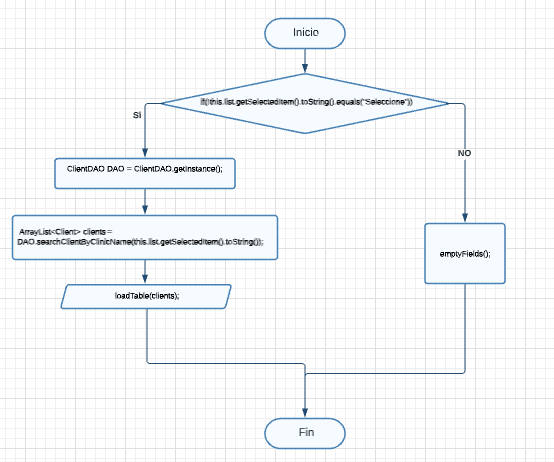
V(G)= 5 – 5 + 2 = 2

## Acción al presionar “Buscar” en “Buscar cliente por nombre de clínica”

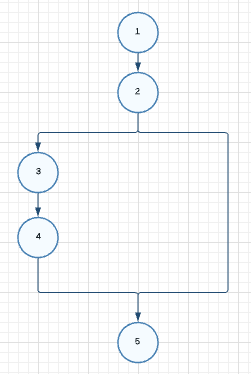
**CÓDIGO FUENTE**

|  |
| --- |
| if(!this.list.getSelectedItem().toString().equals("Seleccione")){  ClientDAO DAO = ClientDAO.getInstance();  ArrayList<Client> clients = DAO.searchClientByClinicName(this.list.getSelectedItem().toString());  loadTable(clients);  } else {  emptyFields();  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5

**R2:** 1, 2, 5

**COMPLEJIDAD CICLOMÁTICA**

V(G)= 5 - 5 + 2 = 2

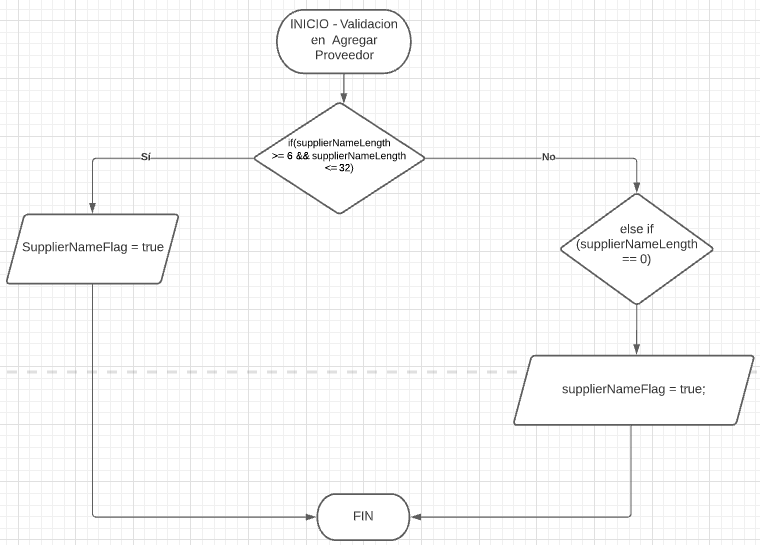
# PROVEEDOR

## Agregar Proveedor

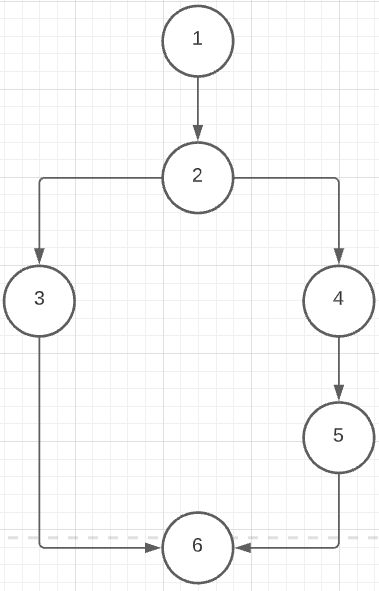
**CÓDIGO FUENTE**

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 6.

**R2:** 1, 2, 4, 5, 6.

**COMPLEJIDAD CICLOMÁTICA**

V(G)= 5 – 5 + 2 = 2

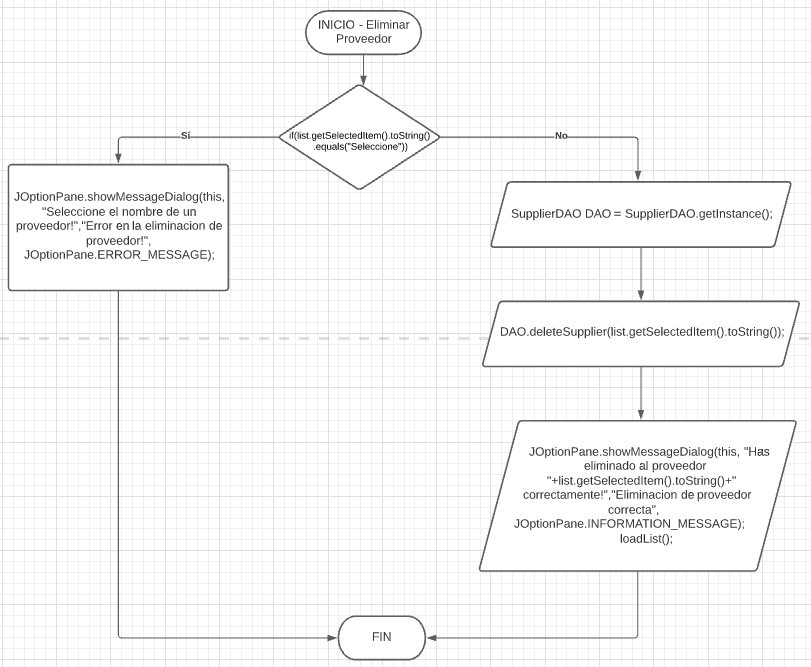
## 

## Eliminar Proveedor

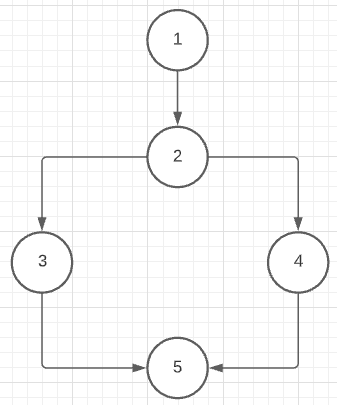
**CÓDIGO FUENTE**

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 5.

