



UNIVERSIDAD MADERO PUEBLA
INCORPORADA A LA SEP

[PESCADERIA PSIP]

SANTIAGO SANTOS DEL VALLE



ISRAEL BONILLA CARRANZA

PABLO FARID MONTORO ARAUJO

PLATAFORMAS ABIERTAS I

OTOÑO 2024

Índice

1. Requerimientos	3
1.1. Objetivo general del proyecto	3
1.2. Datos maestros	3
1.3. Procesos y Datos Transaccionales	3
2. Análisis y Diseño	4
2.1. Diagrama de clases	4
3. Implementación	5
3.1. Código de clases en Java	5
3.2. Configuración inicial para conexión a Base de datos	16
3.3. Pantallas	16

1. Requerimientos

1.1. Objetivo general del proyecto

Desarrollar una aplicación de gestión para una pescadería, que permita administrar el inventario de pescados mediante una base de datos MySQL, facilitando la adición, actualización, eliminación y consulta de productos. La aplicación debe permitir a los usuarios generar tickets de compra con información detallada sobre los productos adquiridos, incluyendo la cantidad y el precio total, mejorando así la eficiencia en la operación y el control del negocio.

1.2. Datos maestros

Los datos maestros son información crítica y esencial que se utiliza como referencia en los procesos operativos de la pescadería.

- **Pescado**
 - o id_pescado, peso, stock, nombre, categoría (fresco, congelado)
- **Empleado**
 - o Id_empleado, nombre, apellido, fecha contratación, teléfono, rol (vendedor, cargador)
- **Cliente**
 - o Id_cliente, correo, contraseña, nombre, apellido, telefono, preferencia
- **Proveedor**
 - o Id_proveedor, nombre, telefono, direccion, correo
-

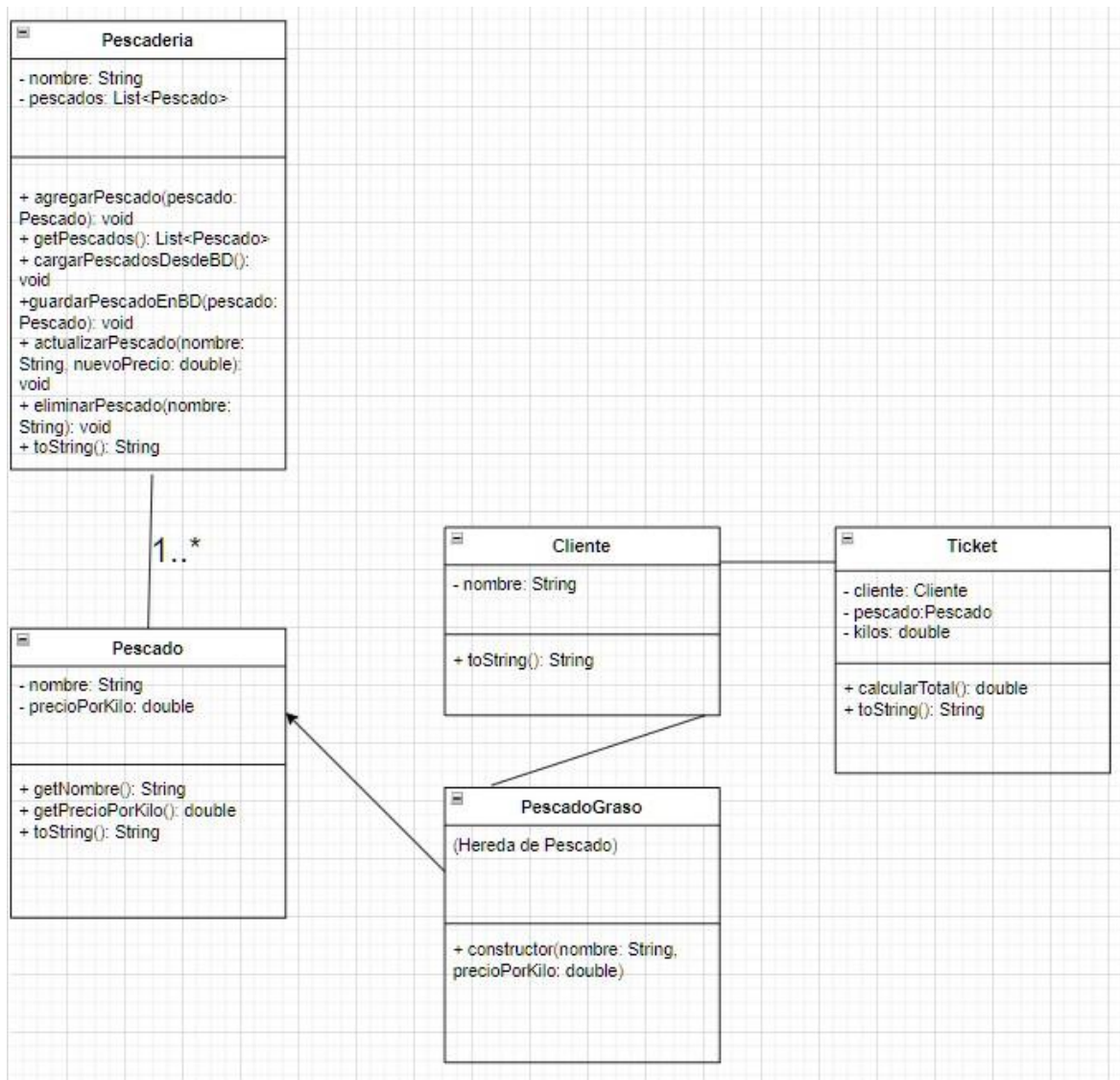
1.3. Procesos y Datos Transaccionales

- **Login y logout**
 - o Acceso y cierre de sesión de un empleado al sistema
- **CRUD de datos maestros**
 - o Venta, Ticket, Pescados Vendidos, Pescados Comprados,
 - o **Creación de venta**
 - **Encabezado de la venta**
 - Empleado
 - Fecha de venta
 - Miembro a quien se venden los pescados
 - **Detalles de la venta**
 - (Pescado, cantidad, fecha de compra)
 - o **Visualización de la venta**
 - Búsqueda de la venta por número de ticket

2. Análisis y Diseño

El **análisis y diseño** son etapas clave en el desarrollo de software, ya que nos permiten estructurar de manera lógica los elementos que componen el sistema. En este apartado, presentamos el **diagrama de clases** del sistema de gestión de pescadería, el cual refleja cómo las distintas entidades, como pescados, clientes y tickets, se relacionan entre sí.

