HITO 1 PROGRAMACION

Francisco Javier Exposito Jurado

30 DE ABRIL DE 2024

CAMPUS FP

3 TRIMESTRE

INDICE

CUESTIÓN 1. POO en Java y colecciones

▪ *Diseña un ejemplo de clases en donde puedas demostrar la herencia en Java. Explica si es posible realizar herencia múltiple en Java y por qué.*

A continuación, mostrare un ejemplo en el cual tendremos la clase Perro, que esta vendrá heredada de otra clase que es la de Animal, lo que significa que Perro adquiere todos los métodos y propiedades de Animal. Esto demuestra el concepto de herencia en Java.

Java no admite herencia múltiple de clases, lo que significa que una clase no puede heredar de más de una clase directamente. Sin embargo, Java permite la implementación de múltiples interfaces.

CLASE ANIMAL

```
package Actividad1;

class Animal {
    void comer() {
        System.out.println("El animal está comiendo");
    }
}
```

CLASE PERRO HEREDADA DE ANIMAL

```
package Actividad1;

class Perro extends Animal {
    void ladrar() {
        System.out.println("El perro está ladrando");
    }
}
```

CLASE MENU

```
package Actividad1;

public class Menu {
    public static void main(String[] args) {
        Perro miPerro = new Perro();
        miPerro.comer();
        miPerro.ladrar();
    }
}
```

En el momento que tengamos estos tres códigos como podremos ver lo que debería de pasar es que en pantalla nos tendrá que mostrar las frases que pusimos de, el animal esta comiendo y el perro esta ladrando

```
El animal está comiendo...
El perro está ladrando...
```

▪ *Siguiendo el ejercicio anterior, propón un ejemplo para diferenciar sobrecarga y sobreescritura. El ejemplo funcionará en consola.*

SOBRECARGA

CLASE CALCULADORA

```java
class Calculadora {
    int sumar(int a, int b) {
        return a + b;
    }

    double sumar(double a, double b) {
        return a + b;
    }
}
```

CLASE MENU

```java
public class Menu {
    public static void main(String[] args) {
        Calculadora calc = new Calculadora();
        int sumaEnteros = calc.sumar(5, 3);
        double sumaDoubles = calc.sumar(2.5, 3.7);
        System.out.println("Suma de enteros: " + sumaEnteros);
        System.out.println("Suma de doubles: " + sumaDoubles);
    }
}
```

En este ejemplo, el método sumar está sobrecargado, lo que significa que hay múltiples métodos con el mismo nombre, pero diferentes tipos de parámetros.

La clase Calculadora tiene dos métodos llamados sumar, pero cada uno tiene una firma diferente. Uno toma dos parámetros de tipo int y devuelve un resultado de tipo int, mientras que el otro toma dos parámetros de tipo double y devuelve un resultado de tipo double.

Se está realizando una llamada sobrecargada al método sumar. En el primer caso, se llama al método que acepta dos enteros y devuelve un entero, y en el segundo caso, se llama al método que acepta dos números en punto flotante (double) y devuelve un resultado en punto flotante.

La sobrecarga de métodos permite que los desarrolladores utilicen un nombre intuitivo para métodos que realizan tareas similares, pero con diferentes tipos de datos, lo que hace que el código sea más legible y expresivo.

SOBRESCRITURA

CLASE ANIMAL

```
class Animal {
    void hacerSonido() {
        System.out.println("El animal hace un sonido...");
    }
}
```

CLASE PERRO HEREDADA DE ANIMAL

```
class Perro extends Animal {
    @Override
    void hacerSonido() {
        System.out.println("El perro ladra...");
    }
}
```

CLASE MENU

```
public class Main {
    public static void main(String[] args) {
        Animal miAnimal = new Animal();
        miAnimal.hacerSonido();  //

        Perro miPerro = new Perro();
        miPerro.hacerSonido();
    }
}
```

La sobreescritura, por otro lado, ocurre cuando se anula un método en una subclase que tiene la misma firma que un método en la clase base. Esto permite que la subclase proporcione una implementación específica de ese método.

En este ejemplo, la clase Perro hereda de la clase Animal y sobrescribe el método hacerSonido(). Cuando se llama al método hacerSonido() en una instancia de Perro, se ejecuta la implementación específica de Perro en lugar de la implementación en la clase Animal. Esto es un ejemplo de sobreescritura en Java. La anotación @Override se usa para indicar que estamos sobrescribiendo un método de la clase padre.

