Nombre:

kevin japa

Fecha de entrega:

8 de junio de 2020.

Tema:

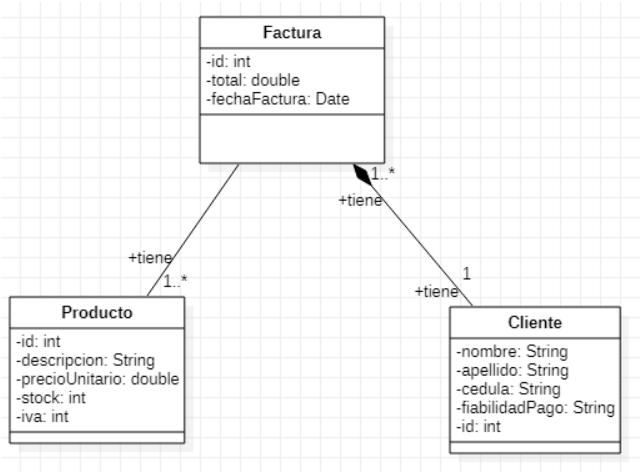
Examen Interciclo (sustentación).

Materia:

Programación Orientada a Objetos.

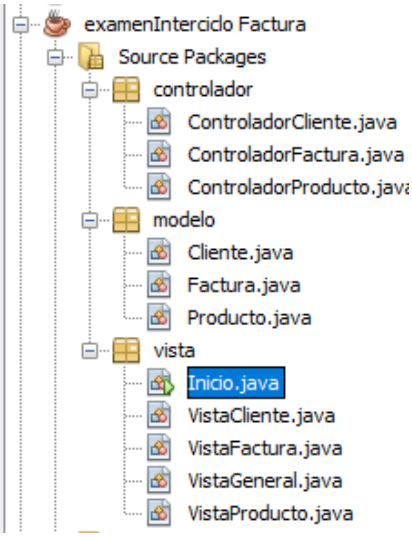
Diagrama UML

Inicio con la realización del diagrama UML en el cual tengo mi clase producto que tiene una asociación Unidireccional de uno a muchos con mi clase factura, en la clase Cliente tengo una relación de composición de en el cual mi factura contiene un solo cliente y mi cliente puede tener muchas facturas



MVC

Para realizar el sistema de facturación en la cual se implementara el uso del patrón de Arquitectura MVC en el paquete de modelo ingresa las clases principal modelo (Factura, Cliente, Producto) para el paquete de controlador ingresa mis clases(Controlador Factura, ControladorCliente, ControladorProducto) y para el paquete de vista ingresa la parte grafica que interactúa con el usuario(VistaCliente, VistaFactura, VistaGeneral, Vista,Producto) en este caso incluso lo que es la clase inicio ya que es la clase principal en la que se instancia la clase de VistaGeneral



Clases Cliente

Declaro mis variables(nombre, apellido, cedula, fiabilidadPago, id) en la cual son de tipo privadas para acceder a las mismas las mismas hago el uso de los encapsuladores GETTERS AND SETTERS y realizando constructor con un paso de parámetros e inicializar las variables y la implementación del método toString() para la impresión del objeto Cliente.

```
public class Cliente
{
    private int id;
    private String nombre;
    private String cedula;
    private String fiabilidadPago;

    public Cliente(int id,String nombre, String cedula,String fiabilidadPago)
    {
        this.id=id;
        this.nombre=nombre;
        this.cedula=cedula;
        this.fiabilidadPago=fiabilidadPago;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
```

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}

public String getFiabilidadPago() {
    return fiabilidadPago;
}

public void setFiabilidadPago(String fiabilidadPago) {
    this.fiabilidadPago = fiabilidadPago;
}

@Override
public String toString() {
    return "" + "id=" + id + ", nombre=" + nombre + ", cedula=" + cedula + ", fiabilidadPago=" + fiabilidadPago + '}';
}
```

Clase Factura

La implementación de la declaración de variables de uso privadas(id, total, fechaFactura, cliente, listaProducto) en la cual la variable de cliente de tipo cliente en la cual me permite almacenar los datos del cliente e implementar la clase Date para la fecha de la factura e incluyendo una lista de tipo producto para poder almacenar los productos en un arreglo y la implementación de constructores para inicializar las variables y para acceder a las variables de tipo privado el uso de los GETTERS AND SETTERS y el uso del toString() para imprimir el objeto.

