
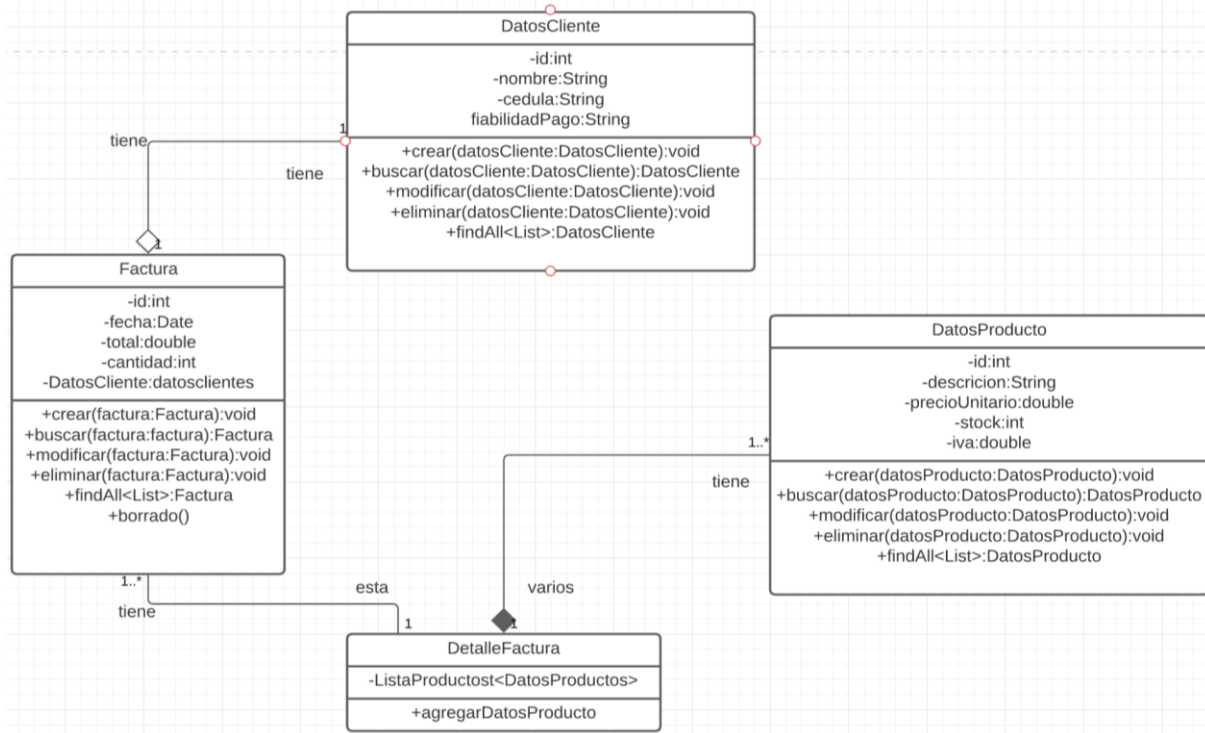
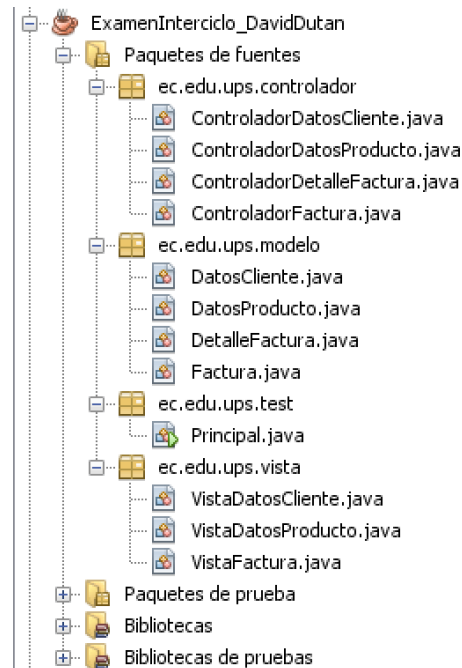
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		Examen Interciclo
CARRERA: Computación		ASIGNATURA: Programación orientada a Objetos
OBJETIVO ALCANZADO: Desarrollar un programa de consola utilizando MVC, además a realizar y entender requerimientos pedidos por el docente con un diagrama UML de las clases. Practicar las buenas normas de la Programación con sus métodos, constructores, encapsulamientos.		
ACTIVIDADES DESARROLLADAS		
1. Realizar un Diagrama UML con el siguiente anunciado: <u>Enunciado:</u> <p>En un programa de ordenador, las facturas tienen necesariamente un conjunto de datos de los productos de los cuales comprar y un conjunto de datos del cliente, un total (valor decimal) y una fecha de la factura.</p> <p>Los datos del cliente son la cadena de caracteres nombre, cedula, y el entero fiabilidad de pago, mientras que los datos del detalle de la factura son sólo su producto, cantidad, valor.</p> <p>Cada clase tendrá los métodos para leer y fijar (“set” y “get”) todos sus atributos.</p> <p>También se debe incluir en la clase Factura un método de “Borrado”, que no devuelve ni recibe ningún parámetro.</p> <p>Los productos que venden la empresa son de carácter de computadores y tiene un id, descripción, precio unitario, stock, IVA (si se calcula el IVA o no).</p> <p>Los atributos serán privados y tendrán métodos públicos para acceder a ellos. Se pide dibujar el diagrama UML de las clases Factura, Datos_del_cliente y Datos_del_producto y generar el sistema en java para realizar los procesos de CRUD para generar facturas, productos, clientes.</p> <u>Diagrama UML</u>		