▪ *En Java existen varias opciones para almacenar datos en colecciones. Explica con un ejemplo qué diferencia hay entre colecciones de tipo lista, pila, cola y vector.*

LISTA

```java
import java.util.ArrayList;
import java.util.List;

public class ListaExample {
    public static void main(String[] args) {
        List<String> lista = new ArrayList<>();
        lista.add("Manzana");
        lista.add("Banana");
        lista.add("Cereza");

        System.out.println("Elementos de la lista:");
        for (String fruta : lista) {
            System.out.println(fruta);
        }
    }
}
```

Una lista en Java es una colección ordenada de elementos que permite elementos duplicados y que permite acceder a los elementos por su índice. Algunas implementaciones comunes de interfaces de lista en Java son ArrayList y LinkedList.

PILA

```java
import java.util.Stack;

public class PilaExample {
    public static void main(String[] args) {
        Stack<Integer> pila = new Stack<>();
        pila.push(1);
        pila.push(2);
        pila.push(3);

        System.out.println("Elementos de la pila:");
        while (!pila.isEmpty()) {
            System.out.println(pila.pop());
        }
    }
}
```

Una pila en Java sigue el principio de "último en entrar, primero en salir" (LIFO). Es decir, el último elemento agregado es el primero en ser eliminado. La interfaz Stack proporciona operaciones push y pop.

COLA

```java
import java.util.LinkedList;
import java.util.Queue;

public class ColaExample {
    public static void main(String[] args) {
        Queue<String> cola = new LinkedList<>();
        cola.offer("Rojo");
        cola.offer("Verde");
        cola.offer("Azul");

        System.out.println("Elementos de la cola:");
        while (!cola.isEmpty()) {
            System.out.println(cola.poll());
        }
    }
}
```

Una cola en Java sigue el principio de "primero en entrar, primero en salir" (FIFO). Es decir, el primer elemento agregado es el primero en ser eliminado. La interfaz Queue proporciona operaciones enqueue y dequeue.

VECTOR

```java
import java.util.Vector;

public class VectorExample {
    public static void main(String[] args) {
        Vector<Integer> vector = new Vector<>();
        vector.add(10);
        vector.add(20);
        vector.add(30);

        System.out.println("Elementos del vector:");
        for (int num : vector) {
            System.out.println(num);
        }
    }
}
```

Un vector en Java es similar a una lista, pero es sincronizado y tiene algunas características adicionales de legado. La clase Vector es una implementación de lista sincronizada. Se prefiere el uso de ArrayList en la mayoría de los casos debido a mejoras de rendimiento.

CUESTIÓN 2. Acceso a información con ficheros

▪ *Crea un archivo de texto con el enunciado de 10 preguntas. Desde consola muestra cada pregunta, cuando el usuario responde, se muestra la siguiente pregunta.*

Bueno en este ejercicio he decidido crear dos clases, una que sea la de crear preguntas, este nos dejara crear las 10 preguntas que queremos hacer, entonces nos empezará a salir en el programa la opción de una a una crear la preguntas.

Mientras tanto hemos creado otra clase en la cual podremos ver las preguntas que hemos creado y podremos ver las 10 preguntas que creamos.