```
blic class Factura

  private int id;
  private double total;
  private  Date fechaFactura ;
  private Cliente cliente;
  private List<Producto> listaProducto;
  public Factura(int id, double total, Date fechaFactura)
  {
      this.id=id;
     this.total=total;
      this.fechaFactura=fechaFactura;
  }

  public Factura(int id, double total, Date fechaFactura, Cliente cliente, List<Producto> listaProducto) {
      this.id = id;
      this.total = total;
      this.fechaFactura = fechaFactura;
      this.cliente = cliente;
      this.listaProducto = listaProducto;
  }


  public int getId() {
      return id;
  }

  public void setId(int id) {
      this.id = id;
```

```
ublic class Factura

    private int id;
    private double total;
    private  Date fechaFactura ;
    private Cliente cliente;
    private List<Producto> listaProducto;
    public Factura(int id, double total, Date fechaFactura)
    {
        this.id=id;
       this.total=total;
        this.fechaFactura=fechaFactura;
    }

    public Factura(int id, double total, Date fechaFactura, Cliente cliente, List<Producto> listaProducto) {
        this.id = id;
        this.total = total;
        this.fechaFactura = fechaFactura;
        this.cliente = cliente;
        this.listaProducto = listaProducto;
    }


    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
```

Clase Controlador Cliente

Para la implementación de MVC con respecto a la clase cliente esta nos permite controlar la información de los clientes en la cual implementa una lista de clientes de tipo clientes que se almacena en un ArrayList.

Se hace la implementación de CRUD en el cual el método crear en la cual nos permite crear clientes y almacenarlos en una lista de clientes en la cual se lo realiza por el paso de parámetros atreves del constructor a la clase persona para el método buscar nos permite buscar un cliente por su cedula que es tipo String en las listaCliente para el método de actualizar nos permite actualizar datos en el cual buscamos por la cedula ya que esta no puede ser actualizada y es única de cada cliente los datos de son actualizados a través del uso de los

encapsuladores (getters and setters) y son actualizados en la listaCliente para la eliminación de un cliente solamente se los busca por la cedula y se los remueve de la lista Cliente con la extencion de remove se realiza un método extra de listar en la cual solamente se rrecorre el arraylist de listacliente

```
ublic class ControladorCliente

    List<Cliente> listaCliente   = new ArrayList();
    Cliente seleccionado;

    public ControladorCliente()
    {


    }

     public int generarId()
    {
        return (listaCliente.size()>0)? listaCliente.get(listaCliente.size()-1).getId()+1:1;
    }
    public void crear(String nombre,String cedula, String fiabilidadPago)
    {

        Cliente cliente=new Cliente(generarId(),nombre,cedula,fiabilidadPago);
        listaCliente.add(cliente);

    }
    public Cliente buscar(String cedula)
    {
        for(Cliente cliente: listaCliente)
        {
            if(cliente.getCedula().equals(cedula))
            {
                seleccionado= cliente;
                return cliente;
```

.

```
public boolean actualizar(String nombre,String cedula, String fiabilidadPago)
{
    Cliente cliente=buscar(cedula);
    if(cliente!=null)
    {
        int posicion=listaCliente.indexOf(cliente);
        cliente.setNombre(nombre);
        cliente.setFiabilidadPago(fiabilidadPago);
        listaCliente.set(posicion, cliente);
        return true;
    }
    return false;
}
public boolean eliminar(String cedula)
{
    Cliente cliente=buscar(cedula);
    return listaCliente.remove(cliente);
}
public void listar()

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

    {

        System.out.println("Cliente "+(i+1)+":"+listaCliente.get(i));
    }
}
```

Controlador Producto

Se implementó el DRUD para la clase de producto se implementa una lista de producto de tipo producto con el uso del ArrayList se implementa el método crear se instancia la clase de vistaProducto en la cual se instancia la clase producto realizando el paso de parámetros a través del constructor pero se realiza una consulta en el cual se determina si se desea calcular el valor del IVA o no calcularlo, para el método de buscar se busca los productos a través de la descripción en la lista privada de listaProducto asiendo el uso del get para obtener la información requerida, en método de actualizar nos permite la actualización de los datos: precio unitario el stock y el iva en el cual para obtener se realiza una búsqueda a través de la descripción y se actualiza los datos con el uso de los encapsuladores.