**R2:** 1, 2, 4, 5.

**COMPLEJIDAD CICLOMÁTICA**

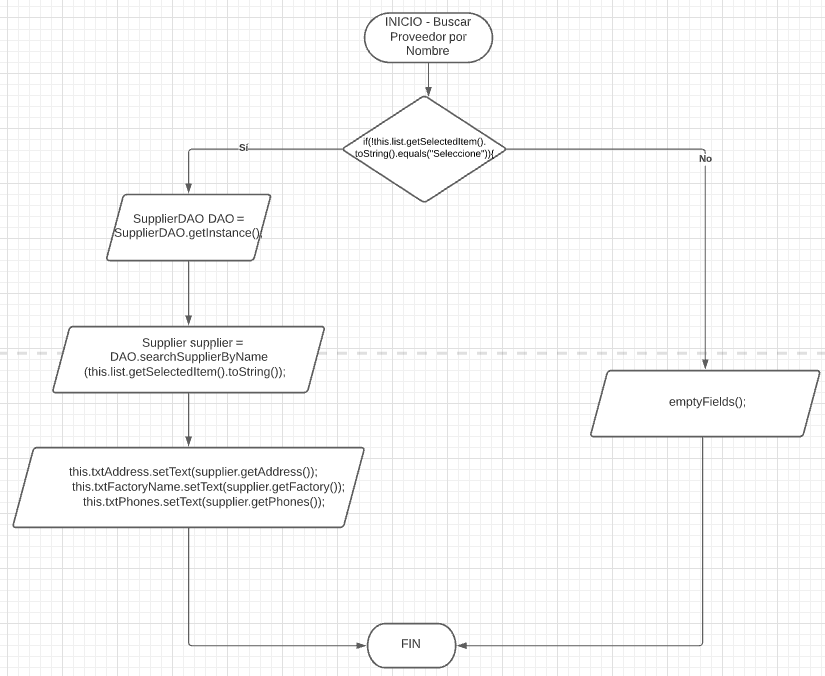
V(G)= 5 – 5 + 2 = 2

## Buscar Proveedor por nombre

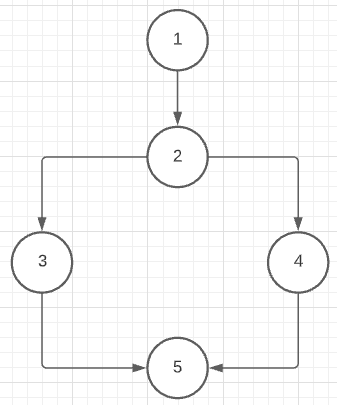
**CÓDIGO FUENTE**

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 5.

**R2:** 1, 2, 4, 5.

**COMPLEJIDAD CICLOMÁTICA**

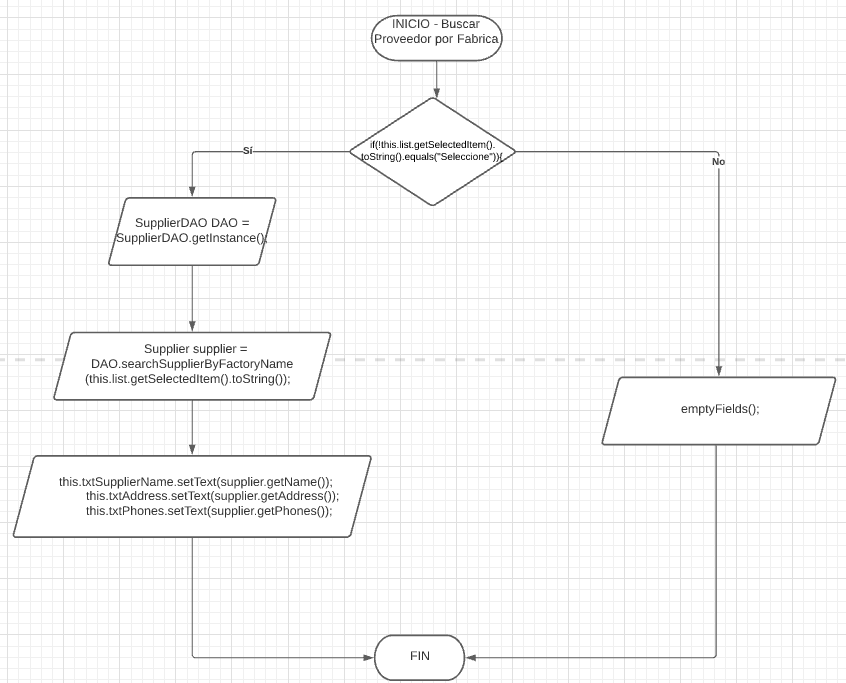
V(G)= 5 – 5 + 2 = 2

## Buscar Proveedor por fabrica

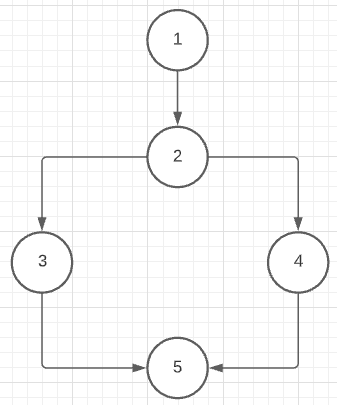
**CÓDIGO FUENTE**

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 5.