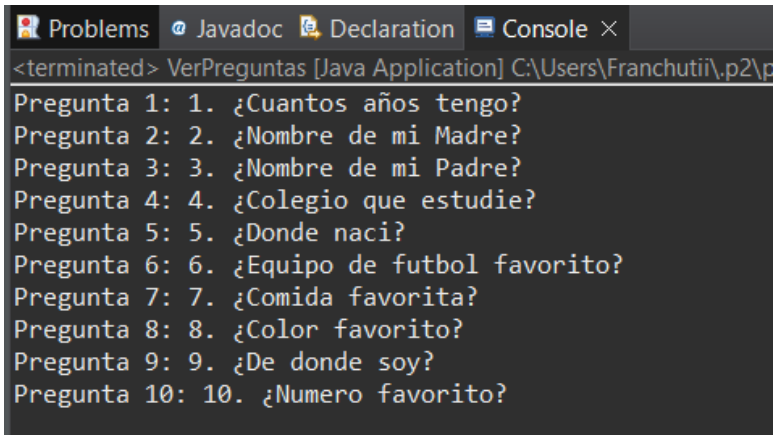
CLASE CREARPREGUNTAS

```
package Hitoo1;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class CrearPreguntas {
    public static void main(String[] args) {
        try {

            FileWriter archivoPreguntas = new FileWriter("preguntas.txt");

            Scanner scanner = new Scanner(System.in);
            for (int i = 1; i <= 10; i++) {
                System.out.print("Ingresa la pregunta " + i + ": ");
                String pregunta = scanner.nextLine();
                archivoPreguntas.write(i + ". " + pregunta + "\n");
            }

            archivoPreguntas.close();

            System.out.println("Archivo de preguntas creado exitosamente.");

        } catch (IOException e) {
            System.out.println("Error al crear el archivo de preguntas: " +
e.getMessage());
        }
    }
}
```

```
Ingresa la pregunta 1: ¿Cuantos años tengo?
Ingresa la pregunta 2: ¿Nombre de mi Madre?
Ingresa la pregunta 3: ¿Nombre de mi Padre?
Ingresa la pregunta 4: ¿Colegio que estudie?
Ingresa la pregunta 5: ¿Donde naci?
Ingresa la pregunta 6: ¿Equipo de futbol favorito?
Ingresa la pregunta 7: ¿Comida favorita?
Ingresa la pregunta 8: ¿Color favorito?
Ingresa la pregunta 9: ¿De donde soy?
Ingresa la pregunta 10: ¿Numero favorito?
Archivo de preguntas creado exitosamente.
```

CLASE VERPREGUNTAS

```java
package Hitoo1;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class VerPreguntas {
    public static void main(String[] args) {
        try {

            BufferedReader archivoPreguntas = new BufferedReader(new
FileReader("preguntas.txt"));

            String linea;
            int numeroPregunta = 1;

            while ((linea = archivoPreguntas.readLine()) != null) {
                System.out.println("Pregunta " + numeroPregunta + ": " +
linea);
                numeroPregunta++;
            }

            archivoPreguntas.close();

        } catch (IOException e) {
            System.out.println("Error al leer el archivo de preguntas: " +
e.getMessage());
        }
    }
}
```



▪ *En otro fichero de texto, o en el mismo anterior, añadimos las respuestas a las preguntas. Ahora, cuando el usuario responde, indica si la respuesta es correcta o no.*

A continuación hemos creado el apartado como el anterior de las preguntas, pero ahora con las respuestas de esas mismas preguntas, hemos creado una clase de crear respuestas en el cual podremos insertar de primeras las respuestas de las preguntas.

Luego hemos creado la clase de ver respuestas, para poder ver todas las respuestas que hemos puesto anteriormente.

CLASE CREARRESPUESTAS

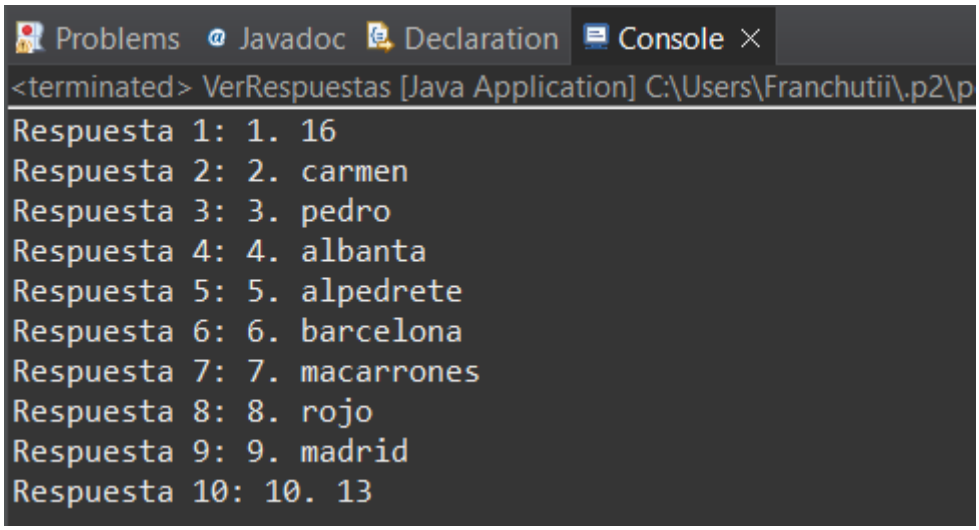
```
package Hitoo1;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class CrearRespuestas {
    public static void main(String[] args) {
        try {

            FileWriter archivoRespuestas = new FileWriter("respuestas.txt");

            Scanner scanner = new Scanner(System.in);
            for (int i = 1; i <= 10; i++) {
                System.out.print("Ingresa la respuesta " + i + ": ");
                String respuesta = scanner.nextLine();
                archivoRespuestas.write(i + ". " + respuesta + "\n");
            }

            archivoRespuestas.close();

            System.out.println("Archivo de respuestas creado exitosamente.");

        } catch (IOException e) {
```

```
            System.out.println("Error al crear el archivo de respuestas: " +
e.getMessage());
        }
    }
}
```

```
Problems  Javadoc  Declaration  Console ×
<terminated> CrearRespuestas [Java Application] C:\Users\Franchut
Ingresa la respuesta 1: 16
Ingresa la respuesta 2: carmen
Ingresa la respuesta 3: pedro
Ingresa la respuesta 4: albanta
Ingresa la respuesta 5: alpedrete
Ingresa la respuesta 6: barcelona
Ingresa la respuesta 7: macarrones
Ingresa la respuesta 8: rojo
Ingresa la respuesta 9: madrid
Ingresa la respuesta 10: 13
Archivo de respuestas creado exitosamente.
```