```java
public class ControladorProducto
{
   private List<Producto> listaProducto;
   private Producto seleccionado;
   double decimal,totalIva;

   public ControladorProducto()
   {
      listaProducto = new ArrayList();

   }
    public int generarId()
        {
            return (listaProducto.size()>0)? listaProducto.get(listaProducto.size()-1).getId()+1:1;
        }
        public void crear(String descripcion, double precioUnitario, int stock, int iva,String opcion)
        {

            if(opcion.equals("si"))
            {
                decimal=iva/100;
                totalIva=precioUnitario*decimal;
                System.out.println("Valor Iva:"+totalIva);
            }
            else if(opcion.equals("no"))
            {
                totalIva=iva;
                System.out.print("No se calculo el Iva:");
            }

            else System.out.print("La opcion ingresada es erronea");
```

```java
            else System.out.print("La opcion ingresada es erronea");

            Producto producto=new Producto(generarId(),descripcion, precioUnitario,stock,totalIva);
            listaProducto.add(producto);

        }
        public Producto buscar(String descripcion)
        {
        for(Producto producto: listaProducto)
        {
            if(producto.getDescripcion().equals(descripcion))
            {
                seleccionado= producto;
                return producto;
            }

        }
         return null;
}
public boolean actualizar(String descripcion, double precioUnitario, int stock, int iva)
{
    Producto producto=buscar(descripcion);
    if(producto!=null)
    {
        int posicion=listaProducto.indexOf(descripcion);
        //producto.setDescripcion(descripcion);
        producto.setIva(iva);
        producto.setPrecioUnitario(precioUnitario);
        producto.setStock(stock);

        listaProducto.set(posicion, producto);

            listaProducto.set(posicion, producto);
            return true;
        }
        return false;
    }
    public void eliminar(String nombre)
    {
        Producto producto=buscar(nombre);
        listaProducto.remove(producto);
    }
    public void listar()

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

        {

            System.out.println("factura "+(i+1)+":"+listaProducto.get(i));
        }
```

Controlador Factura

Se implemento el DRUD, se implementó la instancia d la clase DateFormat y método de SimpleDateFormat en la cual nos permite darle un formato a la fecha ingresada por el usuario, se creo una lista de factura en el cual contiene un constructor en el cual inicializa ArrayList para la listaFactura, para el método de crear en este caso el método crear recibe como tipos de dato total pero aquí varia a la de las otras ya se recibe un dato de tipo date para la fecha y una lista de producto para su lista y un cliente las cuales son variables que contiene clase como tipo de dato y que son enviadas por el constructor de la clase factura en este caso como

la clase factura tiene una relación de composición no almacena un nombre como tal en cambio almacena un objeto de tipo persona con la información del cliente y se hace la instancia por los getters de cliente para realizar una búsqueda en la lista cliente por su cedula, para el uso de la actualización de datos funciona de igual manera se envía una cedula al método de buscar por medio de la Factura y este nos permite cambiar la fecha, total y la fecha para el método de eliminar se busca al cliente por la cedula llamando al método buscar y se utiliza el método remove para eliminar de la lista Factura

```
public class ControladorFactura {
    DateFormat formatoFecha=new SimpleDateFormat("dd/mm/yyyy");
    private List<Factura> listaFactura;
    private Factura seleccionado;
    public ControladorFactura()
    {
        listaFactura = new ArrayList();

    }
        public int generarId()
        {
            return (listaFactura.size()>0)? listaFactura.get(listaFactura.size()-1).getId()+1:1;
        }
        public boolean crear(double total, Date fechaFactura)
        {
            Factura factura=new Factura(generarId(),total,fechaFactura);
            return listaFactura.add(factura);

        }
        public boolean crear(double total, Date fechaFactura,List<Producto> listaProducto,Cliente cliente)
        {
            Factura factura=new Factura(generarId(),total,fechaFactura,cliente,listaProducto);
            return listaFactura.add(factura);

        }
        public Factura buscar(String cedula)
        {
        for(Factura factura: listaFactura)
        {
```

```
    public Factura buscar(String cedula)
    {
    for(Factura factura: listaFactura)
    {
        if(factura.getCliente().getCedula().equals(cedula))
        {
            seleccionado= factura;
            return factura;
        }