2. Crear un proyecto en NetBeans con sus respectivos paquetes y clases:



En el siguiente programa tenemos 4 paquetes los paquetes respectivos para el modelo, vista, controlador y test.

Cada uno de ellos con sus respectivas clases y la clase principal donde se ejecutara e programa.

3. Explicación código de las clases del paquete modelo.

Clase Datos Cliente:

```
public class DatosCliente {

    private int id;
    private String nombre;
    private String cedula;
    private String fialidadPago;

    public DatosCliente(int id, String nombre, String cedula, String fialidadPago) {
        this.id = id;
        this.nombre = nombre;
        this.cedula = cedula;
        this.fialidadPago = fialidadPago;
    }

    public DatosCliente() {
    }

    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 String getCedula() {
        return cedula;
    }

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

    public String getFialidadPago() {
        return fialidadPago;
    }

    public void setFialidadPago(String fialidadPago) {
        this.fialidadPago = fialidadPago;
    }

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

Clase DatosProducto

```
public class DatosProducto {

    private int codigo;
    private String descripcion;
    private double precioUnitario;
    private int stock;
    private double iva;

    public DatosProducto(int codigo, String descripcion, double precioUnitario, int stock, double iva) {
        this.codigo = codigo;
        this.descripcion = descripcion;
        this.precioUnitario = precioUnitario;
        this.stock = stock;
        this.iva = iva;
    }

    public DatosProducto() {
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public double getPrecioUnitario() {
        return precioUnitario;
    }

    public void setPrecioUnitario(double precioUnitario) {
        this.precioUnitario = precioUnitario;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }

    public double getIva() {
        return iva;
    }

    public void setIva(double iva) {
        this.iva = iva;
    }

    @Override
    public String toString() {
        return "DatosProducto(" + "codigo=" + codigo + ", descripcion=" + descripcion + ", precioUnitario=" + precioUnitario + ", stock=" + stock + ")";
    }
}
```

Clase Factura

```
public class Factura {  
  
    private int numero;  
    private Date fecha;  
    private double total;  
    private int cantidad;  
    private List<DatosCliente> datosClientes;  
  
    public Factura(int id, Date fecha, double total, int cantidad) {  
        this.numero = id;  
        this.fecha = fecha;  
        this.total = total;  
        this.cantidad = cantidad;  
    }  
  
    public Factura() {  
        datosClientes = new ArrayList<>();  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public void setNumero(int id) {  
        this.numero = numero;  
    }  
  
    public Date getFecha() {  
        return fecha;  
    }  
  
    public void setFecha(Date fecha) {  
        this.fecha = fecha;  
    }  
  
    public double getTotal() {  
        return total;  
    }  
  
    public void setTotal(double total) {  
        this.total = total;  
    }  
  
    public int getCantidad() {  
        return cantidad;  
    }  
  
    public void setCantidad(int cantidad) {  
        this.cantidad = cantidad;  
    }  
  
    public void agregarDatosCliente(DatosCliente datosCliente) {  
        datosClientes.add(datosCliente);  
    }  
  
    public List<DatosCliente> imprimirDatosClientes() {  
        return datosClientes;  
    }  
  
    @Override  
    public String toString() {  
        return "Factura[" + "numero=" + numero + ", fecha=" + fecha + ", total=" + total + ", cantidad=" + cantidad + '']';  
    }  
  
    public void borrador() {  
  
    }  
}
```

Clase DetalleFactura:

```
public class DetalleFactura {  
    private List<DatosProducto> datosProductos;  
  
    public DetalleFactura(List<DatosProducto> datosProductos) {  
        this.datosProductos = datosProductos;  
    }  
  
    public DetalleFactura() {  
        datosProductos = new ArrayList<>();  
    }  
  
    public void agregarDatosProductos(DatosProducto datosProducto) {  
        datosProductos.add(datosProducto);  
    }  
  
    public List<DatosProducto> imprimirDatosProductos() {  
        return datosProductos;  
    }  
  