**R2:** 1, 2, 4, 5.

**COMPLEJIDAD CICLOMÁTICA**

V(G)= 5 – 5 + 2 = 2

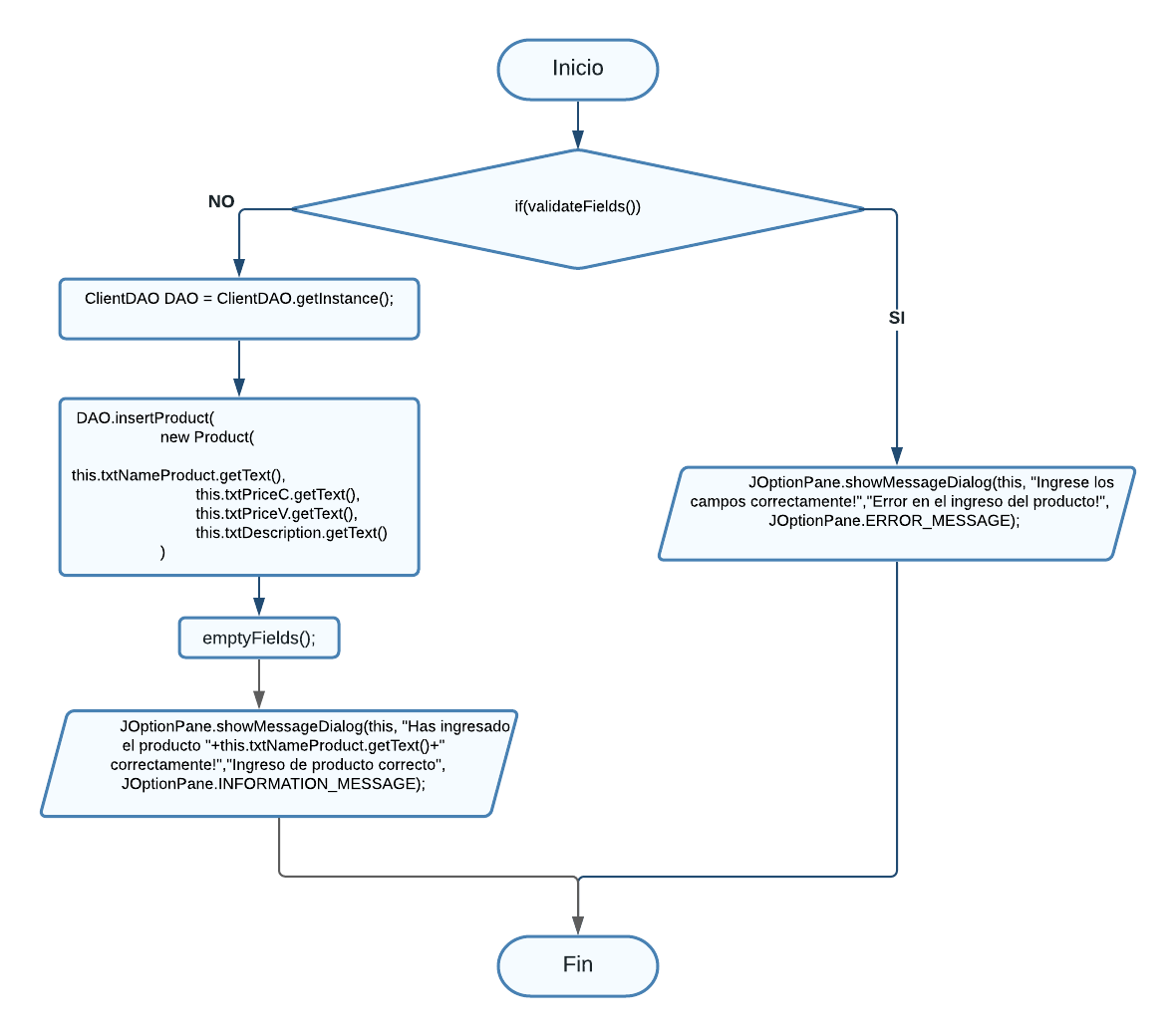
# PRODUCTO

## Agregar producto

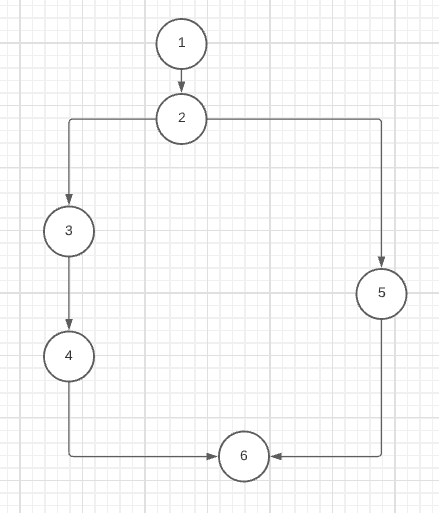
**CÓDIGO FUENTE**

|  |
| --- |
| if(validateFields()){  ProductDAO DAO = ProductDAO.getInstance();  DAO.insertProduct(  new Product(  this.txtNameProduct.getText(),  this.txtPriceC.getText(),  this.txtPriceV.getText(),  this.txtDescription.getText()  )  );    emptyFields();  JOptionPane.showMessageDialog(this, "Has ingresado el producto "+this.txtNameProduct.getText()+" correctamente!","Ingreso de producto correcto", JOptionPane.INFORMATION\_MESSAGE);  } else {  JOptionPane.showMessageDialog(this, "Ingrese los campos correctamente!","Error en el ingreso del producto!", JOptionPane.ERROR\_MESSAGE);  } |

**Diagrama de flujo**



**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 6

**R2:** 1, 2, 5,6

**COMPLEJIDAD CICLOMÁTICA**

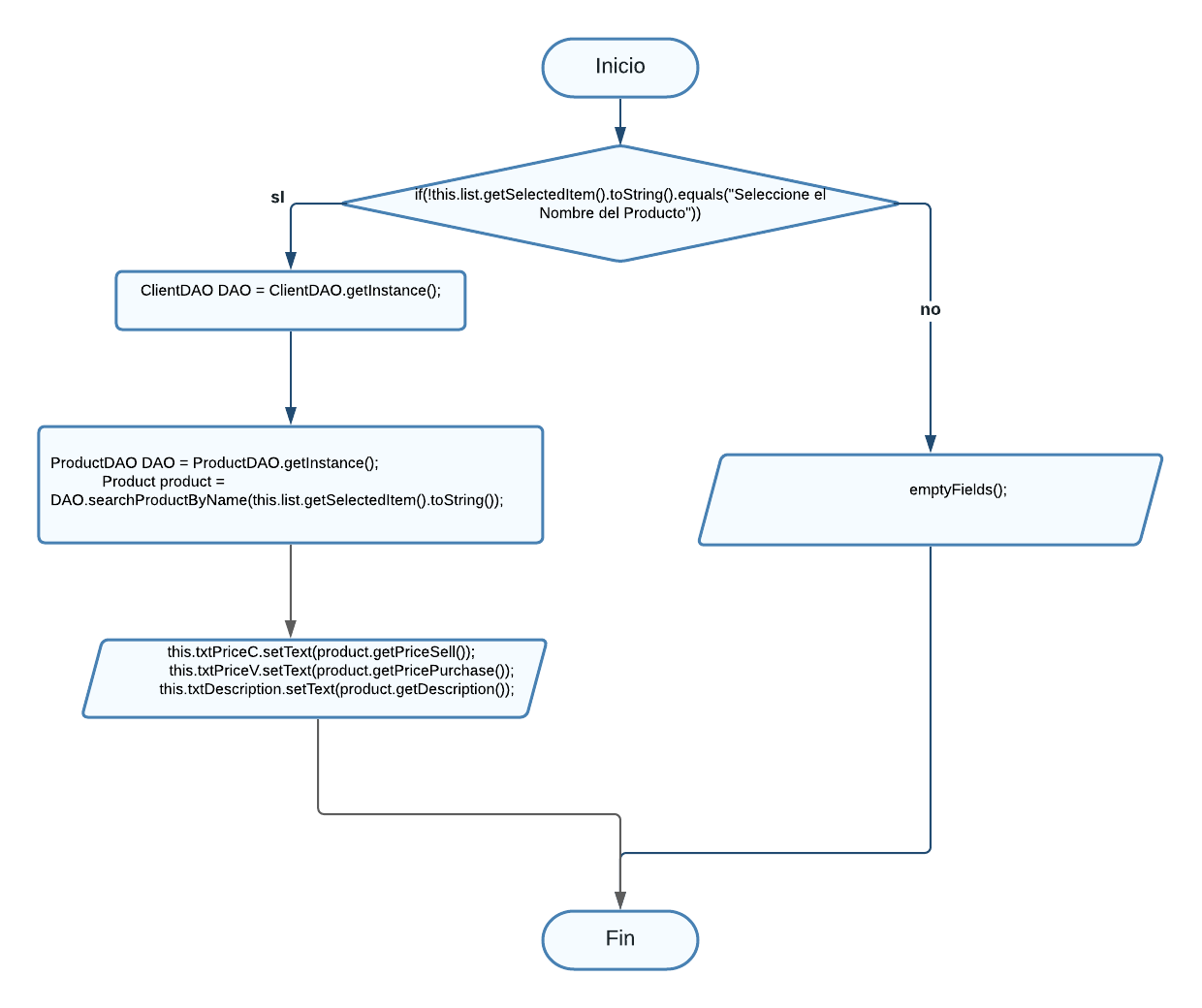
V(G)= 6 – 6 + 2 = 2

## Buscar Producto por Nombre

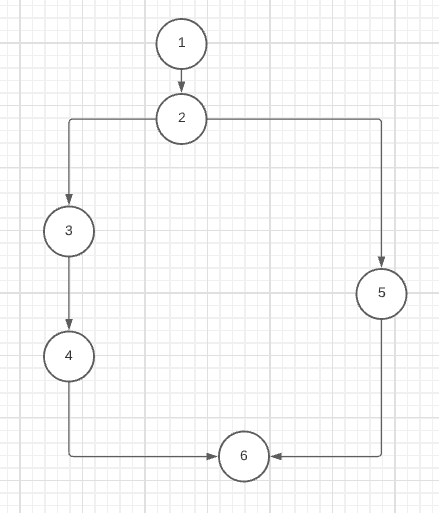
**CÓDIGO FUENTE**

|  |
| --- |
| if(!this.list.getSelectedItem().toString().equals("Seleccione el Nombre del Producto")){  ProductDAO DAO = ProductDAO.getInstance();  Product product = DAO.searchProductByName(this.list.getSelectedItem().toString());  this.txtPriceC.setText(product.getPriceSell());  this.txtPriceV.setText(product.getPricePurchase());  this.txtDescription.setText(product.getDescription());  } else {  emptyFields();  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 6