CLASE VERRESPUESTAS

```
package Hitoo1;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class VerRespuestas {
    public static void main(String[] args) {
        try {

            BufferedReader archivoRespuestas = new BufferedReader(new
FileReader("respuestas.txt"));

            String linea;
            int numeroRespuesta = 1;

            while ((linea = archivoRespuestas.readLine()) != null) {
                System.out.println("Respuesta " + numeroRespuesta + ": " +
linea);
                numeroRespuesta++;
            }

            archivoRespuestas.close();

        } catch (IOException e) {
            System.out.println("Error al leer el archivo de respuestas: " +
e.getMessage());
        }
    }
}
```



▪ *Muestra la puntuación obtenida por el usuario. Cada respuesta acertada suma un punto, y cada errónea resta 0.5. La nota necesaria para aprobar es un 5. Para las rutas de los ficheros, usa rutas relativas.*

Para poder responder tendremos que poner antes de la respuesta el numero de la respuesta para que nos lo pueda coger correctamente. Ejemplo: 1. 16

```
package Hitoo1;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class Cuestionario {
    public static void main(String[] args) {
        try {
            String[] preguntas = cargarArchivo("preguntas.txt");
            String[] respuestas = cargarArchivo("respuestas.txt");

            if (preguntas.length != respuestas.length) {
                System.out.println("Error: El número de preguntas y
respuestas no coincide.");
                return;
            }

            double puntuacion = 0;

            Scanner scanner = new Scanner(System.in);
            for (int i = 0; i < preguntas.length; i++) {
                System.out.println(preguntas[i]);
                String respuestaUsuario = scanner.nextLine();
                String respuestaCorrecta = respuestas[i];

                if (respuestaUsuario.equalsIgnoreCase(respuestaCorrecta)) {
                    System.out.println("¡Respuesta correcta!");
                    puntuacion += 1;
                } else {
```

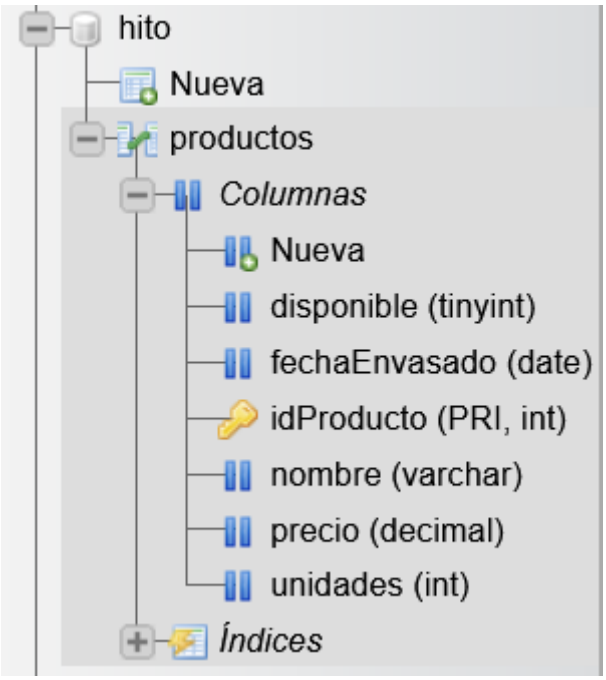
```
                    System.out.println("Respuesta incorrecta. La respuesta
correcta es: " + respuestaCorrecta);
                    puntuacion -= 0.5;
                }
            }

            System.out.println("Puntuación final: " + puntuacion);

            if (puntuacion >= 5) {
                System.out.println("¡Has aprobado!");
            } else {
                System.out.println("Lo siento, has reprobado.");
            }

        } catch (IOException e) {
            System.out.println("Error al leer los archivos: " +
e.getMessage());
        }
    }

    private static String[] cargarArchivo(String nombreArchivo) throws
IOException {
        BufferedReader lector = new BufferedReader(new
FileReader(nombreArchivo));
        String linea;
        StringBuilder contenido = new StringBuilder();
        while ((linea = lector.readLine()) != null) {
            contenido.append(linea).append("\n");
        }
        lector.close();
        return contenido.toString().split("\n");
    }
}
```

```
Cuestionario [Java Application] C:\Users\Franchutii\.p2\pool\plugins\org.eclipse.justj.o
1. ¿Cuantos años tengo?
16
Respuesta incorrecta. La respuesta correcta es: 1. 16
2. ¿Nombre de mi Madre?
2. carmen
¡Respuesta correcta!
3. ¿Nombre de mi Padre?
```