    @Override  
    public String toString() {  
        return "DetalleFactura(" + "datosProductos=" + datosProductos + ')';  
    }  
}
```

Como podemos ver en todas las clases tenemos nuestros atributos de tipo privados, además sus respectivos constructores con sus atributos inicializados, su respectivo encapsulamiento con los métodos getters y setters donde extrae y muestra la información de cada uno de sus atributos y por último sus métodos toString donde tiene toda la información de cada uno de los objetos creados.

4.Explicacion de las clases del paquete controlador.

Clase ControladorDatosCliente:

```
public class ControladorDatosCliente {

    private DatosCliente datosCliente;
    private Map<Integer, DatosCliente> diccionario;

    public ControladorDatosCliente(DatosCliente datosCliente) {
        this.datosCliente = datosCliente;
        this.diccionario = new HashMap<Integer, DatosCliente>();
    }

    public ControladorDatosCliente() {
    }

    public void crear(int id, String nombre, String Cedula, String fialidadPago) {
        datosCliente = new DatosCliente(id, nombre, Cedula, fialidadPago);
        diccionario.put(id, datosCliente);
    }

    public DatosCliente buscar(String id) {
        if (datosCliente != null) {
            return datosCliente;
        } else {
            System.out.println("El cliente no existe");
            return null;
        }
    }

    public boolean modificar(int id, String nombre, String Cedula, String fialidadPago) {
        if (datosCliente != null) {
            // datosCliente=buscar(id);
            datosCliente.setNombre(nombre);
            datosCliente.setCedula(Cedula);
            datosCliente.setFialidadPago(fialidadPago);
            diccionario.replace(id, datosCliente);
            return true;
        } else {
            return false;
        }
    }

    public boolean eliminar(int id) {
        if (buscar(String.valueOf(id)) != null) {
            datosCliente = new DatosCliente(id, null, null, null);
            diccionario.remove(id);
            return true;
        } else {
            return false;
        }
    }

    public Collection<DatosCliente> findAll() {
        Collection<DatosCliente> datosClientes = diccionario.values();
        return datosClientes;
    }
}
```

Clase ControladorDatosProducto:

```
public class ControladorDatosProducto {

    private DatosProducto datosProducto;
    private Map<Integer, DatosProducto> diccionario;

    public ControladorDatosProducto(DatosProducto datosProducto) {
        this.datosProducto = datosProducto;
        this.diccionario = new HashMap<Integer, DatosProducto>();
    }

    public ControladorDatosProducto() {
    }

    public void crear(int codigo, String descripcion, double precioUnitario, int stock, double iva) {

        datosProducto = new DatosProducto(codigo, descripcion, precioUnitario, stock, iva);
        diccionario.put(codigo, datosProducto);

    }

    public DatosProducto buscar(int codigo) {
        if (datosProducto != null) {
            return datosProducto;
        } else {
            System.out.println("El Producto no existe");
        }
        return null;
    }

    public boolean modificar(int codigo, String descripcion, double precioUnitario, int stock, double iva) {
        if (buscar(codigo) != null) {
            // datosProducto=buscar(codigo);
            datosProducto.setDescripcion(descripcion);
            datosProducto.setPrecioUnitario(precioUnitario);
            datosProducto.setStock(stock);
            datosProducto.setIva(iva);
            //datosProducto = new DatosProducto(codigo, descripcion, precioUnitario, stock, iva);
            diccionario.replace(codigo, datosProducto);
            return true;
        } else {
            return false;
        }

        return false;
    }

}

public boolean eliminar(int codigo) {
    if(buscar(codigo) != null){
        datosProducto = new DatosProducto(codigo, null, 0.00, 0, 0.00);
        diccionario.remove(codigo);
        return true;
    }else{
        return false;
    }
}

public Collection<DatosProducto> findAll() {
    Collection<DatosProducto> datosProductos = diccionario.values();
    return datosProductos;
}
```


Clase ControladorFactura:

```
public class ControladorFactura {  
  
    private DatosCliente datosCliente;  
    private DatosProducto datosProducto;  
    private DetalleFactura detalleFactura;  
    private Factura factura;  
    private Map<Integer, Factura> diccionario;  
    private ControladorDatosCliente controladorDatosCliente;  
  
    public ControladorFactura(DatosCliente datosCliente, DatosProducto datosProducto, DetalleFactura detalleFactura, Factura factura,  
        this.datosCliente = datosCliente;  
        this.datosProducto = datosProducto;  
        this.detalleFactura = detalleFactura;  
        this.factura = factura;  
        this.diccionario = new HashMap<>();  
        this.controladorDatosCliente = controladorDatosCliente;  
    }  
  
    public ControladorFactura() {  
    }  
  
    public boolean create(int numero, Date fecha, double total, int cantidad, int datosClientes) {  
        if (controladorDatosCliente.buscar(String.valueOf(datosClientes)) != null) {  
            factura = new Factura(numero, fecha, total, cantidad);  
            // Set<DatosCliente> clientes1 = new HashSet<>();  
            //for (DatosCliente cliente : clientes1) {  
            //    factura.agregarDatosCliente(datosCliente);  
            //clientes1.add(cliente);  
            diccionario.put(numero, factura);  
  
            return true;  
            // }  
        }  
        return false;  
    }  
  
    public Factura buscar(int codigo) {  
        if (factura != null) {  
            return factura;  
        } else {  
            System.out.println("la factura no existe");  
        }  
    }  
  
    public boolean eliminar(int codigo) {  
        if (buscar(codigo) != null) {  
            factura = new Factura(codigo, null, 00.0, 0);  
            diccionario.clear();  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public Collection<Factura> findALL() {  
        Collection<Factura> facturas = diccionario.values();  
        return facturas;  
    }  
}
```

Clase ControladorDetalleFactura:

```
public class ControladorDetalleFactura {
    private ArrayList<Integer> diccionario;
    private DetalleFactura detalleFactura;
    private Factura factura;

    public ControladorDetalleFactura(DetalleFactura detalleFactura, Factura factura) {
        this.diccionario = new ArrayList<>();
        this.detalleFactura = detalleFactura;
        this.factura = factura;
    }

    public ControladorDetalleFactura() {
    }

    public void aniadirFactura(Factura factura){
        this.factura= factura;
    }

    public void aniadirLista(List<Integer>codigoProducto){
        for (int i = 0; i < codigoProducto.size(); i++) {
            diccionario.add(codigoProducto.get(i));
        }
    }

    public List<Integer> MostrarLista(){
        return diccionario;
    }
}
```

En todos los constructores importamos el modelo de su respectiva clase y creamos un objeto de ella, además implementamos los métodos CRUD y lo realizamos de acuerdo a cada clase de lo que se requiera crear con los atributos de cada una de ellas, además aquí implementamos el método findAll que me guarda toda la información en un diccionario que creamos anteriormente en la de la clase HasMap y ArrayList.

Cada constructor tiene inicializadas sus atributos de las clases importadas.

5. Demostración de las clases del paquete vista.

Clase vistaDatosClientes:

```
public class VistaDatosCliente {
    private ControladorDatosCliente controladorDatosCliente;
    Scanner scanner = new Scanner(System.in);

    public VistaDatosCliente(ControladorDatosCliente controladorDatosCliente) {
        this.controladorDatosCliente = controladorDatosCliente;
    }

    public VistaDatosCliente() {
    }

    public void crearCliente() {
        System.out.println("Ingrese los datos del cliente: ");
        System.out.println(" elCodigo del cliente");
        int id = scanner.nextInt();
        System.out.println("Ingrese el nombre del Cliente: ");
        String nombre = scanner.next();
        System.out.println("Ingrese la cedula del cliente: ");
        String cedula = scanner.next();
        System.out.println("Ingrese la fialidad de pago del Cliente: ");
        String fialidadPago = scanner.next();
        controladorDatosCliente.crear(id, nombre, cedula, fialidadPago);
        System.out.println("-----Usuario Creado-----");
    }

    public VistaDatosCliente buscar() {
        System.out.println("Ingrese la ID del cliente a buscar");
        int id = scanner.nextInt();
        if (controladorDatosCliente.buscar(String.valueOf(id)) != null) {
            System.out.println(controladorDatosCliente.buscar(String.valueOf(id)).toString());
        } else {
            System.out.println("usuario no existe");
        }
        return null;
    }

    public void modificar() {
        System.out.println("Ingrese el codigo del cliente a Modificar");
        int id = scanner.nextInt();
        if (controladorDatosCliente.buscar(String.valueOf(id)) != null) {
            System.out.println("----Ingrese los nuevo datos del cliente----");
            System.out.println("Ingrese el nuevo nombre");
            String nombre = scanner.next();
            System.out.println("Ingrese la nueva cedula");
            String cedula = scanner.next();
            System.out.println("Ingrese fialidad de pago");
            String fialidadPago = scanner.next();

            controladorDatosCliente.modificar(id, nombre, cedula, fialidadPago);
            System.out.println("Cliente modificado");
        } else {
            System.out.println("usuario no existe");
        }
    }

    public void eliminar() {
        System.out.println("Ingrese el codigo del cliente a eliminar:");
        int id = scanner.nextInt();
        if (controladorDatosCliente.buscar(String.valueOf(id)) != null) {
            controladorDatosCliente.eliminar(id);
            System.out.println("Producto eliminado con exito");
        } else {
            System.out.println("Producto no existente");
        }
    }

    public void ListarClientes() {
        controladorDatosCliente.findAll().stream().forEach(object -> System.out.println(object));
    }
}
```

En esta vista importamos la clase Scanner para pedir al usuario que ingrese los datos por teclado en cada método CRUD pedimos que ingresen los datos de a cuerdo a lo que el usuario quiere hacer y le mandamos a que el controlador lo cree, busque, modifique o elimine además tenemos el método listar cliente donde los imprimen todos los datos clientes que existen en el diccionario creado.

Clase VistaDatosProducto:

```
public class VistaDatosProducto {

    ControladorDatosProducto controladorDatosProducto;
    Scanner scanner = new Scanner(System.in);

    public VistaDatosProducto(ControladorDatosProducto controladorDatosProducto) {
        this.controladorDatosProducto = controladorDatosProducto;
    }

    public VistaDatosProducto() {
    }

    public void crear() {
        System.out.println("Ingrese los datos del producto :");
        System.out.println("Ingrese el codigo del producto: ");
        int codigo = scanner.nextInt();
        System.out.println("Ingrese la descripcion del producto:");
        String descripcion = scanner.next();
        System.out.println("Ingrese el precio unitario del producto: ");
        double preciounitario = scanner.nextDouble();
        System.out.println("Ingrese el stock del producto: ");
        int stock = scanner.nextInt();
        System.out.println("Ingrese el iva del producto: ");
        double iva = scanner.nextDouble();

        controladorDatosProducto.crear(codigo, descripcion, preciounitario, stock, iva);
        System.out.println("Producto Creado");
    }

    public VistaDatosProducto buscar() {
        System.out.println("Ingrese el codigo del producto a buscar");
        int codigo = scanner.nextInt();
        if (controladorDatosProducto.buscar(codigo) != null) {
            System.out.println(controladorDatosProducto.buscar(codigo).toString());
        } else {
            System.out.println("El producto no existe");
        }
        return null;
    }

    public void modificar() {
        System.out.println("Ingrese el Codigo de producto a modificar: ");
        int codigo = scanner.nextInt();
        if (controladorDatosProducto.buscar(codigo) != null) {
            System.out.println("-----Ingrese los nuevos datos del producto-----");
            System.out.println("Ingrese la nueva descripcion del producto");
            String descripcion = scanner.next();
            System.out.println("Ingrese el nuevo precio");
            double preciounitario = scanner.nextDouble();
            System.out.println("Ingrese el nuevo stock");
            int stock = scanner.nextInt();
            System.out.println("Ingrese el nuevo iva del producto");
            double iva = scanner.nextDouble();

            controladorDatosProducto.modificar(codigo, descripcion, preciounitario, stock, iva);
            System.out.println("Producto modificado");
        } else {
            System.out.println("El producto no existe");
        }
    }

    public void eliminar() {
        System.out.println("Ingrese el codigo del producto a eliminar:");
        int codigo = scanner.nextInt();
        if (controladorDatosProducto.buscar(codigo) != null) {
            controladorDatosProducto.eliminar(codigo);
            System.out.println("Producto eliminado");
        } else {
            System.out.println("Producto no existente");
        }
    }

    public void listarProductos() {
        controladorDatosProducto.findAll().stream().forEach(object -> System.out.println(object));
    }
}
```

En esta vista importamos la clase Scanner para pedir al usuario que ingrese los datos por teclado en cada método CRUD pedimos que ingresen los datos de acuerdo a lo que el usuario quiere hacer y le mandamos a que el controlador lo cree, busque, modifique o elimine además tenemos el método listar cliente donde los imprimen todos los datos clientes que existen en el diccionario creado.

Clase vistaFactura:

```
public class VistaFactura {

    ControladorFactura controladorFactura;
    ControladorDatosProducto controladorDatosProducto;
    ControladorDetalleFactura controladorDetalleFactura;
    ControladorDatosCliente controladorDatosCliente;
    public DateFormat formatoFecha;
    List<Integer> datosProductos = new ArrayList<>();
    private double subtotal = 0;

    public VistaFactura(ControladorFactura controladorFactura, ControladorDatosProducto controladorDatosProducto, ControladorDetalleFactura
        this.controladorFactura = controladorFactura;
        this.controladorDatosProducto = controladorDatosProducto;
        this.controladorDetalleFactura = controladorDetalleFactura;
        this.controladorDatosCliente = controladorDatosCliente;
        formatoFecha = new SimpleDateFormat("dd/mm/yyyy");
    }

    Scanner input = new Scanner(System.in);
    boolean salir = false;
    int opcion4;

    public void crear() {
        System.out.println("Ingrese los Datos de la Factura: ");
        System.out.println("Ingrese numero de Factura: ");
        int numero = input.nextInt();
        System.out.println("Fecha (dd/mm/yyyy):");
        String fecha = input.next();
        while (!salir) {
            System.out.println(" 1.Agregar producto\n 2.Facturar\n 3.Salir");
            try {
                System.out.println("Elija una opcion");
                opcion4 = input.nextInt();