2.1. Diagrama de clases



3. Implementación

3.1. Código de clases en Java

```
package com.mycompany.pescaderiapsip2;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * La clase PescaderiaPSIP2 es la aplicación principal de la pescadería,
 * que interactúa
 * con una base de datos MySQL para gestionar los pescados
 * disponibles, agregar nuevos pescados,
 * generar tickets de compra y mostrar la información al usuario.
 *
 * @author Farid, Santiago, Israel, Damian
 */
public class PescaderiaPSIP2 {

    public static void main(String[] args) {
        // Crear una instancia de la pescadería
        Pescaderia pescaderia = new Pescaderia("La PescaderíaPSIP");

        // Agregar pescados predeterminados a la base de datos
        pescaderia.agregarPescado(new Pescado("Salmon", 451.0));
        pescaderia.agregarPescado(new Pescado("Pulpo", 324.0));
    }
}
```

```

pescaderia.agregarPescado(new Pescado("Mojarra", 56.0));
pescaderia.agregarPescado(new PescadoGraso("Atún", 300.0));
pescaderia.agregarPescado(new PescadoGraso("Sardina", 185.0));
pescaderia.agregarPescado(new PescadoGraso("Caballa", 100.0));

// Mostrar los pescados disponibles en la pescadería
System.out.println(pescaderia);

// Crear un cliente y generar un ticket
Cliente cliente = new Cliente("Alejandro Barroeta");
Ticket ticket = new Ticket(cliente, pescaderia.getPescados().get(0),
7.2);
System.out.println(ticket);

// Permitir que el usuario ingrese un nuevo pescado si lo desea
Scanner scanner = new Scanner(System.in);
System.out.println("¿Quieres agregar un nuevo pescado? (si/no)");
String respuesta = scanner.nextLine();

if (respuesta.equalsIgnoreCase("si")) {
    // Solicitar el nombre y precio del pescado al usuario
    System.out.print("Introduce el nombre del pescado: ");
    String nombrePescado = scanner.nextLine();
    System.out.print("Introduce el precio por kilo: ");
    double precio = scanner.nextDouble();

    // Validación del precio (no puede ser menor o igual a cero)

```

```

        if (precio <= 0) {
            System.out.println("Error: El precio no puede ser menor o igual
a cero.");
        } else {
            // Crear el nuevo pescado y agregarlo a la pescadería
            Pescado nuevoPescado = new Pescado(nombrePescado,
precio);
            pescaderia.agregarPescado(nuevoPescado);
            System.out.println("Pescado agregado: " + nuevoPescado);
        }
    }

    // Mostrar los pescados después de agregar el nuevo pescado
    System.out.println("Pescados actualizados: " + pescaderia);
}

/**
 * Clase que representa una pescadería. Contiene una lista de pescados
y
 * permite agregar pescados, cargar desde la base de datos y guardar
cambios.
 */
class Pescaderia {
    private String nombre; // Nombre de la pescadería
    private List<Pescado> pescados; // Lista de pescados disponibles
    private static final String DB_URL =
"jdbc:mysql://localhost:3306/pescaderia";

```

```
private static final String DB_USER = "root"; // Cambiar según la
configuración de tu DB
```

```
private static final String DB_PASSWORD = ""; // Cambiar según la
configuración de tu DB
```

```
public Pescaderia(String nombre) {

    this.nombre = nombre;

    this.pescados = new ArrayList<>();

    cargarPescadosDesdeBD(); // Cargar los pescados desde la base
de datos al iniciar

}
```

```
/**
```

```
 * Agrega un nuevo pescado a la lista y lo guarda en la base de datos.
```

```
*/
```

```
public void agregarPescado(Pescado pescado) {

    pescados.add(pescado);

    guardarPescadoEnBD(pescado); // Guardar el pescado en la base
de datos

}
```

```
/**
```

```
 * Carga los pescados desde la base de datos MySQL.
```

```
*/
```

```
private void cargarPescadosDesdeBD() {

    try (Connection conn = DriverManager.getConnection(DB_URL,
DB_USER, DB_PASSWORD);

        Statement stmt = conn.createStatement();
```



```

        ResultSet rs = stmt.executeQuery("SELECT nombre,
precioPorKilo FROM pescados")) {

        while (rs.next()) {

            String nombre = rs.getString("nombre");

            double precioPorKilo = rs.getDouble("precioPorKilo");

            pescados.add(new Pescado(nombre, precioPorKilo)); //
Agregar cada pescado desde la base de datos

        }

    } catch (SQLException e) {

        e.printStackTrace(); // En caso de error en la conexión o
consulta

    }

}

/**
 * Guarda un pescado en la base de datos.
 */

private void guardarPescadoEnBD(Pescado pescado) {

    String query = "INSERT INTO pescados (nombre, precioPorKilo)
VALUES (?, ?)";

    try (Connection conn = DriverManager.getConnection(DB_URL,
DB_USER, DB_PASSWORD);

        PreparedStatement pstmt = conn.prepareStatement(query)) {

        pstmt.setString(1, pescado.getNombre());

        pstmt.setDouble(2, pescado.getPrecioPorKilo());