**R2:** 1, 2, 5, 6

**COMPLEJIDAD CICLOMÁTICA**

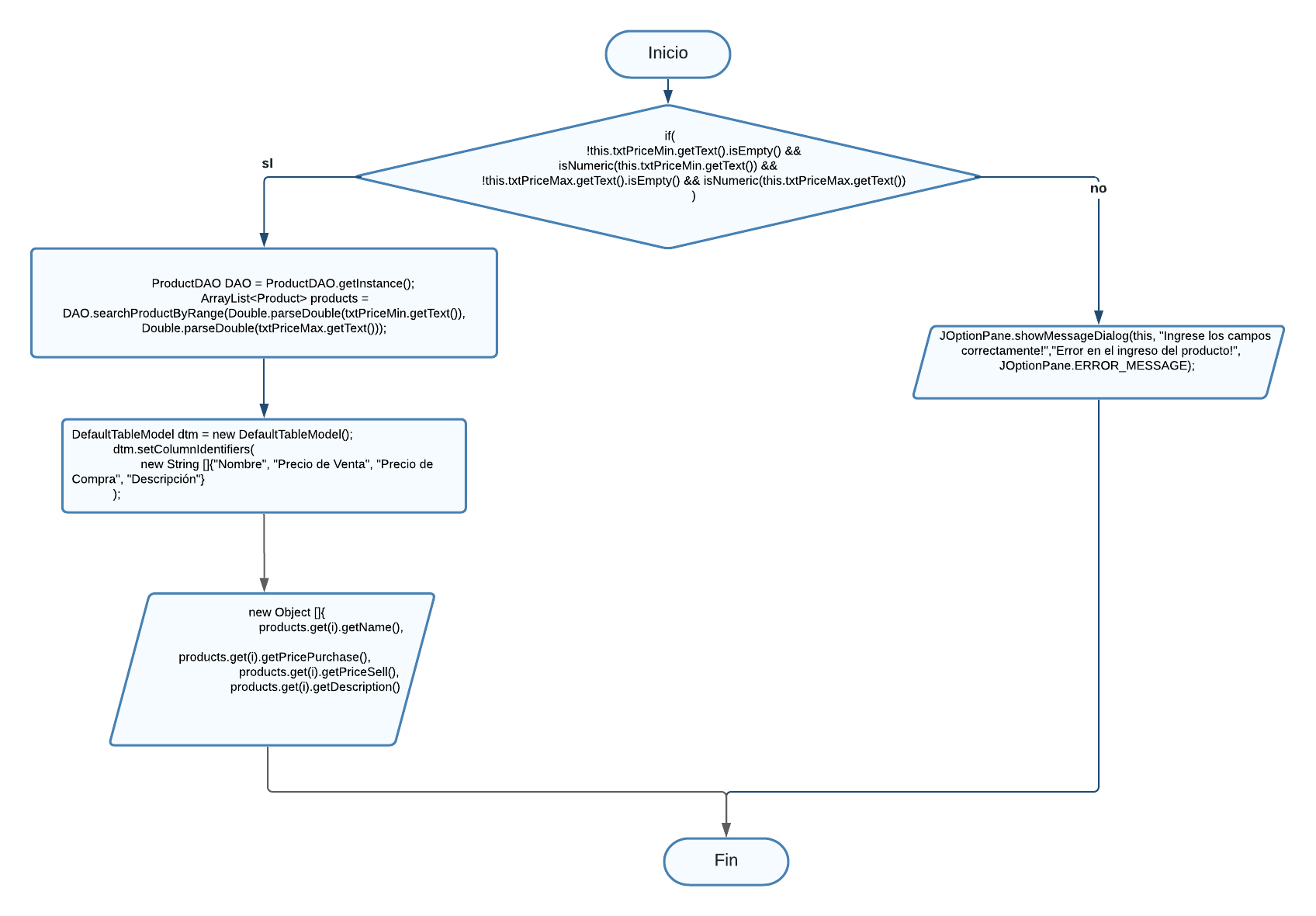
V(G)= 6 – 6 + 2 = 2

## Buscar Producto por rango de precio

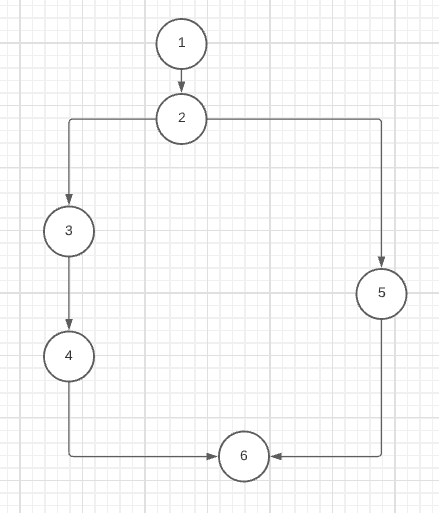
**CÓDIGO FUENTE**

|  |
| --- |
| private void loadTable() {  if(  !this.txtPriceMin.getText().isEmpty() && isNumeric(this.txtPriceMin.getText()) &&  !this.txtPriceMax.getText().isEmpty() && isNumeric(this.txtPriceMax.getText())  ){    ProductDAO DAO = ProductDAO.getInstance();  ArrayList<Product> products = DAO.searchProductByRange(Double.parseDouble(txtPriceMin.getText()), Double.parseDouble(txtPriceMax.getText()));  DefaultTableModel dtm = new DefaultTableModel();  dtm.setColumnIdentifiers(  new String []{"Nombre", "Precio de Venta", "Precio de Compra", "Descripción"}  );  for (int i = 0; i < products.size(); i++) {  dtm.addRow(  new Object []{  products.get(i).getName(),  products.get(i).getPricePurchase(),  products.get(i).getPriceSell(),  products.get(i).getDescription()  }  );  }  this.table.setModel(dtm);  } else {  JOptionPane.showMessageDialog(this, "Ingrese los campos correctamente!","Error en el ingreso del producto!", JOptionPane.ERROR\_MESSAGE);  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 6