                switch (opcion4) {

                    case 1:
                        System.out.println("ingrese el codigo del producto para agregar a la factura");
                        int codigoProducto = input.nextInt();
                        if (controladorDatosProducto.buscar(codigoProducto) != null) {
                            datosProductos.add(codigoProducto);
                            subtotal += controladorDatosProducto.buscar(codigoProducto).getPrecioUnitario();
                            System.out.println("Producto agregado ala factura");
                        } else {
                            System.out.println("codigo de producto no existe");
                        }
                        break;
                    case 2:
                        System.out.println("Dijite el numero del Cliente");
                        int id = input.nextInt();
                        if (controladorDatosCliente.buscar(String.valueOf(id)) != null) {
                            controladorFactura.create(numero, Calendar.getInstance().getTime(), subtotal, 0, id);
                            controladorDetalleFactura.anadirLista(datosProductos);
                        } else {
                            System.out.println("Cliente no existe");
                        }
                        break;
                    case 3:
                        salir = true;
                    default:
                        System.out.println("Solo numeros entre el 1 y 2");
                }
            } catch (InputMismatchException e) {
                System.out.println("Debes insertar un numero.");
                input.next();
            }
        }
    }
}
```

```
public void buscar() {
    System.out.println("Ingrese el numero de Factura a buscar");
    int codigo = input.nextInt();
    if (controladorFactura.buscar(codigo) != null) {
        System.out.println(controladorDatosProducto.buscar(codigo).toString());
    } else {
        System.out.println("la factura no existe");
    }
}

public void eliminar() {
    System.out.println("Ingrese el numero de factura a eliminar:");
    int codigo = input.nextInt();
    if (controladorFactura.buscar(codigo) != null) {
        controladorFactura.eliminar(codigo);
        System.out.println("Factura eliminado");
    } else {
        System.out.println("factura no existente");
    }
}


public void listarFacturas() {
    controladorFactura.findAll().stream().forEach(object -> System.out.println(object));
    System.out.println(controladorDetalleFactura.MostrarLista());
}
```

En esa clase además de importar la clase Scanner también importamos todos los controladores y creamos un objeto de ellos.

En el método crear, agregamos un mini menú, donde primero nos pide ingresar los datos de la factura y de allí preguntamos si deseamos agregar productos ala factura con la llave única que es el Id del producto.

Para facturar y agregar el detalle de la factura en el mismo menú nos pide que ingresemos el Id de la persona o cliente que hace uso de la aplicación y que compro los productos para así agregarle al encabezado de la factura.

De allí todos los métodos buscar y eliminar a si mismo pide que ingresemos los datos por teclado y lo modifica o lo elimina.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

6.Explicacion de la clase Principal del paquete test:

```
DatosCliente datosCliente = new DatosCliente();
DatosProducto datosProducto = new DatosProducto();
DetalleFactura detalleFactura = new DetalleFactura();
Factura factura = new Factura();

ControladorDatosCliente controladorDatosCliente = new ControladorDatosCliente(datosCliente);
ControladorDatosProducto controladorDatosProducto = new ControladorDatosProducto(datosProducto);
ControladorFactura controladorFactura = new ControladorFactura(datosCliente, datosProducto, detalleFactura, factura, controladorDatosCliente);
ControladorDetalleFactura controladorDetalleFactura = new ControladorDetalleFactura(detalleFactura, factura);

VistaDatosCliente vistaDatosCliente = new VistaDatosCliente(controladorDatosCliente);
VistaDatosProducto vistaDatosProducto = new VistaDatosProducto(controladorDatosProducto);
VistaFactura vistaFactura = new VistaFactura(controladorFactura, controladorDatosProducto, controladorDetalleFactura, controladorDatosCliente);

Scanner input = new Scanner(System.in);
boolean salir = false;
boolean salir1 = false;
boolean salir2 = false;
boolean salir3 = false;
int opcion;
int opcion1;
int opcion2;
int opcion3;
```

En esta clase principal importamos y creamos un objeto de todas las vistas, controladores y modelos de todas las clases. Además unas opciones y un booleano para los menus.

```
while (!salir) {
    System.out.println("Bienvenido al sistema de David Dutan ");
    System.out.println(" 1.gestionar Cliente\n 2.gestionar Productos\n 3.crear Factura\n 4.salir");
    try {
        System.out.println("Elija una opcion");
        opcion = input.nextInt();

        switch (opcion) {
```

Creamos un pequeño menú donde podemos elegir entre 4 opciones para gestionar clientes, productos y facturar.

```
case 1:
    while (!salir1) {
        System.out.println("QUE DESEEA HACER");
        System.out.println("*****");
        System.out.println(
            " 1.Crear Cliente\n "
            + "2.Buscar Cliente \n "
            + "3.modificar Cliente\n "
            + "4.Eliminar Cliente\n "
            + "5.Listar clientes\n "
            + "6.salir\n ");
```

Si elegimos la opción 1 para gestionar clientes no aparece otro menú interno para crear, buscar, eliminar, modificar o imprimir todos los clientes.