```

```

        pstmt.executeUpdate(); // Ejecutar la inserción en la base de
datos
    } catch (SQLException e) {
        e.printStackTrace(); // En caso de error en la conexión o
ejecución
    }
}

/**
 * Actualiza el precio de un pescado en la base de datos.
 */
public void actualizarPescado(String nombre, double nuevoPrecio) {
    if (nuevoPrecio <= 0) {
        System.out.println("Error: El precio no puede ser menor o igual a
cero.");
        return;
    }

    String query = "UPDATE pescados SET precioPorKilo = ? WHERE
nombre = ?";

    try (Connection conn = DriverManager.getConnection(DB_URL,
DB_USER, DB_PASSWORD);
        PreparedStatement pstmt = conn.prepareStatement(query)) {

        pstmt.setDouble(1, nuevoPrecio);
        pstmt.setString(2, nombre);
        int filasAfectadas = pstmt.executeUpdate();

```

```

        if (filasAfectadas > 0) {
            System.out.println("Pescado actualizado correctamente.");
        } else {
            System.out.println("No se encontró el pescado con ese
nombre.");
        }
    } catch (SQLException e) {
        e.printStackTrace(); // En caso de error en la conexión o
actualización
    }
}

/**
 * Elimina un pescado de la base de datos.
 */
public void eliminarPescado(String nombre) {
    String query = "DELETE FROM pescados WHERE nombre = ?";
    try (Connection conn = DriverManager.getConnection(DB_URL,
DB_USER, DB_PASSWORD);
        PreparedStatement pstmt = conn.prepareStatement(query)) {

        pstmt.setString(1, nombre);
        int filasAfectadas = pstmt.executeUpdate();

        if (filasAfectadas > 0) {
            System.out.println("Pescado eliminado correctamente.");
        } else {

```

```

        System.out.println("No se encontró el pescado con ese
nombre.");
    }
} catch (SQLException e) {
    e.printStackTrace(); // En caso de error en la conexión o
eliminación
}
}

/**
 * Devuelve la lista de pescados disponibles en la pescadería.
 */
public List<Pescado> getPescados() {
    return pescados;
}

@Override
public String toString() {
    return "Pescadería: " + nombre + " : Pescados que tenemos: " +
pescados;
}
}

/**
 * Clase base que representa un pescado, con su nombre y precio por
kilo.
 */
class Pescado {

```

```

private String nombre; // Nombre del pescado
private double precioPorKilo; // Precio por kilo del pescado

public Pescado(String nombre, double precioPorKilo) {
    this.nombre = nombre;
    this.precioPorKilo = precioPorKilo;
}

public String getNombre() {
    return nombre;
}

public double getPrecioPorKilo() {
    return precioPorKilo;
}

@Override
public String toString() {
    return nombre + " : " + precioPorKilo + " $/kg";
}
}

/**
 * Clase derivada que hereda de Pescado. Representa un pescado
 * graso.
 */
class PescadoGraso extends Pescado {

```

```

    public PescadoGraso(String nombre, double precioPorKilo) {
        super(nombre, precioPorKilo);
    }
}

/**
 * Clase que representa a un cliente de la pescadería.
 */
class Cliente {
    private String nombre; // Nombre del cliente

    public Cliente(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return nombre;
    }
}

/**
 * Clase que representa un ticket de compra. Contiene la información del
 * cliente,
 * el pescado adquirido y la cantidad de kilos.
 */
class Ticket {

```

```

private Cliente cliente; // Cliente que realizó la compra
private Pescado pescado; // Pescado comprado
private double kilos; // Cantidad de kilos comprados

public Ticket(Cliente cliente, Pescado pescado, double kilos) {
    this.cliente = cliente;
    this.pescado = pescado;
    this.kilos = kilos;
}

/**
 * Calcula el total de la compra en base al precio por kilo y los kilos
adquiridos.
 */
public double calcularTotal() {
    return pescado.getPrecioPorKilo() * kilos;
}

@Override
public String toString() {
    return "Orden para " + cliente + ": " + kilos + " kg de " + pescado + "
: Total: " + calcularTotal() + "$";
}
}

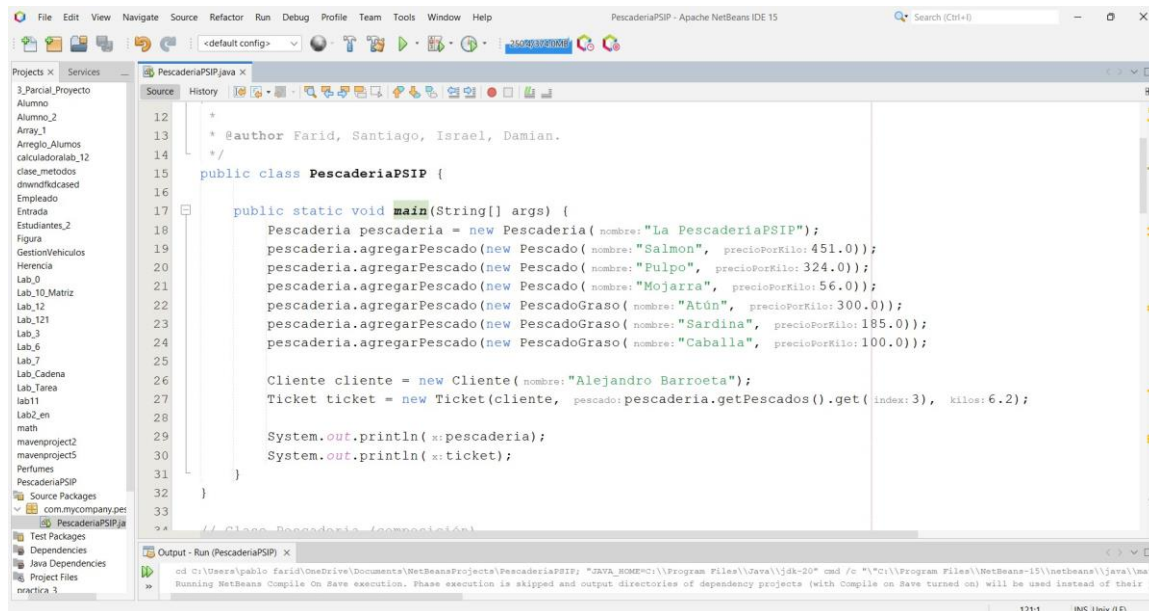
```