    }
     return null;
}
public boolean actualizar(String cedula,double total, Date fechaFactura,List<Producto> listaProducto,Cliente cliente)
{
    Factura factura=buscar(cedula);
    if(factura!=null)
    {
        int posicion=listaFactura.indexOf(factura);
        factura.setFechaFactura(fechaFactura);
        factura.setTotal(total);
        factura.setCliente(cliente);
        factura.setListaProducto(listaProducto);
        listaFactura.set(posicion, factura);
        return true;
    }
    return false;
}
public boolean actualizar(String cedula,double total,Date fechaFactura)
{
    Factura factura=buscar(cedula);

        int posicion=listaFactura.indexOf(factura);
        factura.setFechaFactura(fechaFactura);
        factura.setTotal(total);
        listaFactura.set(posicion, factura);
        return true;
    }
    return false;
}
public void eliminar(String cedula)
{
    Factura factura=buscar(cedula);
    listaFactura.remove(factura);
}
public void listar()

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

    {

        System.out.println("factura "+(i+1)+":"+listaFactura.get(i));
    }
}

public List<Factura> getListaFactura() {
    return listaFactura;
}

public void setListaFactura(List<Factura> listaFactura) {
    this.listaFactura = listaFactura;
}
```

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

    {

        System.out.println("factura "+(i+1)+":"+listaFactura.get(i));
    }
}

public List<Factura> getListaFactura() {
    return listaFactura;
}

public void setListaFactura(List<Factura> listaFactura) {
    this.listaFactura = listaFactura;
}

public Factura getSeleccionado() {
    return seleccionado;
}

public void setSeleccionado(Factura seleccionado) {
    this.seleccionado = seleccionado;
}
```

Vista Producto

En la implementación de MVC nos permite que las vista se mantenga por separado en la cual nos permite recibir los datos ingresados por el usuario que son enviados al controlador Producto en esta clase se realiza la instancia de la clase del controlador y el uso de los métodos genéricos del Scanner que nos permite leer el teclado se procede a crear un método de vista producto en el cual se utiliza el bucle if para realizar el menú de opciones como métodos secundarios se realiza 4 métodos (crear, listar, actualizar, borrar) para recibir datos ingresado por el usuario las cuales estos métodos son llamados en el método de menú principal según la opción que ingrese al usuario.

```
public class VistaProducto
{
    //VistaFactura vistaFactura=new VistaFactura();
    ControladorProducto controladorProducto=new ControladorProducto();
    Scanner teclado=new Scanner(System.in);
    public void menuProducto()
    {
        int op=1;
        while(op>0)
        {
            System.out.println("-----------------------------------------------------");
            System.out.println("--Menu Cliente-- \n0.Salir \n1.Crear \n2.buscar \n3.Actualizar \n4.eliminar \n5.listar \n5.CrearF
            System.out.print("Ingrese una opcion:");
            op=teclado.nextInt();
            if(op==1)
            {
                crear();
            }
            else if(op==2)
            {
                buscar();
            }
            else if(op==3)
            {
                actualizar();
            }
            else if(op==4)
            {
                eliminar();
```

```
        }
        else if(op==5)
        {
            controladorProducto.listar();
        }
        else if(op==6)
        {
            //vistaFactura.menuFactura();
        }
    }

}
public void crear()
{
    //Producto producto=crear();
    /*System.out.print("Ingrese el nombre:");

    String nombre=teclado.next();*/
    System.out.print("Ingrese la descripcion:");
    teclado.next();
    String descripcion=teclado.nextLine();
    System.out.print("Ingrese el precio Unitario:");
    double precioUnitario=teclado.nextDouble();
    System.out.print("Ingrese el Stock:");
    int stock=teclado.nextInt();
    System.out.print("Ingrese el iva:");
    int iva=teclado.nextInt();
    System.out.println("El producto tiene iva:si/no");
    teclado.nextLine();
    String opcion=teclado.nextLine();

    controladorProducto.crear(descripcion, precioUnitario,stock,iva,opcion);
    //return producto;
}
public void buscar()
{
    System.out.print("Ingrese el nombre:");
    teclado.nextLine();
    String nombre=teclado.nextLine();
    controladorProducto.buscar(nombre);
}
public void actualizar()
{
    System.out.print("Ingrese la Descripcon:");
        String descripcion=teclado.next();
    if(controladorProducto.buscar(descripcion)==null)
    {
        System.out.print("Ingrese un Cliente existente");
    }
    else
    {
        System.out.print("Ingrese el iva:");
        int iva=teclado.nextInt();
        System.out.println("Ingrese el Precio Unitario:");
        double precioUnitario = teclado.nextDouble();
        System.out.println("Ingrese el Stock:");
        int stock=teclado.nextInt();
        controladorProducto.actualizar(descripcion,precioUnitario,stock, iva);
    }

}

}
public void eliminar()
{
    System.out.print("Ingrese el nombre:");
    String nombre=teclado.next();
    controladorProducto.eliminar(nombre);
}
```

