

Universidad Mariano Gálvez

Sede Boca del Monte

Facultad de Ingeniería en Sistemas

Sección: A

Catedrático: Ingeniero Luis Alvarado

Tarea:

Segundo Proyecto

Nombre: Walter Yair Ramos Carreto

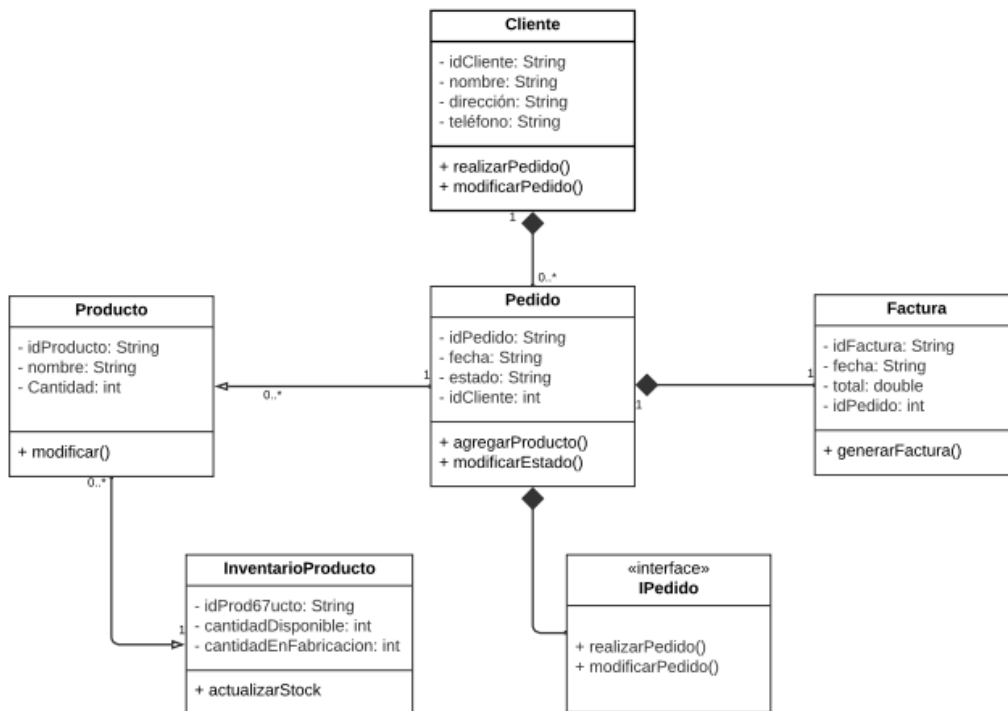
Carnet: 7690-23-17204

Guatemala, 01 de octubre de 2024

Introducción: el siguiente proyecto se realizo con El Patrón de Diseño de Única Responsabilidad, al igual se utilizaron archivos .json, los cuales nos servirán para almacenar la información y el control de productos por estados.