3.2. Configuración inicial para conexión a Base de datos

La configuración inicial para la conexión a la base de datos en este proyecto se realiza en la clase `Pescadería`, donde se definen las constantes necesarias como la URL, usuario y contraseña para conectar con una base de datos MySQL. Utilizando JDBC, se establece la conexión y se ejecutan operaciones como la carga, inserción, actualización y eliminación de registros de pescados. Esto permite gestionar los datos de forma eficiente dentro de la aplicación.

3.3. Pantallas

Class Pescadería PSIP

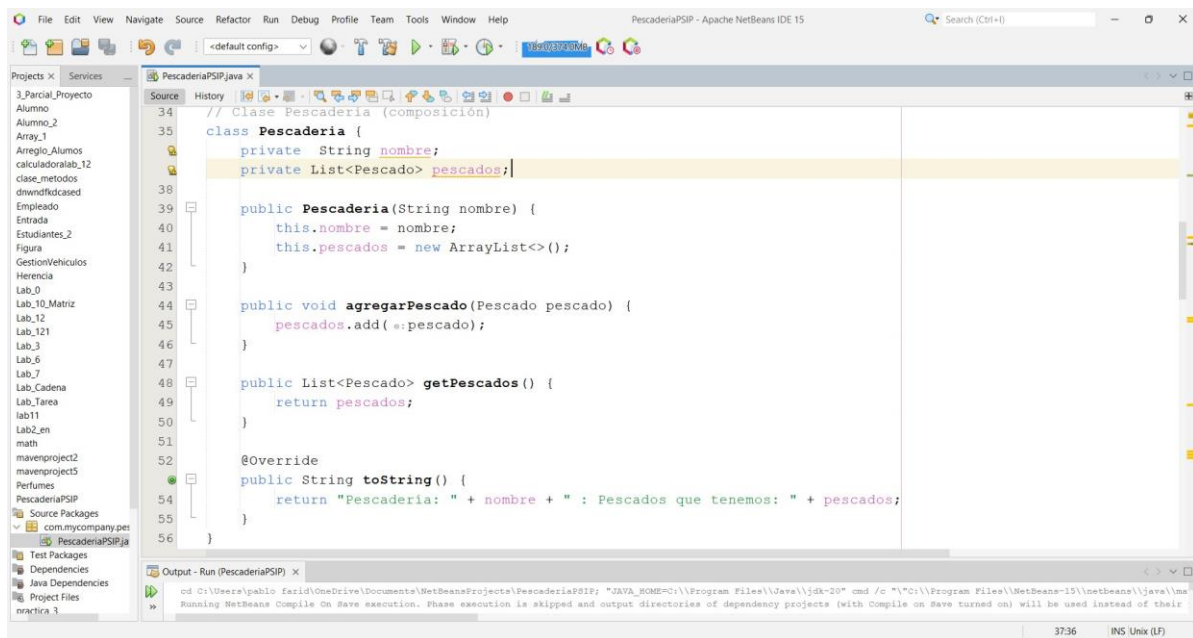


```
12  *
13  * @author Farid, Santiago, Israel, Damian.
14  */
15  public class PescaderiaPSIP {
16
17      public static void main(String[] args) {
18          Pescaderia pescaderia = new Pescaderia(nombre: "La PescaderiaPSIP");
19          pescaderia.agregarPescado(new Pescado(nombre: "Salmon", precioPorKilo: 451.0));
20          pescaderia.agregarPescado(new Pescado(nombre: "Pulpo", precioPorKilo: 324.0));
21          pescaderia.agregarPescado(new Pescado(nombre: "Mojarra", precioPorKilo: 56.0));
22          pescaderia.agregarPescado(new PescadoGraso(nombre: "Atún", precioPorKilo: 300.0));
23          pescaderia.agregarPescado(new PescadoGraso(nombre: "Sardina", precioPorKilo: 185.0));
24          pescaderia.agregarPescado(new PescadoGraso(nombre: "Caballa", precioPorKilo: 100.0));
25
26          Cliente cliente = new Cliente(nombre: "Alejandro Barroeta");
27          Ticket ticket = new Ticket(cliente, pescado: pescaderia.getPescados().get(index: 3), kilos: 6.2);
28
29          System.out.println(x: pescaderia);
30          System.out.println(x: ticket);
31      }
32  }
33
34  // Class Pescaderia (composición)
```