Vista Cliente

En la parte grafica de cliente por consola instancio la clase de controlador, pero inicializado en el constructor en el cual para poder acceder a una opción que ingrese el usuario se implementa el método de menú cliente en el cual se llama a los métodos de crear, actualizar, listar, eliminar somo métodos para ingresar datos que son llamados en el método de menú cliente son los de crear, actualizar, listar, eliminar en el cual ingresa el usuario los datos y son enviados por los métodos instanciados del controlador de cliente.

```java
public class VistaCliente
{
    //VistaFactura vistaFactura=new VistaFactura();
    private ControladorCliente controladorCliente;
    Scanner teclado=new Scanner(System.in);
    public VistaCliente()
    {
        controladorCliente=new  ControladorCliente();
    }
    public void menuCliente()
    {
        int op=1;
        while(op>0)
        {
            System.out.println("-----------------------------------------------------");
            System.out.println("--Menu Cliente-- \n0.Salir \n1.Crear \n2.buscar \n3.Actualizar \n4.eliminar \n5.listar \n5.Crear ");
            System.out.print("Ingrese una opcion:");
            op=teclado.nextInt();
            if(op==1)
            {
                crear();

            }
            else if(op==2)
            {
                buscar();
            }
            else if(op==3)
            {
                actualizar();
            }
            else if(op==4)
            {
                eliminar();
            }
            else if(op==5)
            {
                controladorCliente.listar();
            }
            else if(op==6)
                    {
                        // vistaFactura.menuFactura();
                    }
        }

    }
    public void crear()
    {


        System.out.print("Ingrese el nombre:");
        teclado.nextLine();
        String nombre=teclado.nextLine();
        System.out.print("Ingrese la cedula:");
        String cedula=teclado.nextLine();
        System.out.print("Ingrese la fiabilidad de pago:");
        String fiabilidadPago=teclado.nextLine();


    }
    public void buscar()
    {
        System.out.print("Ingrese la cedula:");
        teclado.nextLine();
        String cedula=teclado.nextLine();
        controladorCliente.buscar(cedula);
    }
    public void actualizar()
    {
        System.out.print("ingrese la cedula");
        String cedula=teclado.next();
        if(controladorCliente.buscar(cedula)==null)
        {
            System.out.print("Ingrese el numero existente");
        }
        else
        {
            System.out.print("Ingrese el nombre:");
            String nombre=teclado.next();
            System.out.print("Ingrese la fiabilidad de Pago:");
            String pago=teclado.next();
            controladorCliente.actualizar(nombre, cedula, pago);
        }

    }

    public void eliminar()
    {
        System.out.print("Ingrese la cedula:");
        String cedula=teclado.next();
        controladorCliente.eliminar(cedula);
    }
```

Vista Factura

En la clase de vista factura se instancia la clase ya creada DateFormat y la instancia de SimpleDateFormat("dd/mm/yyyy") que nos permite dar un formato a la fecha ingresada, se realiza la instancia las clases controlador factura, controlador cliente, controlador producto y