**R2:** 1, 2, 5, 6

**COMPLEJIDAD CICLOMÁTICA**

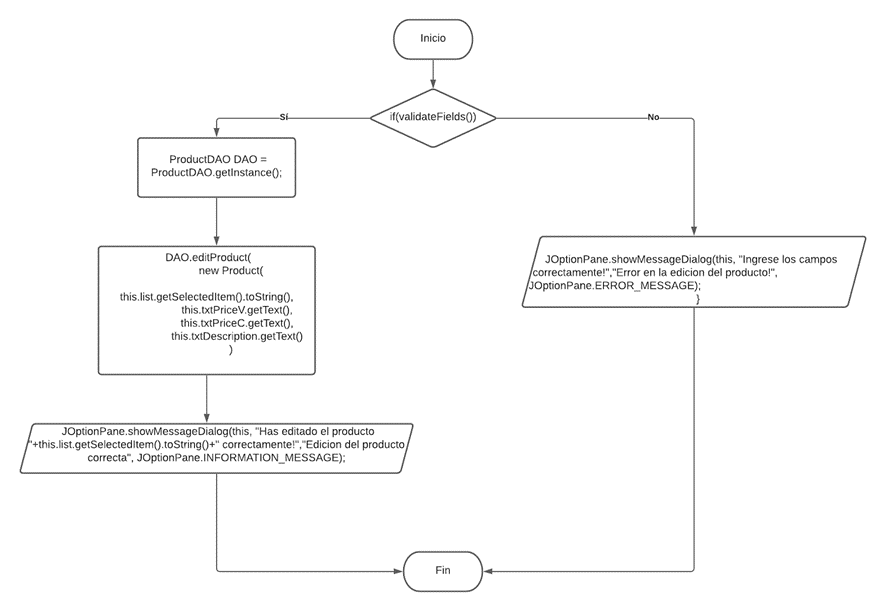
V(G)= 6 – 6 + 2 = 2

**Editar producto**

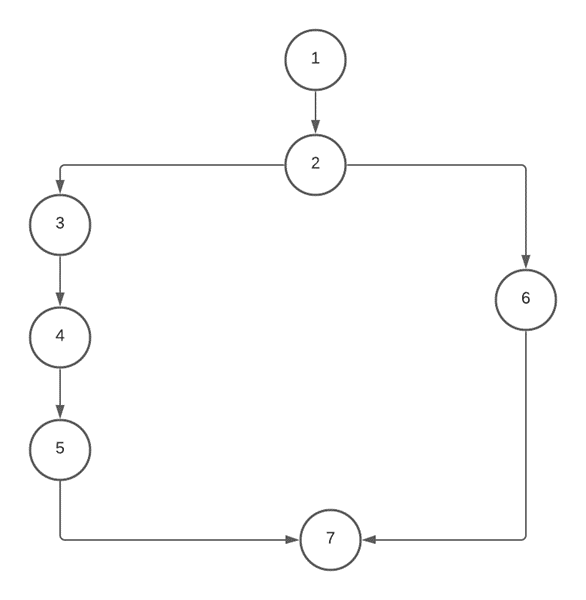
**CÓDIGO FUENTE**

|  |
| --- |
| if(validateFields()){  ProductDAO DAO = ProductDAO.getInstance();  DAO.editProduct(  new Product(  this.list.getSelectedItem().toString(),  this.txtPriceV.getText(),  this.txtPriceC.getText(),  this.txtDescription.getText()  )  );  emptyFields();  JOptionPane.showMessageDialog(this, "Has editado el producto "+this.list.getSelectedItem().toString()+" correctamente!","Edicion del producto correcta", JOptionPane.INFORMATION\_MESSAGE);  } else {  JOptionPane.showMessageDialog(this, "Ingrese los campos correctamente!","Error en la edicion del producto!", JOptionPane.ERROR\_MESSAGE);  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

R1: 1, 2, 3, 4, 5, 7

R2: 1, 2, 6, 7

**COMPLEJIDAD CICLOMÁTICA**

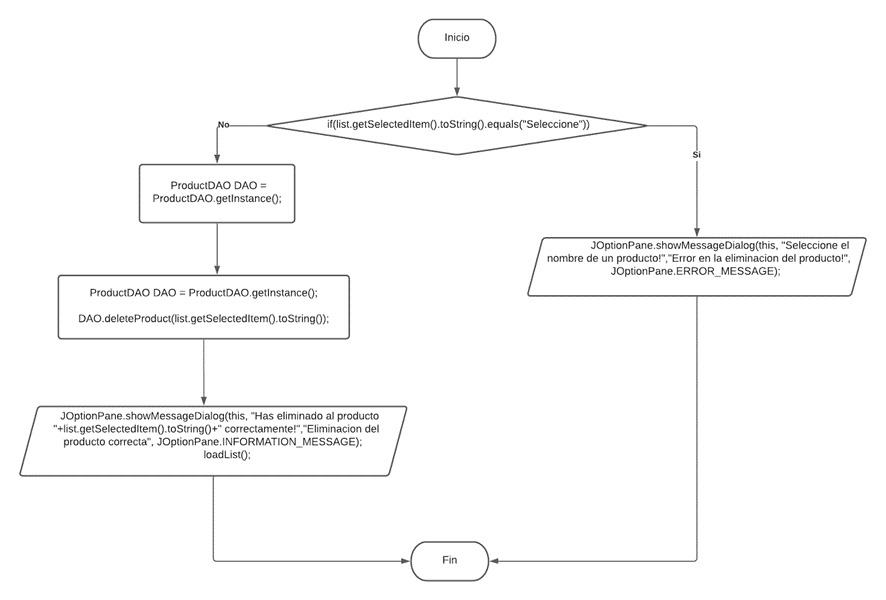
V(G)= 7 – 7 + 2 = 2

**Eliminar producto**

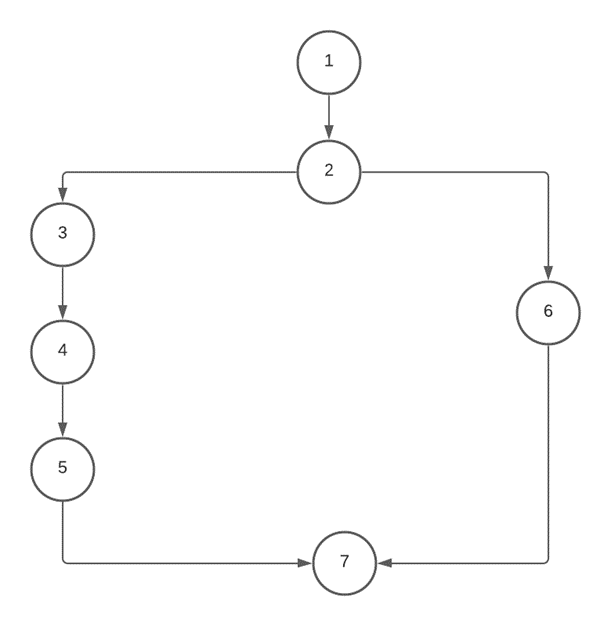
**CÓDIGO FUENTE**

|  |
| --- |
| if(list.getSelectedItem().toString().equals("Seleccione")){  JOptionPane.showMessageDialog(this, "Seleccione el nombre de un producto!","Error en la eliminacion del producto!", JOptionPane.ERROR\_MESSAGE);  } else {  ProductDAO DAO = ProductDAO.getInstance();  DAO.deleteProduct(list.getSelectedItem().toString());    JOptionPane.showMessageDialog(this, "Has eliminado al producto "+list.getSelectedItem().toString()+" correctamente!","Eliminacion del producto correcta", JOptionPane.INFORMATION\_MESSAGE);  loadList();  }  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

R1: 1, 2, 3, 4, 5, 7

R2: 1, 2, 6, 7

**COMPLEJIDAD CICLOMÁTICA**

V(G)= 7 – 7 + 2 = 2

# VENTA

## 

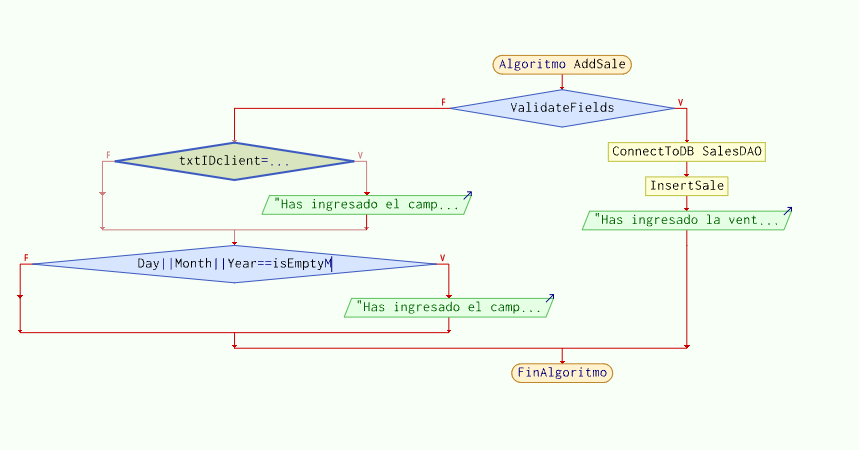
## Agregar Venta

**CÓDIGO FUENTE**

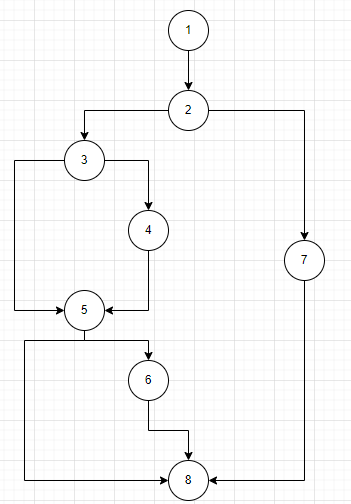
|  |
| --- |
|  |

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 7, 8.