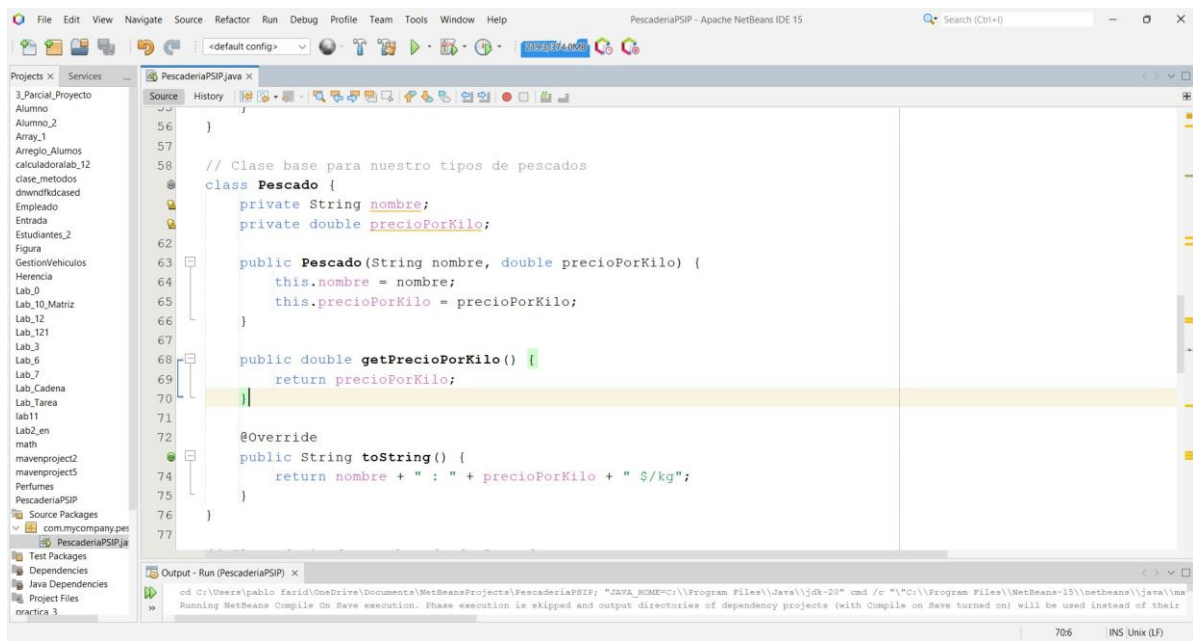
Output - Run (PescaderiaPSIP) x

```
cd C:\Users\pablo.farid\OneDrive\Documents\NetBeansProjects\PescaderiaPSIP; "JAVA_HOME=C:\Program Files\Java\jdk-20" cmd /c "%C%\Program Files\NetBeans-15\netbeans\java\ma
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their
```

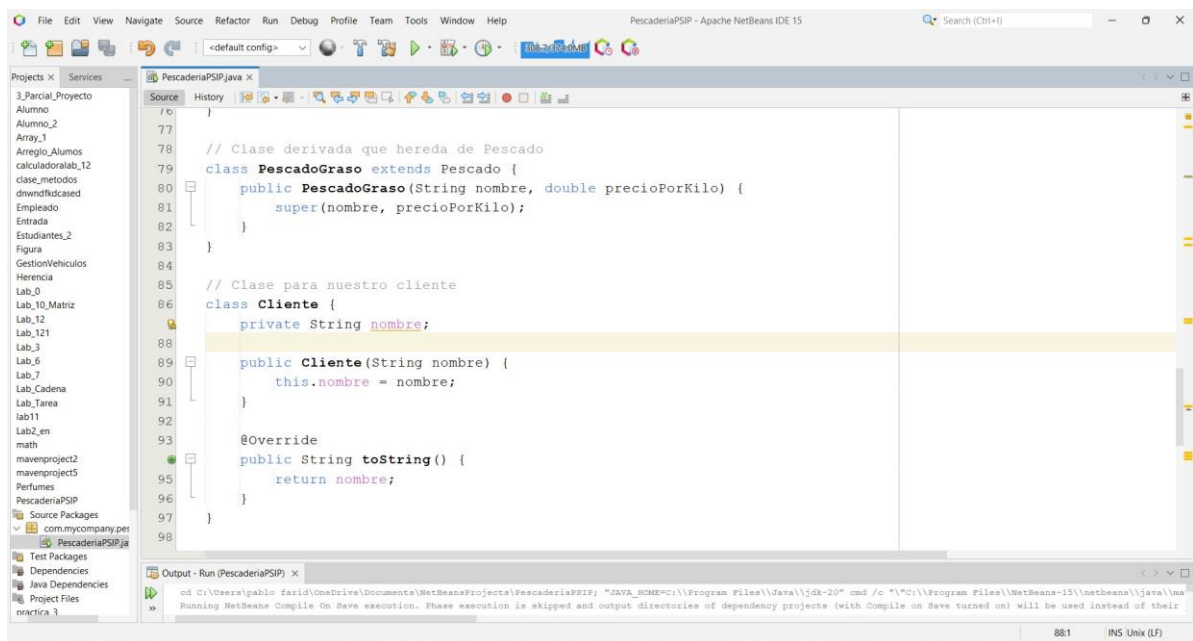

Clase Pescadería (composición)