```
try {
    System.out.println("Elija una opcion: ");
    opcion1 = input.nextInt();
    switch (opcion1) {
        case 1:
            vistaDatosCliente.crearCliente();
            System.out.println("Cliente creado");
            break;
        case 2:
            vistaDatosCliente.buscar();
            System.out.println("Cliente encontrado");
            break;
        case 3:
            vistaDatosCliente.modificar();
            System.out.println("Cliente modificado");
            break;
        case 4:
            vistaDatosCliente.eliminar();
            System.out.println("Cliente eliminado");
            break;
        case 5:
            vistaDatosCliente.ListarClientes();
            break;
        case 6:
            salir1 = true;
            break;
        default:
            System.out.println("Solo numeros entre el 1 y 6");
    }
} catch (InputMismatchException e) {
    System.out.println("Debes insertar un numero.");
    input.nextInt();
}
```

Y de acuerdo con cada opción ingresada por el usuario. Llamamos ala vista para que lo cree, modifique, elimine, busque o imprima.

```
case 2:
    while (!salir2) {
        System.out.println("QUE DESEEA HACER");
        System.out.println("*****");
        System.out.println(
            " 1.Registrar Producto\n "
            + "2.buscar Producto\n "
            + "3.modificar Producto\n "
            + "4.eliminar producto\n "
            + "5.Listar productos\n "
            + "6.salir\n "
        );
        try {
            System.out.println("Elija una opcion");
            opcion1 = input.nextInt();
            switch (opcion1) {
```

A si mismo del muenu principal para gestionar los datos del producto nos ingresa a otro menú interno para realizar lo que el usuario quiera hacer.


```
switch (opcion1) {
    case 1:
        vistaDatosProducto.crear();
        System.out.println("Producto creado");
        break;
    case 2:
        vistaDatosProducto.buscar();
        System.out.println("Producto encontrado");
        break;
    case 3:
        vistaDatosProducto.modificar();
        System.out.println("Producto modificado");
        break;
    case 4:
        vistaDatosProducto.eliminar();
        System.out.println("Producto eliminado");
        break;
    case 5:
        vistaDatosProducto.listarProductos();
        break;
    case 6:
        salir2 = true;
        break;
    default:
        System.out.println("Solo numeros entre el 1 y 6");
}
catch (InputMismatchException e) {
    System.out.println("Debes insertar un numero.");
    input.nextInt();
}
```

Tenemos cada opción para el crear, buscar, modificar, eliminar o imprimir. Llamamos ala vista de cada clase para lo realiza cada opción.

```
case 3:
    while (!salir3) {
        System.out.println("QUE DESEEA HACER");
        System.out.println("*****");
        System.out.println(
            " 1.Registrar Factura\n "
            + "2.buscar Factura \n "
            + "3.eliminar Factura\n "
            + "4.Listar facturas\n "
            + "5.salir\n " );
        try {
            System.out.println("Elija una opcion");
            opcion1 = input.nextInt();
            switch (opcion1) {
```

Para gestionar la factura de igual manera nos ingresa a un submenú para crear, buscar y eliminar.

```
case 1:
    vistaFactura.crear();
    break;

case 2:
    vistaFactura.buscar();
    System.out.println("Factura encontrada");
    break;

case 3:
    vistaFactura.eliminar();
    System.out.println("Factura Eliminada ");
    break;

case 4:
    vistaFactura.listarFacturas();
    break;

case 5:
    salir3 = true;
    break;

default:
    System.out.println("Solo numeros entre el 1 y 5");
```


Aquí cada opción para que las vistas de cada clase realice lo que el usuario desee hacer.

```
        }
    } catch (InputMismatchException e) {
        System.out.println("Debes insertar un numero.");
        input.nextInt();
    }
}
break;
case 4:
    salir = true;
    break;
default:
    System.out.println("Solo numeros entre el 1 y 5");
}
} catch (InputMismatchException e) {
    System.out.println("Debes insertar un numero.");
    input.nextInt();
}
}
}
```

El caso salir que sería la opción 4.

En todo el menú y los sub menús trabajamos con excepciones si por algún caso ingresamos letras y no numero o una opción fuera del número de opciones creadas.

7.Demostracion del programa por consola:

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

Bienvenido al sistema de David Dutan
1.gestionar Cliente
2.gestionar Productos
3.crear Factura
4.salir
Elija una opcion

```

Menu principal.

DatosCliente

```

Elija una opcion
1
QUE DESEEA HACER
*****
1.Crear Cliente
2.Buscar Cliente
3.modificar Cliente
4.Eliminar Cliente
5.Listar clientes
6.salir

Elija una opcion:

```

El submenú para gestionar clientes.

```

Elija una opcion:
1
Ingrese los datos del cliente:
elCodigo del cliente
01
Ingrese el nombre del Cliente:
david
Ingrese la cedula del cliente:
0106113632
Ingrese la fialidad de pago del Cliente:
buena
-----Usuario Creado-----
Cliente creado

```


Creamos el cliente

```

Elija una opcion:
2
Ingrese la ID del cliente a buscar
01
DatosCliente(id=1, nombre=david, cedula=0106113632, fialidadPago=buena)
Cliente encontrado
QUE DESEEA HACER

```

Mandamos a buscar el cliente

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

Elija una opcion:
3
Ingrese el codigo del cliente a Modificar
01
----Ingrese los nuevo datos del cliente----
Ingrese el nuevo nombre
jose
Ingrese la nueva cedula
0106113636
Ingrese fialidad de pago
mala
Cliente modificado

```

Modificamos el cliente, y posterior mandamos a que imprima a ver si modifiko o no.