**R2:** 1, 2, 3, 4, 5, 6, 8

**R3:** 1, 2, 3, 5,6, 8

**R4:** 1, 2, 3,4, 5, 6, 10

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=3+1=4

* V(G) = A – N + 2

V(G)= 10– 8 + 2 =4

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

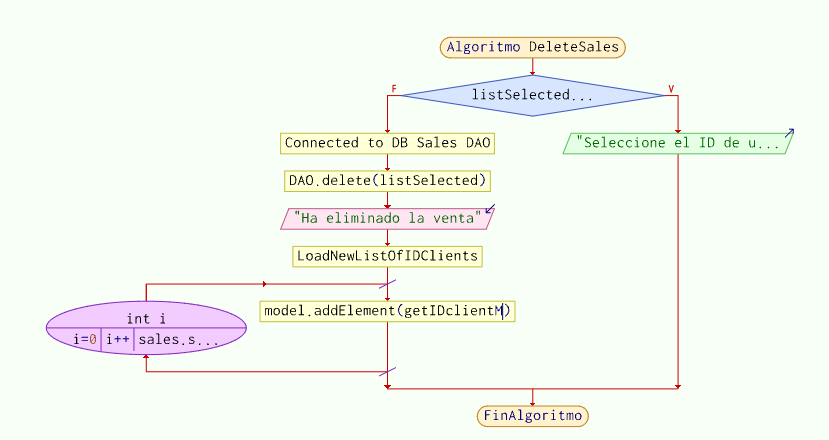
## Eliminar Venta

**CÓDIGO FUENTE**

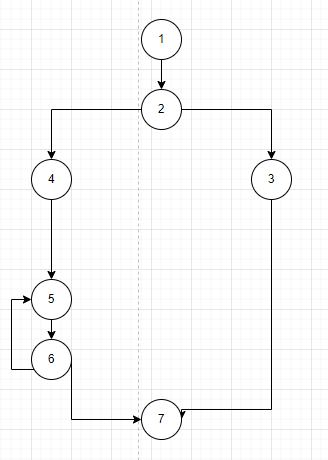
|  |
| --- |
|  |

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 7

**R2:** 1, 2, 3, 4,( 5, 6)^n, 7

**R2:** 1, 2, 3, 4,5,6, 7

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=2+1=3

* V(G) = A – N + 2

V(G)= 8– 7 + 2 = 3

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

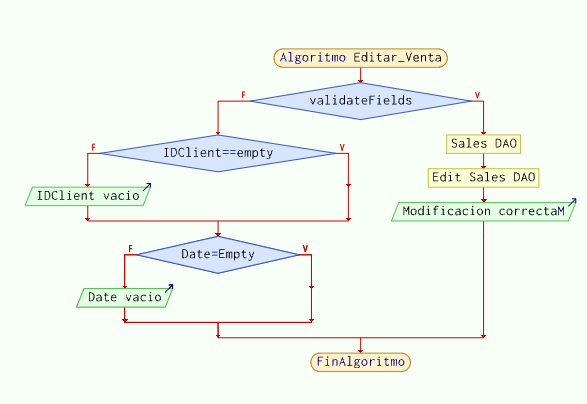
## Modificar Venta

**CÓDIGO FUENTE**

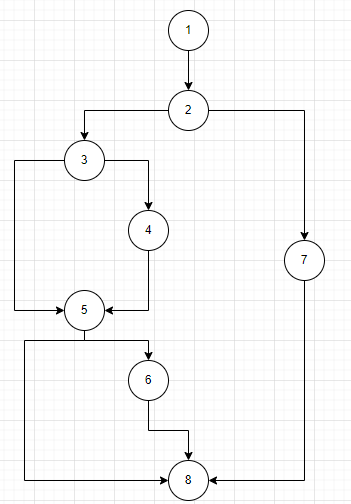
|  |
| --- |
|  |

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 7, 8.