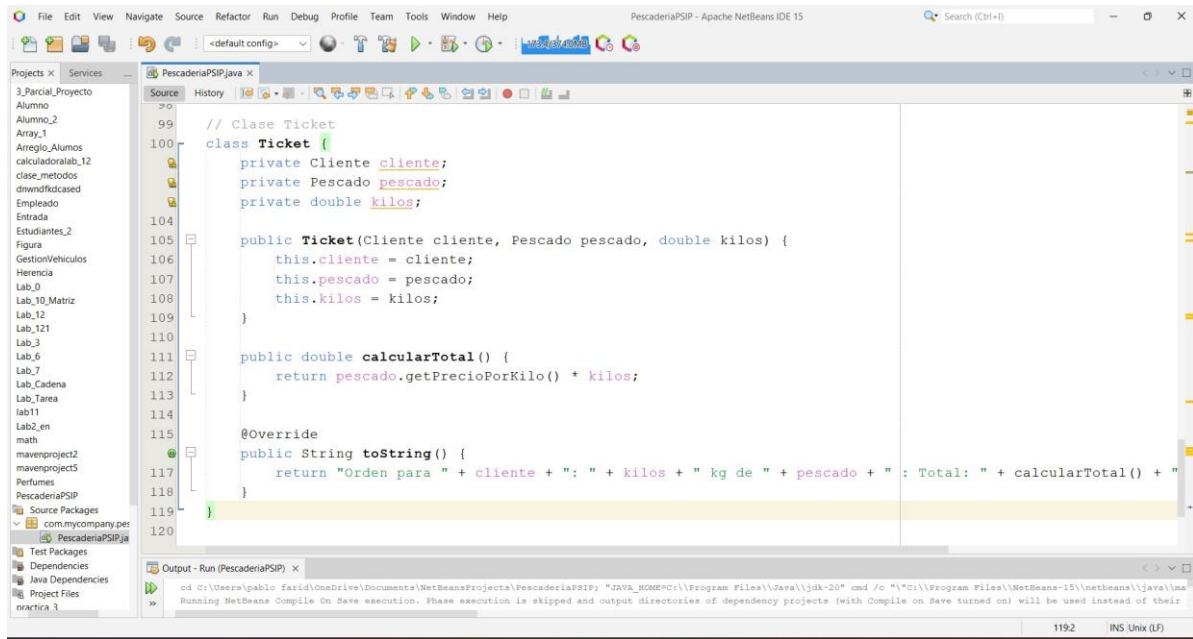
Clase base para nuestro tipos de pescados



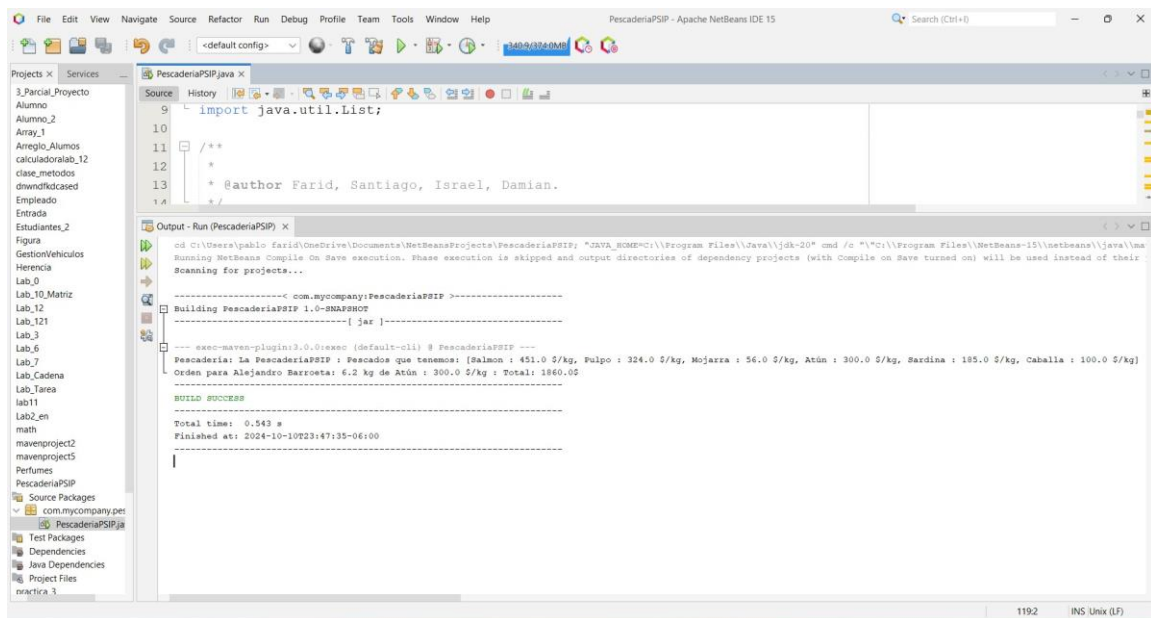
Clase derivada que hereda de Pescado y Clase para nuestro cliente

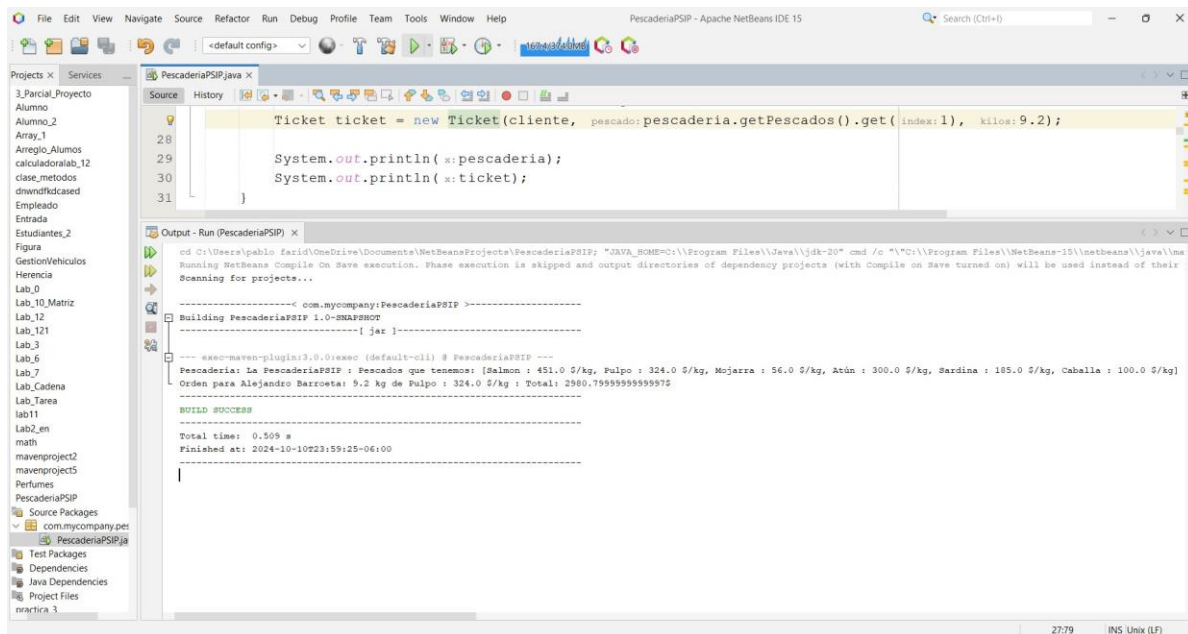


Clase Ticket



Compilación (3 imágenes)