Nuestro UML

PROYECTO WALTER RAMOS



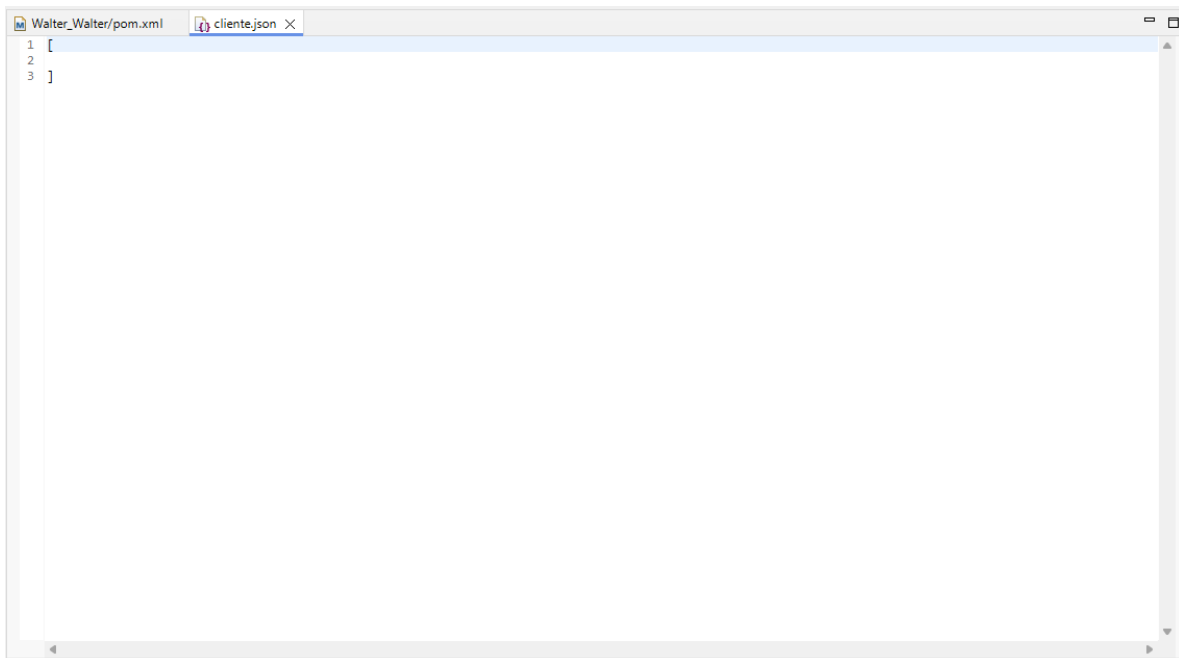
A continuación,

nuestro `pom.xml`

En el cual se utilizó Maven con Java 17

```
Walter_Walter/pom.xml X
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>org.example</groupId>
8   <artifactId>Walter</artifactId>
9   <version>1.0-SNAPSHOT</version>
10  <dependencies>
11    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
12    <dependency>
13      <groupId>com.fasterxml.jackson.core</groupId>
14      <artifactId>jackson-databind</artifactId>
15      <version>2.17.2</version>
16    </dependency>
17  </dependencies>
18  <properties>
19    <maven.compiler.source>17</maven.compiler.source>
20    <maven.compiler.target>17</maven.compiler.target>
21    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22  </properties>
23
24 </project>
```

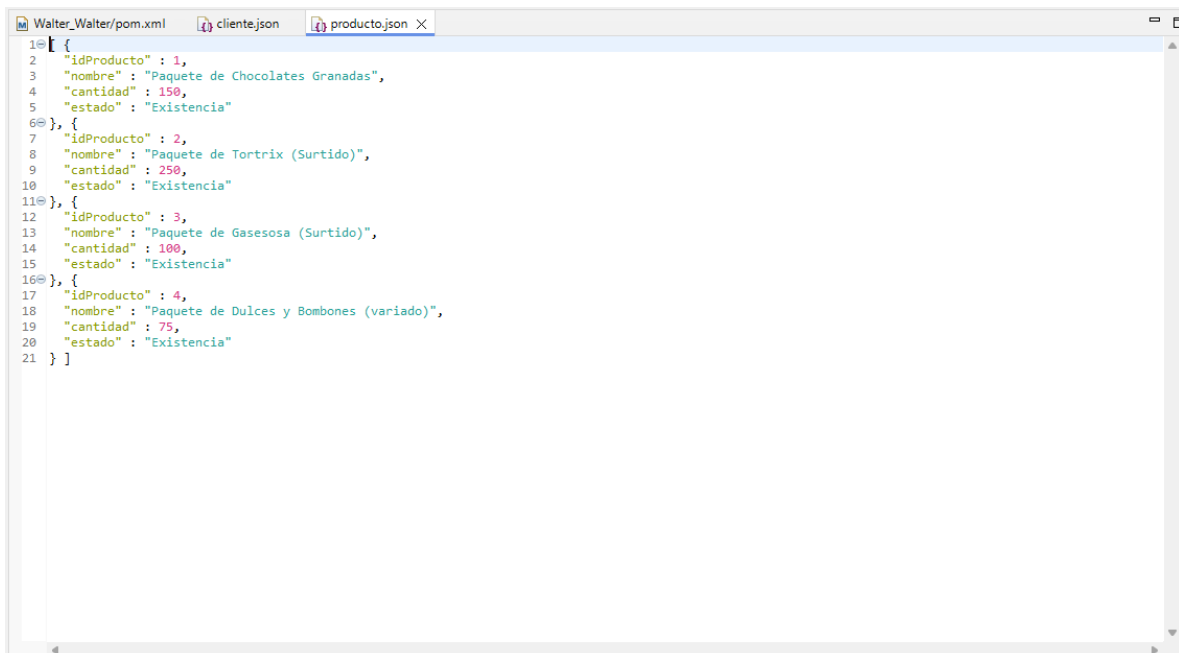
En nuestros Archivos **.json** los cuales usaremos dos con nombre **cliente.json** y **producto.json**



```
1 [
2
3 ]
```

Este archivo se encuentra limpio, ya que no se agregado nada.