**R2:** 1, 2, 3, 4, 5, 6, 8

**R3:** 1, 2, 3, 5,6, 8

**R4:** 1, 2, 3,4, 5, 6, 10

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=3+1=4

* V(G) = A – N + 2

V(G)= 10– 8 + 2 =4

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

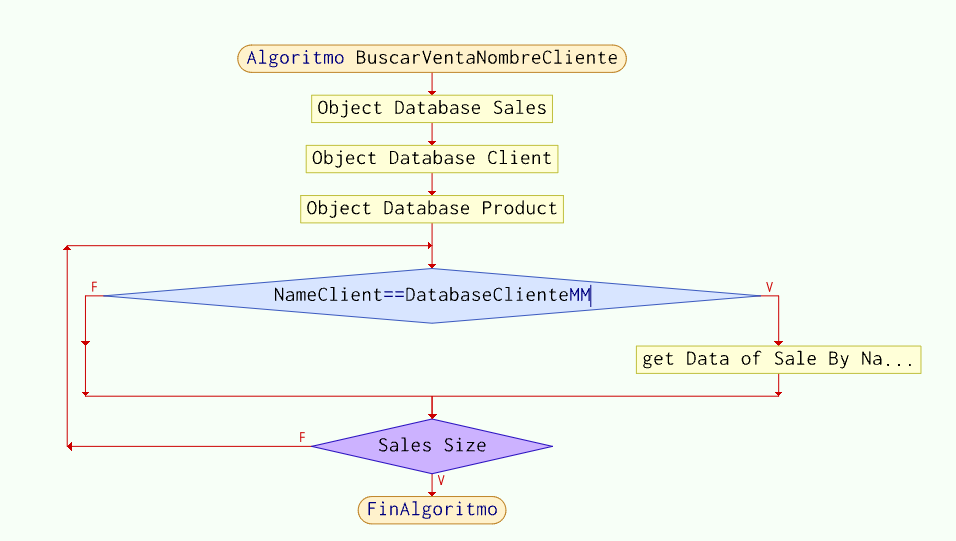
**N:** Número de nodos

## Buscar Venta por Nombre del Cliente

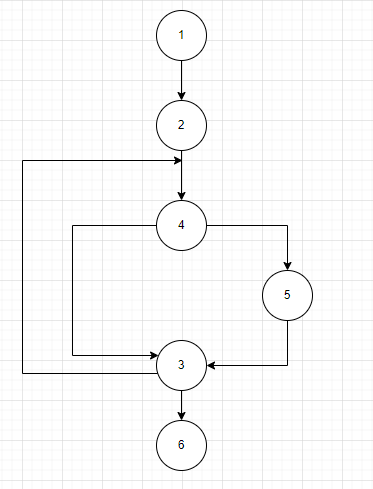
**CÓDIGO FUENTE**

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5, 6

**R2:** 1, 2, (3, 4, 5)^n, 6

**R3:** 1, 2, 3, 4, 6

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=2+1=3

* V(G) = A – N + 2

V(G)= 7– 6 + 2 =3

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

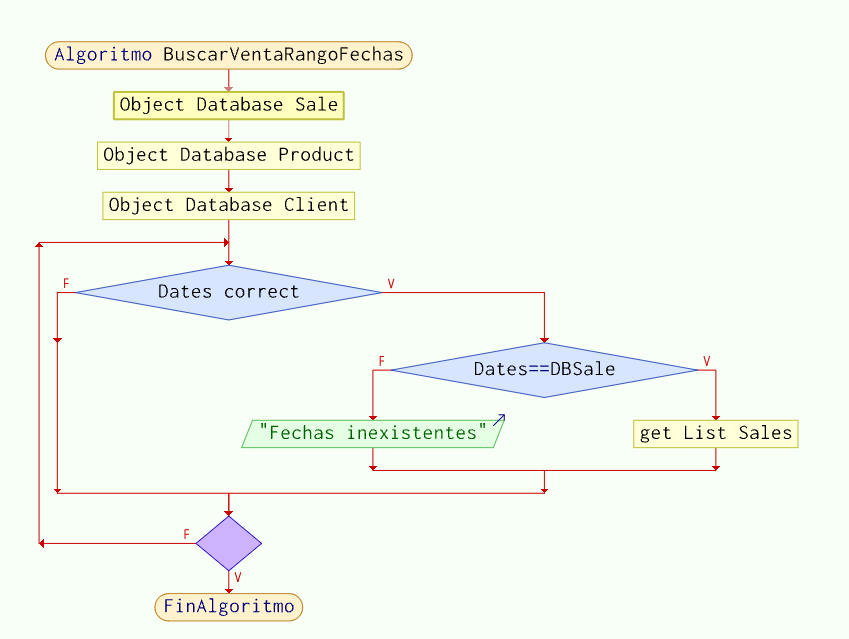
**N:** Número de nodos

## Buscar Venta por Rango de Fechas

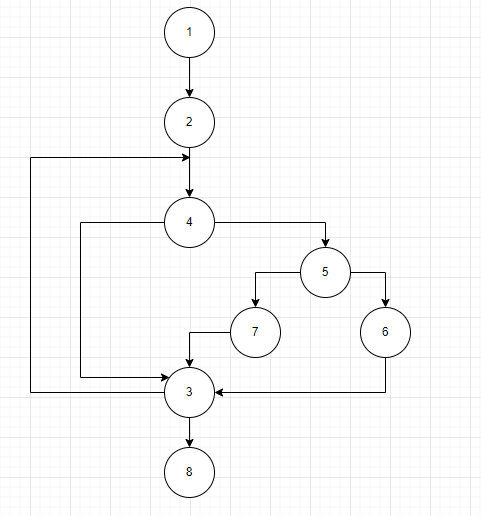
**CÓDIGO FUENTE**

|  |
| --- |
|  |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5, 6, 8

**R2:** 1, 2, 3, 4, 5, 7, 8

**R3:** 1, 2, 3, 4, 8

**R4:** 1, 2, (3, 4, 5, 6)^n, 8

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=3+1=4

* V(G) = A – N + 2

V(G)= 10– 8 + 2 =4

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

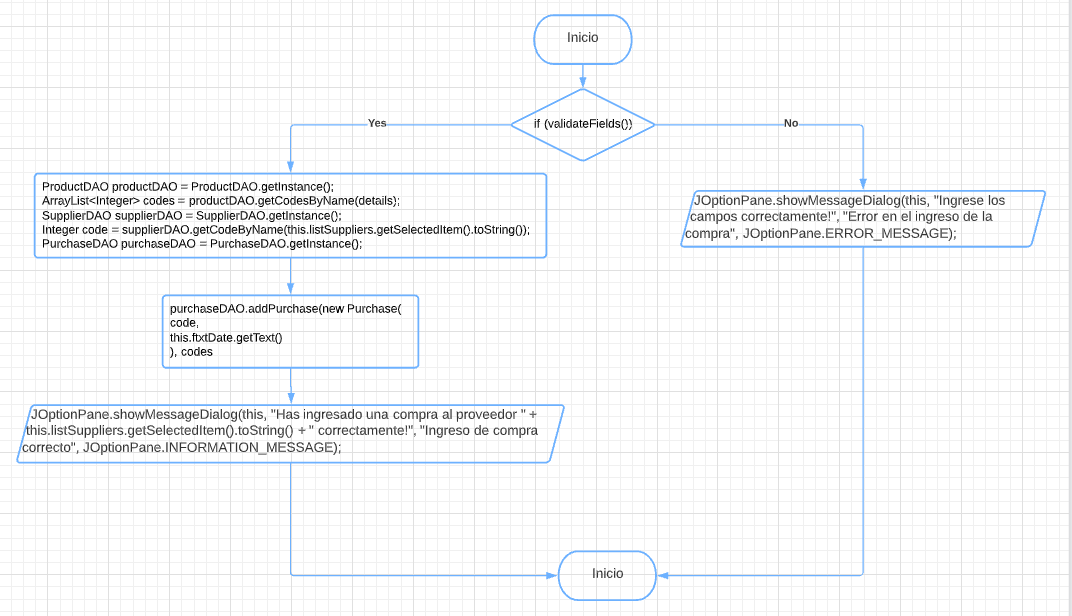
# COMPRA

## Agregar compra

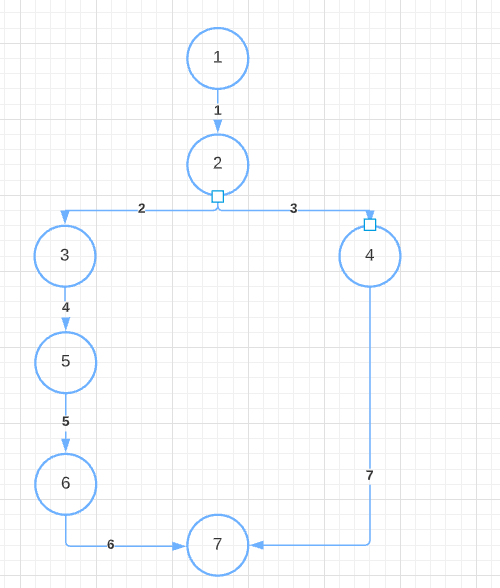
**CÓDIGO FUENTE**

|  |
| --- |
| if (validateFields()) {  ProductDAO productDAO = ProductDAO.getInstance();  ArrayList<Integer> codes = productDAO.getCodesByName(details);  SupplierDAO supplierDAO = SupplierDAO.getInstance();  Integer code = supplierDAO.getCodeByName(this.listSuppliers.getSelectedItem().toString());  PurchaseDAO purchaseDAO = PurchaseDAO.getInstance();  purchaseDAO.addPurchase(new Purchase(  code,  this.ftxtDate.getText()  ), codes  ); |

**Diagrama de flujo**



**GRAFO**

****

**RUTAS**

**R1:** 1,3,7

**R2:** 1, 2, 4, 5, 6

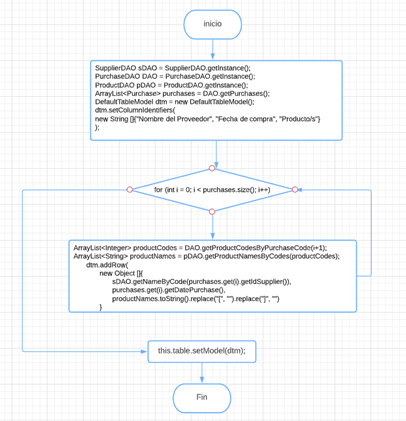
**COMPLEJIDAD CICLOMÁTICA V(G)=** 7 – 7 + 2 = 2

## Ver compra

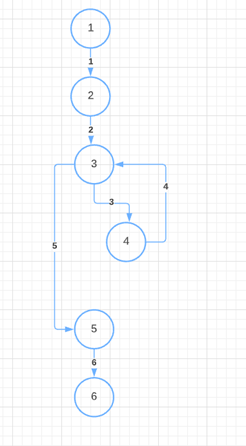
**CÓDIGO FUENTE**

|  |
| --- |
| SupplierDAO sDAO = SupplierDAO.getInstance();  PurchaseDAO DAO = PurchaseDAO.getInstance();  ProductDAO pDAO = ProductDAO.getInstance();  ArrayList<Purchase> purchases = DAO.getPurchases();  DefaultTableModel dtm = new DefaultTableModel();  dtm.setColumnIdentifiers(  new String []{"Nombre del Proveedor", "Fecha de compra", "Producto/s"}  );  for (int i = 0; i < purchases.size(); i++) {  ArrayList<Integer> productCodes = DAO.getProductCodesByPurchaseCode(i+1);  ArrayList<String> productNames = pDAO.getProductNamesByCodes(productCodes);    dtm.addRow(  new Object []{  sDAO.getNameByCode(purchases.get(i).getIdSupplier()),  purchases.get(i).getDatePurchase(),  productNames.toString().replace("[", "").replace("]", "")  }  );  } |