```

Elija una opcion:
5
DatosCliente{id=1, nombre=jose, cedula=0106113636, fialidadPago=mala}
DatosCliente{id=2, nombre=pedro, cedula=02020202020, fialidadPago=exelente}
QUE DESEEA HACER
*****

```

Aquí también creamos otro cliente pero vemos que se modifiko el cliente con los datos nuevos ingresados

```

Elija una opcion:
4
Ingrese el codigo del cliente a eliminar:
01
Producto eliminado con exito
Cliente eliminado

```

Mandamos a eliminar un cliente y luego imprimimos a ver si se eliminó.

```

Elija una opcion:
5
DatosCliente{id=2, nombre=pedro, cedula=02020202020, fialidadPago=exelente}
QUE DESEEA HACER

```


Datos Producto:

```

Elija una opcion
2
QUE DESEEA HACER
*****
1.Registrar Producto
2.buscar Producto
3.modificar Producto
4.eliminar producto
5.Listar productos
6.salir

Elija una opcion
.

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

Elija una opcion
1
Ingrese los datos del producto :
Ingrese el codigo del producto:
03
Ingrese la descripcion del producto:
papas
Ingrese el precio unitario del producto:
1
Ingrese el stock del producto:
100
ingrese el iva del producto:
9
Producto Creado

```

Creamos el producto

```

Elija una opcion
2
Ingrese el codigo del producto a buscar
03
DatosProducto{codigo=3, descripcion=papas, precioUnitario=1.0, stock=100, iva=9.0}
Producto encontrado

```

Mandamos a buscar al producto creado

```

Elija una opcion
3
Ingrese el Codigo de producto a modificar:
03
-----Ingrese los nuevos datos del producto-----
Ingrese la nueva descripcion del producto
papasRuffles
Ingrese el nuevo precio
2
Ingrese el nuevo stock
500
Ingrese el nuevo iva del producto
12
Producto modificado

```

Modificamos al mismo producto


```
DatosProducto{codigo=3, descripcion=papasRuffles, precioUnitario=2.0, stock=500, iva=12.0}
```

Nos muestra ya con los valores cambiados

```

DatosProducto{codigo=3, descripcion=papasRuffles, precioUnitario=2.0, stock=500, iva=12.0}
DatosProducto{codigo=4, descripcion=helado, precioUnitario=55.0, stock=1000, iva=3.0}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

Elija una opcion
4
Ingrese el codigo del producto a eliminar:
03
Producto eliminado

```

```

Elija una opcion
5
DatosProducto{codigo=4, descripcion=helado, precioUnitario=55.0, stock=1000, iva=3.0}
QUE DESEEA HACER

```

Imprimimos los productos y nos muestra que no existe el producto eliminado anteriormente

Factura

```

Elija una opcion
3
QUE DESEEA HACER
*****
1.Registrar Factura
2.buscar Factura
3.eliminar Factura
4.Listar facturas
5.salir

```


El sub menu para gestionar la factura.

```

Elija una opcion
1
Ingrese los Datos de la Factura:
Ingrese numero de Factura:
55
Fecha (dd/mm/yyyy):
08/06/2021
1.Agregar producto
2.Facturar
3.Salir
Elija una opcion
1
ingrese el codigo del producto para agregar a la factura
2
Producto agregado ala factura
1.Agregar producto
2.Facturar
3.Salir
Elija una opcion
2
Dijite el numero del Cliente
01
1.Agregar producto
2.Facturar
3.Salir
Elija una opcion
3

```

Creamos la factura con los datos que ingresa el usuario y ademas preguntamos si desea agregar productos, posterios a ello preguntamos si desea facturar y mandamos el emcabazado de la bactura que serin los datos del cliente todo ello solo por el Id de cada cliente o producto.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Ingrese el numero de Factura a buscar

55

```
DatosProducto{codigo=3, descripcion=papas, precioUnitario=10.0, stock=1000, iva=9.0}
```

return resultado;

Impresión de la factura con los datos y buscamos.

```
1
Factura{numero=55, fecha=Tue Jun 08 16:08:43 EDT 2021, total=10.0, cantidad=0}
[2]
```

Mandamos a eliminar.

Elija una opcion

3

Ingrese el numero de factura a eliminar:

55

Factura eliminado

CONCLUSIONES:

Realizar una practica buena implementando las buenas nomas de la programación, utilizar repositorios y softwares para la creación del UML

Nombre de estudiante: DUTAN SANCHEZ DAVID GUSTAVO