En nuestro **producto.json** se agregaron productos a nuestro stock



```
1 [ {
2   "idProducto" : 1,
3   "nombre" : "Paquete de Chocolates Granadas",
4   "cantidad" : 150,
5   "estado" : "Existencia"
6 }, {
7   "idProducto" : 2,
8   "nombre" : "Paquete de Tortrix (Surtido)",
9   "cantidad" : 250,
10  "estado" : "Existencia"
11 }, {
12  "idProducto" : 3,
13  "nombre" : "Paquete de Gasesosa (Surtido)",
14  "cantidad" : 100,
15  "estado" : "Existencia"
16 }, {
17  "idProducto" : 4,
18  "nombre" : "Paquete de Dulces y Bombones (variado)",
19  "cantidad" : 75,
20  "estado" : "Existencia"
21 } ]
```

Por consiguiente, tenemos nuestro main.java, en el cual nos servirá para agregar clientes y verificar los productos de nuestro stock, para ser llamados a través de otras clases.

```
Walter_Walter/pom.xml cliente.json producto.json Main.java X
1 package org.example;
2
3 import java.io.IOException;
4
5
6 public class Main {
7     public static void main(String[] args) throws Exception {
8         RepositorioProducto.cargarProductos();
9         RepositorioCliente.cargarClientes();
10        Scanner scanner = new Scanner(System.in);
11        Integer opcion = 0;
12        do {
13            System.out.println("1. Realizar una compra");
14            System.out.println("2. Verificar Stock de Productos");
15            System.out.println("3. Ver Clientes y Stock de Compras");
16            System.out.println();
17            System.out.println("- Presione 99 para salir: ");
18
19
20            opcion = scanner.nextInt();
21            switch (opcion) {
22                case 1:
23                    realizarCompra(scanner);
24                    break;
25                case 2:
26                    RepositorioProducto.getProductos().forEach(producto -> {
27                        System.out.println("Id: " + producto.getIdProducto() + " Nombre: " + producto.getNombre() + " Estado: " + producto.getEstado());
28                    });
29                    System.out.println();
30                    break;
31                case 3:
32                    RepositorioCliente.getClientes().forEach(cliente -> {
33                        System.out.println("Cliente: " + cliente.getNombre());
34                        System.out.println("Nit: " + cliente.getNit());
35                        System.out.println("Direccion: " + cliente.getDireccion());
36                        System.out.println("Telefono: " + cliente.getTelefono());
37                        System.out.println("Pedidos: ");
38                        cliente.getPedidos().forEach(pedido -> {
39                            System.out.println("    Id Pedido: " + pedido.getIdPedido());
40                            System.out.println("    Estado: " + pedido.getEstado());
41                            System.out.println("    Fecha de la venta: " + pedido.getFecha());
42                            List<Producto> productos = pedido.getProductos();
43                            if (productos != null) {
44                                for (Producto producto : productos) {
45                                    System.out.println("        Producto: " + producto.getNombre());
46                                    System.out.println("        Cantidad: " + producto.getCantidad());
47                                    System.out.println("        Id del producto: " + producto.getIdProducto());
48                                }
49                            }
50                            System.out.println();
51                        });
52                    });
53                    break;
54            }
55        } while (opcion != 99);
56        scanner.close();
57    }
58
59
60 private static void realizarCompra(Scanner scanner) throws IOException {
61     Cliente cliente = new Cliente();
62     System.out.println("Ingrese el NIT del consumidor: ");
63     cliente.setNit(scanner.next());
64     System.out.println("Ingrese el Nombre del consumidor: ");
65     cliente.setNombre(scanner.next());
66     System.out.println("Ingrese el Telefono del consumidor: ");
67     cliente.setTelefono(scanner.next());
68     System.out.println("Ingrese la Direccion del consumidor: ");
69     cliente.setDireccion(scanner.next());
70
71     Pedido pedido = new Pedido();
72     pedido.setIdPedido((int) (Math.random() * 100));
73 }
```

```
Walter_Walter/pom.xml cliente.json producto.json Main.java X
35 System.out.println("Direccion: " + cliente.getDireccion());
36 System.out.println("Telefono: " + cliente.getTelefono());
37 System.out.println("Pedidos: ");
38 cliente.getPedidos().forEach(pedido -> {
39     System.out.println("    Id Pedido: " + pedido.getIdPedido());
40     System.out.println("    Estado: " + pedido.getEstado());
41     System.out.println("    Fecha de la venta: " + pedido.getFecha());
42     List<Producto> productos = pedido.getProductos();
43     if (productos != null) {
44         for (Producto producto : productos) {
45             System.out.println("        Producto: " + producto.getNombre());
46             System.out.println("        Cantidad: " + producto.getCantidad());
47             System.out.println("        Id del producto: " + producto.getIdProducto());
48         }
49     }
50     System.out.println();
51 });
52 });
53 break;
54 }
55 }
56 } while (opcion != 99);
57 scanner.close();
58 }
59
60 private static void realizarCompra(Scanner scanner) throws IOException {
61     Cliente cliente = new Cliente();
62     System.out.println("Ingrese el NIT del consumidor: ");
63     cliente.setNit(scanner.next());
64     System.out.println("Ingrese el Nombre del consumidor: ");
65     cliente.setNombre(scanner.next());
66     System.out.println("Ingrese el Telefono del consumidor: ");
67     cliente.setTelefono(scanner.next());
68     System.out.println("Ingrese la Direccion del consumidor: ");
69     cliente.setDireccion(scanner.next());
70
71     Pedido pedido = new Pedido();
72     pedido.setIdPedido((int) (Math.random() * 100));
73 }
```