**Diagrama de flujo**



**GRAFO**

****

**RUTAS**

**R1:** 1,2,3,4,5,6

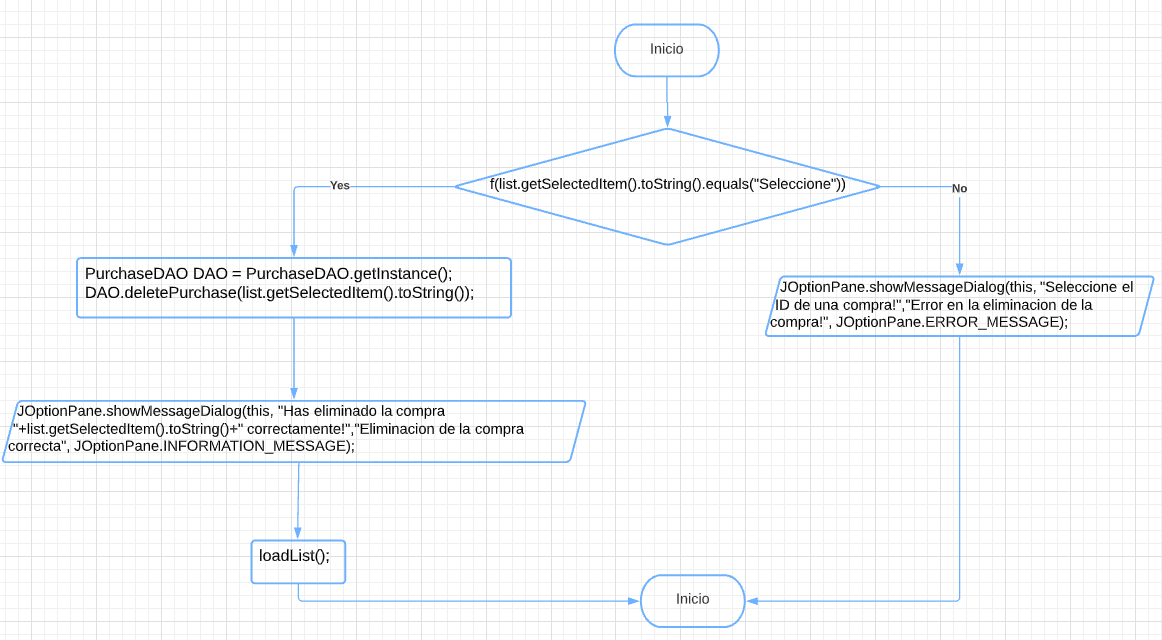
**R2:** 1, 2, 5, 6

**COMPLEJIDAD CICLOMÁTICA V(G)=** 6 – 6 + 2 = 2

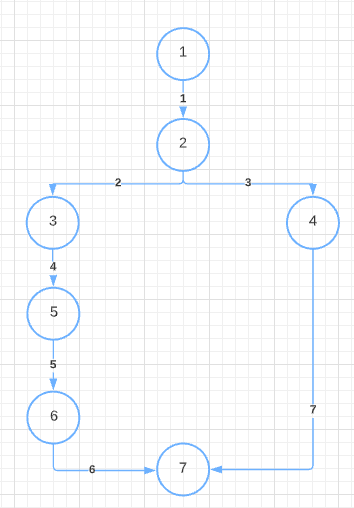
## Eliminar compra

|  |
| --- |
| if(list.getSelectedItem().toString().equals("Seleccione")){  JOptionPane.showMessageDialog(this, "Seleccione el ID de una compra!","Error en la eliminacion de la compra!", JOptionPane.ERROR\_MESSAGE);  } else {  PurchaseDAO DAO = PurchaseDAO.getInstance();  DAO.deletePurchase(list.getSelectedItem().toString());  JOptionPane.showMessageDialog(this, "Has eliminado la compra "+list.getSelectedItem().toString()+" correctamente!","Eliminacion de la compra correcta", JOptionPane.INFORMATION\_MESSAGE);  loadList();  } |

**Diagrama de flujo**



**Grafo**

****

**RUTAS**

**R1:** 1,3,7

**R2:** 1, 2, 4, 5, 6

**COMPLEJIDAD CICLOMÁTICA V(G)=** 7 – 7 + 2 = 2

## Buscar compra por id

SupplierDAO sDAO = SupplierDAO.getInstance();

PurchaseDAO DAO = PurchaseDAO.getInstance();

ProductDAO pDAO = ProductDAO.getInstance();

ArrayList<Purchase> purchases = DAO.getPurchases();

DefaultTableModel dtm = new DefaultTableModel();

dtm.setColumnIdentifiers(

new String[]{"Nombre del Proveedor", "Fecha de compra", "Producto/s", "Precio total"}

);

for (int i = 0; i < purchases.size(); i++) {

Integer purchasesSize = DAO.getPurchases().size();

Integer purchaseCode = DAO.getPurchases().get(i).getIdPurchase();

ArrayList<Integer> productCodes = DAO.getProductCodesByPurchaseCode(purchaseCode);

ArrayList<String> productNames = pDAO.getProductNamesByCodes(productCodes);

String supplierName = sDAO.getNameByCode(purchases.get(i).getIdSupplier());

if (this.listSuppliers.getSelectedItem().toString().equals(supplierName)) {

dtm.addRow(

new Object[]{

supplierName,

purchases.get(i).getDatePurchase(),

productNames.toString().replace("[", "").replace("]", ""),

purchases.get(i).getTotalPrice()

}

);

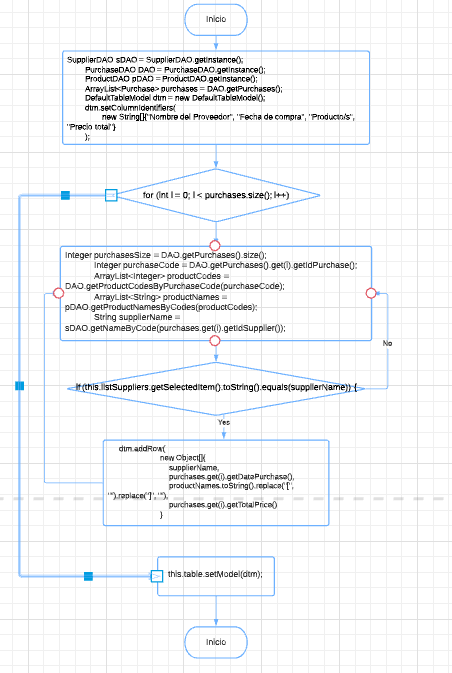
}

}

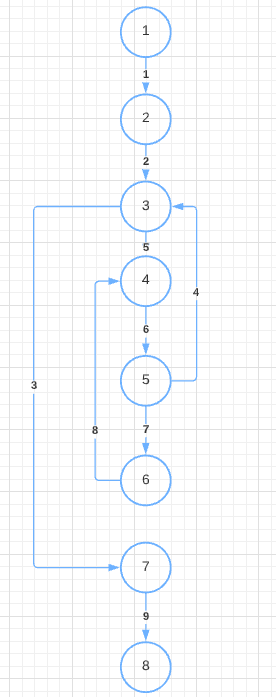
this.table.setModel(dtm);

}

**Diagrama de flujo**

****

**Grafo**

****

**RUTAS**

**R1:** 1,2,5,6,4,3,9

**R2:** 1, 2, 5, 67,8,4,3,9

**COMPLEJIDAD CICLOMÁTICA V(G)=** 9 – 8 + 2 = 3