la vista Cliente en el cual estas clases son inicializadas en el constructor y se crea un método de menú factura en el cual este método no devuelve ningún valor en el cual llama a los 4 métodos creados en la que se llaman según la opción ingresada por el usuario en el cual en este caso en el método crear se utiliza controlador de errores try-catch para el ingreso correcto de la fecha según el formato indicado en el cual el usuario ingresa la cedula y esta es enviada al método buscar nos informa si el usuario existe o no en la base de datos en la que se instancia la clase buscar y crear del controlador factura, esta estructura integrando el try- catch también son implementados en el método de actualizar.

```
public class VistaFactura {
    DateFormat formatoFecha=new SimpleDateFormat("dd/mm/yyyy");
    private ControladorFactura controladorFactura;
    Scanner teclado=new Scanner(System.in);
    private ControladorCliente controladorCliente;
    private ControladorProducto controladorProducto;
    VistaCliente vistaCliente=new  VistaCliente();
    List<Producto> listaProducto=new ArrayList();
    public VistaFactura()
    {
        controladorFactura=new ControladorFactura();
        controladorCliente=new ControladorCliente();
        controladorProducto= new ControladorProducto();
    }
    public void menuFactura()
    {
        int op=1;
        while(op>0)
        {
            System.out.println("-----------------------------------------------------");
            System.out.println("--Menu Factura-- \n0.Salir \n1.Crear \n2.buscar \n3.Actualizar \n4.eliminar \n5.listar");
            System.out.print("Ingrese una opcion:");
            op=teclado.nextInt();
            if(op==1)
            {
                crear();
            }
            else if(op==2)
            {

        else if(op==3)
        {
            actualizar();
        }
        else if(op==4)
        {
            eliminar();
        }
        else if(op==5)
        {
            controladorFactura.listar();
        }
    }
}
public void crear()
{
    try {

        System.out.print("Ingrese la cedula:");
        teclado.next();
        String cedula=teclado.nextLine();
        System.out.print("Ingrese nombre del producto:");
        String nombre=teclado.nextLine();
        Cliente cliente=controladorCliente.buscar(cedula);
        System.out.print("Ingrese el total:");
        double total=teclado.nextDouble();
        System.out.print("ingrese la fecha(\"dd/mm/yyyy\"):");
        String fecha=teclado.next();
        controladorFactura.crear(total,formatoFecha.parse(fecha));
        if(cliente!=null)
```

```
        String fecha=teclado.next();
        controladorFactura.crear(total,formatoFecha.parse(fecha));
        if(cliente!=null)
        {
            Producto producto=controladorProducto.buscar(nombre);
            if(producto!=null)
            {
                listaProducto.add(producto);
                controladorFactura.actualizar(cedula,total, formatoFecha.parse(fecha), listaProducto, cliente);

            }
            else
            {
                System.out.println("El producto ingresado no existe");

            }
        }
        else
        {

            System.out.println("El cliente Ingresado no existe");
            System.out.println("--Crear Cliente--");
            vistaCliente.menuCliente();

        }

    } catch (ParseException ex) {
        System.out.println(ex.getMessage());
    }
}
```

```
public void buscar()
{
    System.out.print("Ingrese la cedula:");
    String cedula=teclado.nextLine();
    controladorFactura.buscar(cedula);
}
public void actualizar()
{
    try {
       teclado.nextLine();
       System.out.print("Ingrese la cedula:");
       String cedula=teclado.nextLine();
       System.out.print("ingrese la fecha(\"dd/mm/yyyy\"):");
       String fecha=teclado.next();
       System.out.println("Ingrese un total:");
       double total=teclado.nextDouble();
       controladorFactura.actualizar(cedula,total,formatoFecha.parse(fecha));
    } catch (ParseException ex) {
       System.out.println(ex.getMessage());
    }
}
public void eliminar()
{
    System.out.print("Ingrese la cedula:");
    String cedula=teclado.nextLine();
    controladorFactura.eliminar(cedula);
}
```

Vista General

Esta clase nos permite unir todas las vistas creadas cliente, factura, producto en la que se instancia las vistas de dichas clases contenidas dentro del método menú general y son llamadas según la opción del usuario y esta clase de vista general es instancia y llamada en la clase principal la cual es la primera que se ejecuta.

```
VistaProducto vistaProducto=new VistaProducto();
VistaFactura vistaFactura=new VistaFactura();
VistaCliente vistaCliente=new VistaCliente();
Scanner teclado=new Scanner(System.in);

public void menuPrincipal()
{
    int op=1;
    while(op>0)
    {
        System.out.println("-----------------------------------------------------");
        System.out.println("--Menu General-- \n0.Salir \n1.Cliente \n2.Factura \n3.Producto");
        System.out.print("Ingrese una opcion:");
        op=teclado.nextInt();
        if(op==1)
        {
            vistaCliente.menuCliente();
        }
        else if(op==2)
        {
            vistaFactura.menuFactura();
        }
        else if(op==3)
        {
            vistaProducto.menuProducto();
        }

    }
}
```

```
public class Inicio {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        VistaGeneral vistaGeneral=new VistaGeneral();
        vistaGeneral.menuPrincipal();
    }

}
```