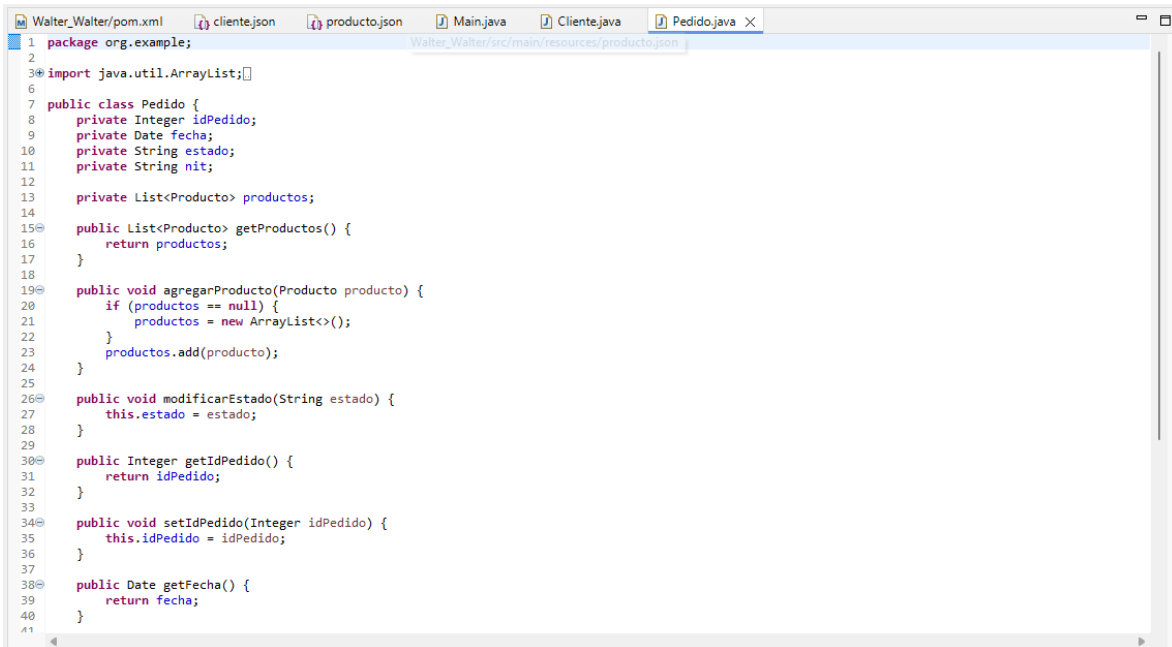
```
Walter_Walter/pom.xml cliente.json producto.json Main.java X
68 System.out.println("Ingrese la direccion del consumidor: ");
69 cliente.setDireccion(scanner.next());
70
71
72 Pedido pedido = new Pedido();
73 pedido.setIdPedido((int) (Math.random() * 100));
74 pedido.setFecha(new Date());
75 pedido.setNit(cliente.getNit());
76 pedido.modificarEstado("Despachado");
77
78 while (true) {
79     RepositorioProducto.getProductos().forEach(producto -> {
80         System.out.println("Id: " + producto.getIdProducto() + " Nombre: " + producto.getNombre() + " Estado: " + producto.getEstado());
81     });
82     scanner.nextLine();
83
84     System.out.println("Ingrese el Numero para el PRODUCTO que necesita: ");
85     Integer idProducto = scanner.nextInt();
86     System.out.println("Ingrese la Cantidad que desea comprar: ");
87     Integer cantidad = scanner.nextInt();
88
89     Producto producto = RepositorioProducto.despacharProducto(idProducto, cantidad);
90     pedido.agregarProducto(producto);
91
92     System.out.println("Necesita agregar mas productos (si/no: ");
93     String respuesta = scanner.next();
94     if ("no".equalsIgnoreCase(respuesta)) {
95         break;
96     }
97 }
98 cliente.setPedidos(List.of(pedido));
99 cliente.realizarPedido();
100 RepositorioCliente.agregarCliente(cliente);
101 RepositorioCliente.guardarClientes();
102 RepositorioProducto.guardarProductos();
103 }
104
105
106 }
```

Tenemos nuestra clase **Cliente.java** en donde nos pedirá los datos del cliente, para que se agreguen en nuestro **Cliente.json** a continuación el código:

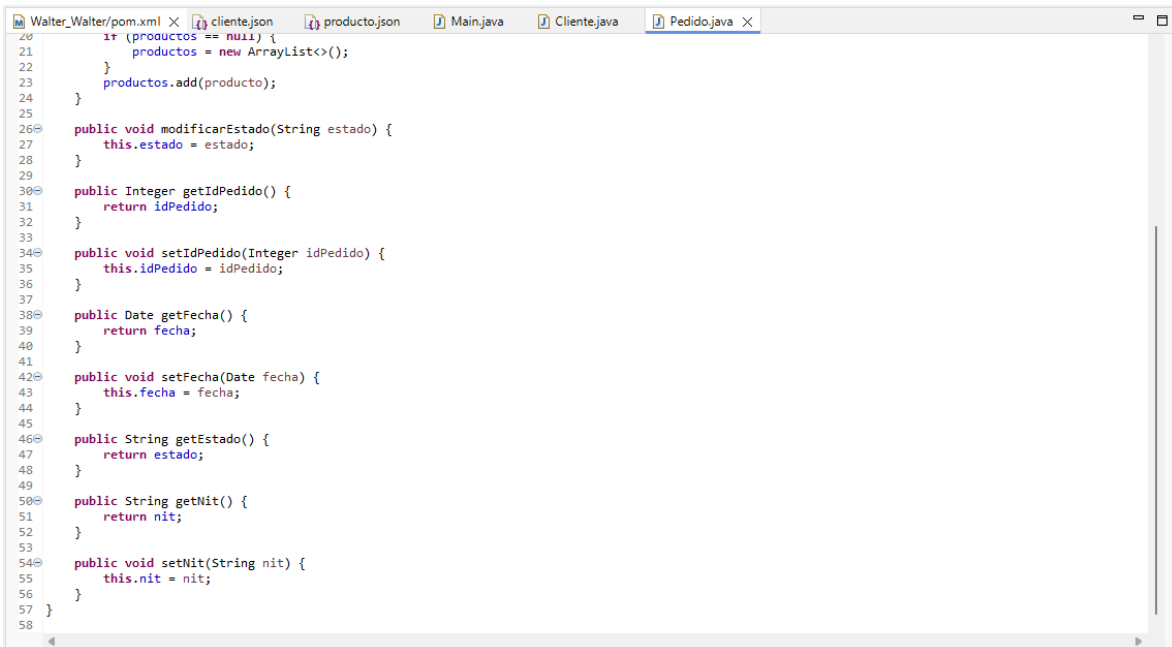
```
Walter_Walter/pom.xml x cliente.json producto.json Main.java Cliente.java x
1 package org.example;
2
3 import java.util.List;
4
5 public class Cliente {
6     private String nit;
7     private String nombre;
8     private String direccion;
9     private String telefono;
10
11     private List<Pedido> pedidos;
12
13
14     public void realizarPedido() {
15         System.out.println("Realizando pedido");
16         System.out.println("/*****");
17         System.out.println("Cliente: " + nombre);
18         System.out.println("Nit: " + nit);
19         System.out.println("Telefono: " + telefono);
20         System.out.println("Direccion: " + direccion);
21
22
23         for (Pedido pedido : pedidos) {
24             System.out.println("-----");
25             System.out.println("Id Pedido: " + pedido.getIdPedido());
26             System.out.println("Estado: " + pedido.getEstado());
27             System.out.println("Fecha de la venta: " + pedido.getFecha());
28             List<Producto> productos = pedido.getProductos();
29             if (productos != null) {
30                 for (Producto producto : productos) {
31                     System.out.println("Producto: " + producto.getNombre());
32                     System.out.println("Cantidad: " + producto.getCantidad());
33                     System.out.println("Id del producto: " + producto.getIdProducto());
34
35                     System.out.println();
36                 }
37             }
38             System.out.println();
39         }
40     }
}
```

```
Walter_Walter/pom.xml x cliente.json producto.json Main.java Cliente.java x
41
42     public void setPedidos(List<Pedido> pedidos) {
43         this.pedidos = pedidos;
44     }
45
46     public String getNit() {
47         return nit;
48     }
49
50     public void setNit(String nit) {
51         this.nit = nit;
52     }
53
54     public String getNombre() {
55         return nombre;
56     }
57
58     public void setNombre(String nombre) {
59         this.nombre = nombre;
60     }
61
62     public String getTelefono() {
63         return telefono;
64     }
65
66     public void setTelefono(String telefono) {
67         this.telefono = telefono;
68     }
69
70     public String getDireccion() {
71         return direccion;
72     }
73
74     public void setDireccion(String direccion) {
75         this.direccion = direccion;
76     }
77
78 }
79
80
81
82
83
84
85
```

En nuestra clase **Pedido.java** tendremos los estados de nuestro producto en Stock que estarán en nuestro **producto.json**, allí se observara si habrá en existencia o ya estará agotado los productos.