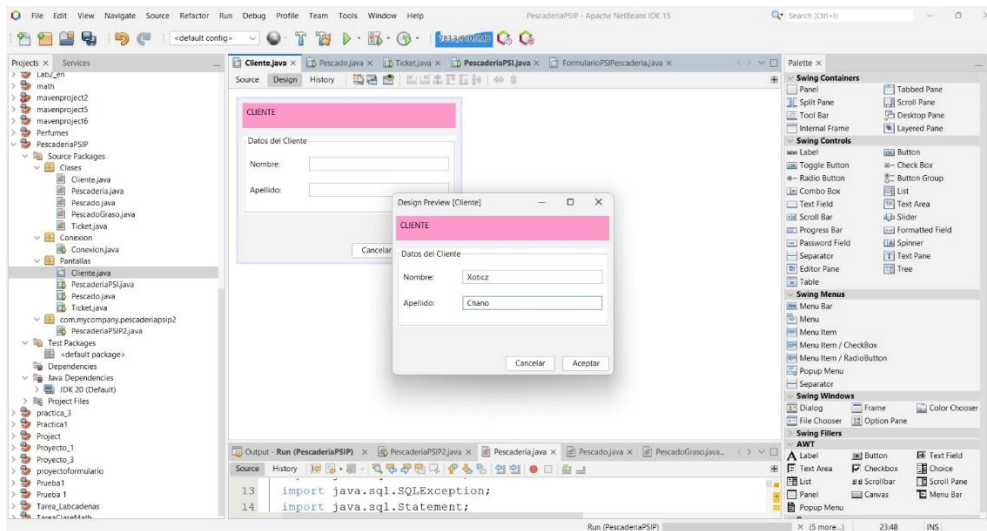


```

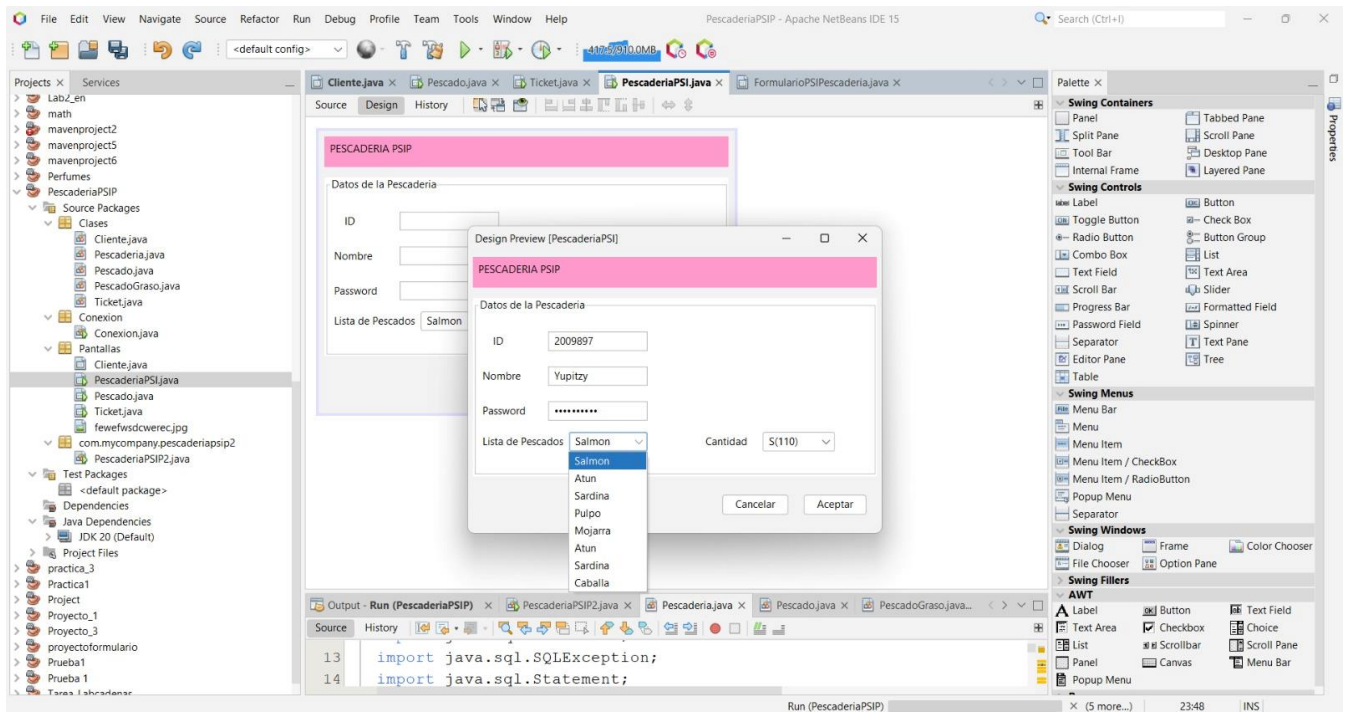
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:16)
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/pescaderia
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:230)
at com.mycompany.pescaderiapsip.Pescaderia.guardarPescadoEnBD(PescaderiaPSIP.java:94)
at com.mycompany.pescaderiapsip.Pescaderia.agregarPescado(PescaderiaPSIP.java:72)
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:19)
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/pescaderia
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:230)
at com.mycompany.pescaderiapsip.Pescaderia.guardarPescadoEnBD(PescaderiaPSIP.java:94)
at com.mycompany.pescaderiapsip.Pescaderia.agregarPescado(PescaderiaPSIP.java:72)
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:20)
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/pescaderia
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:230)
at com.mycompany.pescaderiapsip.Pescaderia.guardarPescadoEnBD(PescaderiaPSIP.java:94)
at com.mycompany.pescaderiapsip.Pescaderia.agregarPescado(PescaderiaPSIP.java:72)
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:21)
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/pescaderia
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:230)
at com.mycompany.pescaderiapsip.Pescaderia.guardarPescadoEnBD(PescaderiaPSIP.java:94)
at com.mycompany.pescaderiapsip.Pescaderia.agregarPescado(PescaderiaPSIP.java:72)
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:22)
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/pescaderia
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:230)
at com.mycompany.pescaderiapsip.Pescaderia.guardarPescadoEnBD(PescaderiaPSIP.java:94)
at com.mycompany.pescaderiapsip.Pescaderia.agregarPescado(PescaderiaPSIP.java:72)
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:23)
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/pescaderia
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:230)
at com.mycompany.pescaderiapsip.Pescaderia.guardarPescadoEnBD(PescaderiaPSIP.java:94)
at com.mycompany.pescaderiapsip.Pescaderia.agregarPescado(PescaderiaPSIP.java:72)
at com.mycompany.pescaderiapsip.PescaderiaPSIP.main(PescaderiaPSIP.java:24)
Pescaderia: La PescaderiaPSIP : Pescados que tenemos: [Salmon : 451.0 $/kg, Pulpo : 324.0 $/kg, Mojarra : 56.0 $/kg, Atún : 300.0 $/kg, Sardina : 185.0 $/kg, Caballa : 100.0 $/kg]
Orden para Alejandro Barroeta: 7.2 kg de Salmon : 451.0 $/kg : Total: 3247.2000000000003$
¿Quieres agregar un nuevo pescado? (si/no)
si
Introduce el nombre del pescado:

```

Pantalla Registro del cliente



Pantalla de Pescaderia



The screenshot displays an IDE environment for developing a Java Swing application. The Project Explorer on the left shows a project structure with packages like 'com.mycompany.pescaderiapsip2' and classes such as 'Cliente.java', 'Pescado.java', and 'Ticket.java'. The central workspace is divided into a Design view and a Source view. The Design view shows a visual representation of the application window, titled 'PESCADOS', which contains a list of fish items and their prices. The Source view shows the corresponding Java code, which includes a method for displaying the fish data in a table. The Palette on the right provides a collection of Swing components and widgets that can be added to the design.

Design Preview [Pescado]

PESCADOS	
<input type="checkbox"/> Salmon	Precio: \$451
<input type="checkbox"/> Pulpo	Precio: \$324
<input type="checkbox"/> Mojarra	Precio: \$56
<input type="checkbox"/> Atun	Precio: \$300
<input type="checkbox"/> Sardina	Precio: \$185
<input type="checkbox"/> Caballa	Precio: \$100

Source View (PescaderiaPSIP.java)

```

18  scados predeterminados a la base de datos
19  regarPescado(new Pescado( nombre: "Salmon", precioPorKilo: 451.0));
20  regarPescado(new Pescado( nombre: "Pulpo", precioPorKilo: 324.0));
21  regarPescado(new Pescado( nombre: "Mojarra", precioPorKilo: 56.0));
22  regarPescado(new PescadoGraso( nombre: "Atún", precioPorKilo: 300.0));
23  regarPescado(new PescadoGraso( nombre: "Sardina", precioPorKilo: 185.0));
24  regarPescado(new PescadoGraso( nombre: "Caballa", precioPorKilo: 100.0));
25
26  s pescados disponibles en la pescaderia
  
```

The screenshot shows the Apache NetBeans IDE interface. The main editor window displays the source code for `Ticket.java`. The code defines a `Ticket` class with attributes for `Cliente`, `Pago`, and `Datos de Compras`. A `Design Preview` window is open, showing a visual representation of the ticket form. The `Output` window shows the execution of the `PescaderiaPSIP` application, displaying a `SQLException` and a message about adding a new fish. The `Palette` window on the right lists various Swing components like labels, buttons, and text areas.

Pantalla de Base de Datos

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Server Explorer' pane shows the connection to 'LAPTOP-B3KFDC9R\SQLEXPRESS (SQL S...)'. The main pane shows the following SQL script:

```
CREATE DATABASE pescaderiaPSI2;
USE pescaderiaPSI2;

CREATE TABLE pescados (
    id INT PRIMARY KEY,
    Nombre VARCHAR(100),
    precioPorKilo INT
);

-- Insertar datos iniciales en la tabla 'pescados'
INSERT INTO pescados VALUES
(1, 'Salmon', 451.0),
(2, 'Pulpo', 324.0),
(3, 'Mojarra', 56.0),
(4, 'Atún', 300.0),
(5, 'Sardina', 185.0),
(6, 'Caballa', 100.0);
```

Below the script, the 'Results' pane shows the data inserted into the 'pescados' table:

	id	Nombre	precioPorKilo
1	1	Salmon	451
2	2	Pulpo	324
3	3	Mojarra	56
4	4	Atún	300
5	5	Sardina	185
6	6	Caballa	100