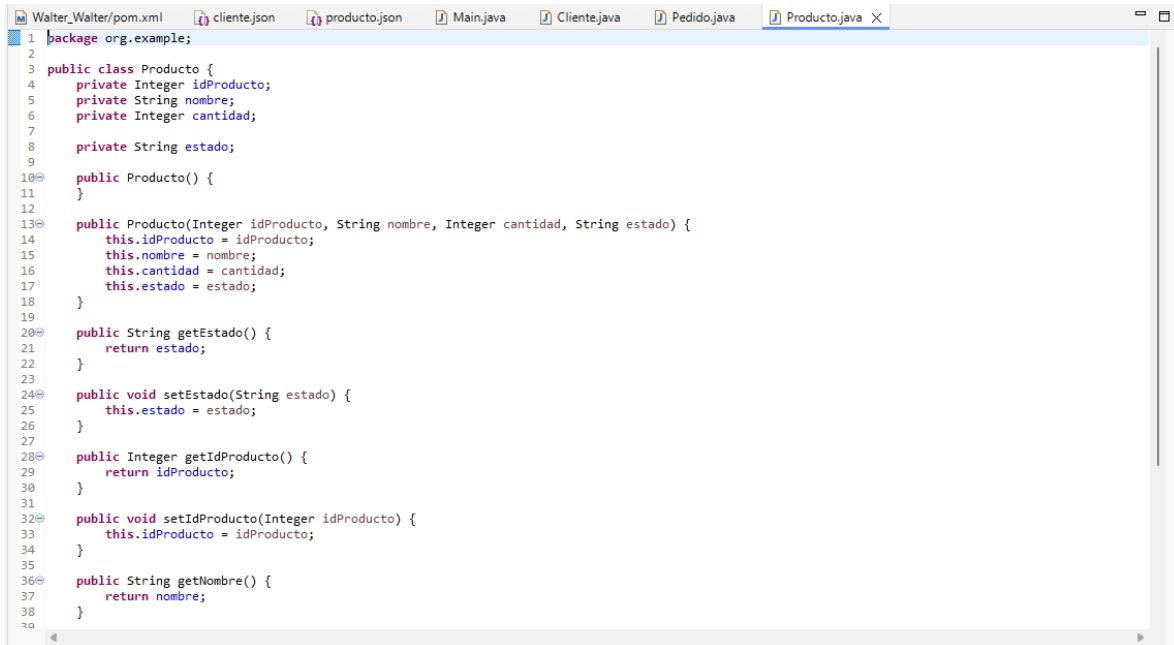


```
1 package org.example;
2
3 import java.util.ArrayList;
4
5
6
7 public class Pedido {
8     private Integer idPedido;
9     private Date fecha;
10    private String estado;
11    private String nit;
12
13    private List<Producto> productos;
14
15    public List<Producto> getProductos() {
16        return productos;
17    }
18
19    public void agregarProducto(Producto producto) {
20        if (productos == null) {
21            productos = new ArrayList<>();
22        }
23        productos.add(producto);
24    }
25
26    public void modificarEstado(String estado) {
27        this.estado = estado;
28    }
29
30    public Integer getIdPedido() {
31        return idPedido;
32    }
33
34    public void setIdPedido(Integer idPedido) {
35        this.idPedido = idPedido;
36    }
37
38    public Date getFecha() {
39        return fecha;
40    }
41}
```

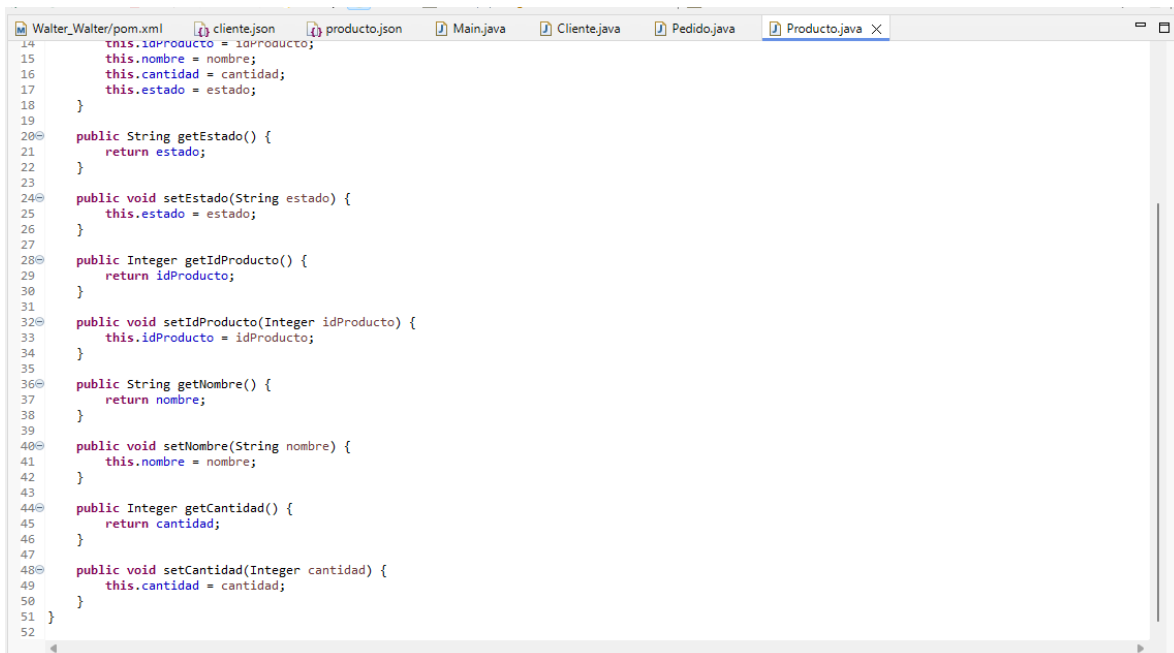


```
42    public void setFecha(Date fecha) {
43        this.fecha = fecha;
44    }
45
46    public String getEstado() {
47        return estado;
48    }
49
50    public String getNit() {
51        return nit;
52    }
53
54    public void setNit(String nit) {
55        this.nit = nit;
56    }
57 }
58
```


En nuestra clase **Producto.java** vamos a observar la cantidad de producto que tendremos en nuestro Stock que será **producto.json**, se encargara de restar lo solicitado por el cliente.



```
1 package org.example;
2
3 public class Producto {
4     private Integer idProducto;
5     private String nombre;
6     private Integer cantidad;
7
8     private String estado;
9
10    public Producto() {
11    }
12
13    public Producto(Integer idProducto, String nombre, Integer cantidad, String estado) {
14        this.idProducto = idProducto;
15        this.nombre = nombre;
16        this.cantidad = cantidad;
17        this.estado = estado;
18    }
19
20    public String getEstado() {
21        return estado;
22    }
23
24    public void setEstado(String estado) {
25        this.estado = estado;
26    }
27
28    public Integer getIdProducto() {
29        return idProducto;
30    }
31
32    public void setIdProducto(Integer idProducto) {
33        this.idProducto = idProducto;
34    }
35
36    public String getNombre() {
37        return nombre;
38    }
39
40 }
```



```
41    this.idProducto = idProducto;
42    this.nombre = nombre;
43    this.cantidad = cantidad;
44    this.estado = estado;
45 }
46
47 public String getEstado() {
48     return estado;
49 }
50
51 public void setEstado(String estado) {
52     this.estado = estado;
53 }
54
55 public Integer getIdProducto() {
56     return idProducto;
57 }
58
59 public void setIdProducto(Integer idProducto) {
60     this.idProducto = idProducto;
61 }
62
63 public String getNombre() {
64     return nombre;
65 }
66
67 public void setNombre(String nombre) {
68     this.nombre = nombre;
69 }
70
71 public Integer getCantidad() {
72     return cantidad;
73 }
74
75 public void setCantidad(Integer cantidad) {
76     this.cantidad = cantidad;
77 }
78
79 }
```

En nuestra clase **RepositorioCliente.java** vamos a observar si nuestro cliente ya está registrado y si ya esta en nuestro sistema, al igual nos ayudara a mostrar a nuestros clientes que ya están en nuestro sistema.

```
1 package org.example;
2
3 import com.fasterxml.jackson.databind.ObjectMapper;
4
12 public class RepositorioCliente {
13     private static List<Cliente> clientes = new ArrayList<>();
14     private static Map<String, String> map = new HashMap<>();
15
16     public static void cargarClientes() throws IOException {
17         URL url = new URL("file:src/main/resources/cliente.json");
18
19         ObjectMapper mapper = new ObjectMapper();
20         Cliente[] clientesTmp = mapper.readValue(new File(url.getPath()), Cliente[].class);
21
22         for (Cliente cliente : clientesTmp) {
23             if (map.get(cliente.getNit()) != null) {
24                 System.out.println("Cliente con ID duplicado nit: " + cliente.getNit() + " nombre: " + cliente.getNombre() + " se procede a ignorarlo.");
25                 continue;
26             }
27             map.put(cliente.getNit(), cliente.getNombre());
28             clientes.add(cliente);
29         }
30
31         System.out.println("clientes cargados: " + clientes.size());
32     }
33
34     public static void guardarClientes() throws IOException {
35         System.out.println("Actualizando clientes...");
36         URL url = new URL("file:src/main/resources/cliente.json");
37
38         ObjectMapper mapper = new ObjectMapper();
39         mapper.writerWithDefaultPrettyPrinter().writeValue(new File(url.getPath()), clientes);
40         System.out.println("clientes guardados: " + clientes.size());
41     }
42
43     public static void agregarCliente(Cliente cliente) {
44         if (map.get(cliente.getNit()) != null) {
45             clientes.forEach(c -> {
46                 if (c.getNit().equals(cliente.getNit())) {
47                     c.getPedidos().add(cliente.getPedidos());
48                 }
49             });
50         }
51         return;
52     }
53     System.out.println("Agregando cliente id: " + cliente.getNit() + " nombre: " + cliente.getNombre());
54     clientes.add(cliente);
55 }
56
57 public static List<Cliente> getClientes() {
58     return clientes;
59 }
60 }
61
```

```
23     for (Cliente cliente : clientesTmp) {
24         if (map.get(cliente.getNit()) != null) {
25             System.out.println("Cliente con ID duplicado nit: " + cliente.getNit() + " nombre: " + cliente.getNombre() + " se procede a ignorarlo.");
26             continue;
27         }
28         map.put(cliente.getNit(), cliente.getNombre());
29         clientes.add(cliente);
30     }
31
32     System.out.println("clientes cargados: " + clientes.size());
33 }
34
35 public static void guardarClientes() throws IOException {
36     System.out.println("Actualizando clientes...");
37     URL url = new URL("file:src/main/resources/cliente.json");
38
39     ObjectMapper mapper = new ObjectMapper();
40     mapper.writerWithDefaultPrettyPrinter().writeValue(new File(url.getPath()), clientes);
41     System.out.println("clientes guardados: " + clientes.size());
42 }
43
44 public static void agregarCliente(Cliente cliente) {
45     if (map.get(cliente.getNit()) != null) {
46         clientes.forEach(c -> {
47             if (c.getNit().equals(cliente.getNit())) {
48                 cliente.getPedidos().forEach(pedido -> c.getPedidos().add(pedido));
49             }
50         });
51         return;
52     }
53     System.out.println("Agregando cliente id: " + cliente.getNit() + " nombre: " + cliente.getNombre());
54     clientes.add(cliente);
55 }
56
57 public static List<Cliente> getClientes() {
58     return clientes;
59 }
60 }
61
```

En nuestro **RepositorioProducto.java** verificaremos que nuestro producto no este duplicado al igual que no haya más productos con ese nombre o con esta existencia en nuestro Stock.

```
1 package org.example;
2
3 import com.fasterxml.jackson.databind.ObjectMapper;
4
5
6
7
8
9
10 public class RepositorioProducto {
11     private static List<Producto> productos = new ArrayList<>();
12     private static Map<Integer, Integer> map = new HashMap<>();
13
14     public static void cargarProductos() throws IOException {
15         System.out.println("Cargando productos...");
16         URL url = new URL("file:src/main/resources/producto.json");
17
18         ObjectMapper mapper = new ObjectMapper();
19         Producto[] productosTemp = mapper.readValue(new File(url.getPath()), Producto[].class);
20
21         Integer index = 0;
22         for (Producto producto : productosTemp) {
23             if (map.get(producto.getIdProducto()) != null) {
24                 System.out.println("Producto con ID duplicado id: " + producto.getIdProducto() + " nombre: " + producto.getNombre() + " se procede a ignorar");
25                 continue;
26             }
27             map.put(producto.getIdProducto(), index++);
28             productos.add(producto);
29         }
30
31         System.out.println("Productos cargados: " + productos.size() + " \n");
32     }
33
34     public static void guardarProductos() throws IOException {
35         System.out.println("Guardando productos...");
36         URL url = new URL("file:src/main/resources/producto.json");
37
38         ObjectMapper mapper = new ObjectMapper();
39         mapper.writerWithDefaultPrettyPrinter().writeValue(new File(url.getPath()), productos);
40         System.out.println("Productos guardados: " + productos.size());
41     }
42
43     public static void agregarProducto(Producto producto) {
44         System.out.println("Agregando producto id: " + producto.getIdProducto() + " nombre: " + producto.getNombre());
45     }
46 }
```

```
41
42 }
43
44 public static void agregarProducto(Producto producto) {
45     System.out.println("Agregando producto id: " + producto.getIdProducto() + " nombre: " + producto.getNombre());
46     if (map.get(producto.getIdProducto()) != null) {
47         throw new IllegalArgumentException("Producto con ID duplicado id: " + producto.getIdProducto() + " nombre: " + producto.getNombre());
48     }
49     productos.add(producto);
50 }
51
52 public static Producto despacharProducto(int idProducto, int cantidadRequerida) {
53     Producto producto = getProducto(idProducto);
54     Integer cantidad = producto.getCantidad();
55
56     if (cantidad < cantidadRequerida) {
57         throw new IllegalArgumentException("No hay suficiente cantidad de producto id: " + idProducto + " nombre: " + producto.getNombre());
58     }
59
60     producto.setCantidad(cantidad - cantidadRequerida);
61
62     if (producto.getCantidad() == 0) {
63         producto.setEstado("Agotado");
64     }
65     return new Producto(producto.getIdProducto(), producto.getNombre(), cantidadRequerida, producto.getEstado());
66 }
67
68 public static List<Producto> getProductos() {
69     return productos;
70 }
71
72 private static Producto getProducto(int idProducto) {
73     Integer productId = map.get(idProducto);
74     if (productId == null) {
75         throw new IllegalArgumentException("Producto no encontrado id: " + idProducto);
76     }
77     return productos.get(productId);
78 }
79 }
```