CUESTIÓN 3. Java CRUD

▪ *Gestión de base de datos relacionales. Utilizando MySQL crea una base de datos en donde dispongamos de una tabla que permita almacenar lo siguientes datos de productos (idProducto, nombre, fechaEnvasado, unidades, precio, disponible). Disponible será un valor booleano.*

```sql
CREATE DATABASE IF NOT EXISTS hito;

USE hito;

CREATE TABLE IF NOT EXISTS productos (
    idProducto INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fechaEnvasado DATE NOT NULL,
    unidades INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    disponible BOOLEAN NOT NULL
);

```



▪ *Realiza una aplicación que nos permita añadir, eliminar, actualizar y mostrar los productos. Realizar esta tarea por consola, utiliza funciones.*

CLASE MENU

```java
package Hitoo; // Define un paquete llamado Hitoo

import java.sql.*; // Importa las clases necesarias para interactuar con bases de datos SQL
import java.util.Scanner; // Importa la clase Scanner para leer la entrada del usuario

public class Menu { // Define una clase llamada Menu

    public static void main(String[] args) { // El punto de entrada de la aplicación
```

```
        try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/hito", "root", "")) { // Intenta conectarse a una base de datos MySQL llamada "hito" en localhost, usando el usuario "root" y una contraseña vacía

            ProductoDAO productoDAO = new ProductoDAO(con); // Crea un objeto ProductoDAO para interactuar con la base de datos
            Scanner scanner = new Scanner(System.in); // Crea un objeto Scanner para leer la entrada del usuario

            while (true) { // Un bucle infinito que muestra el menú y espera la entrada del usuario

                System.out.println("\nMenú:"); // Imprime el menú en la consola
                System.out.println("1. Añadir producto");
                System.out.println("2. Ver productos");
                System.out.println("3. Eliminar producto");
                System.out.println("4. Actualizar producto");
                System.out.println("5. Salir");
                System.out.print("Seleccione una opción: "); // Pide al usuario que seleccione una opción

                int opcion = scanner.nextInt(); // Lee la opción seleccionada por el usuario
                scanner.nextLine(); // Limpia el buffer del scanner

                switch (opcion) { // Comienza a ejecutar diferentes acciones basadas en la opción seleccionada

                    case 1: // Si el usuario elige "Añadir producto"
                        // Solicita al usuario que ingrese los detalles del nuevo producto
                        System.out.print("Ingrese el nombre del producto: ");
                        String nombre = scanner.nextLine();
                        System.out.print("Ingrese la fecha de envasado (YYYY-MM-DD): ");
                        String fechaEnvasado = scanner.nextLine();
                        System.out.print("Ingrese la cantidad de unidades: ");
                        int unidades = scanner.nextInt();
                        System.out.print("Ingrese el precio: ");
                        double precio = scanner.nextDouble();
                        scanner.nextLine(); // Limpia el buffer del scanner
                        System.out.print("¿Está disponible? (Y/N): ");
                        boolean disponible = scanner.nextBoolean();
                        scanner.nextLine(); // Limpia el buffer del scanner

                        // Crea un nuevo objeto Producto con los detalles ingresados
                        Producto nuevoProducto = new Producto(nombre, fechaEnvasado, unidades, precio, disponible);
                        // Inserta el nuevo producto en la base de datos
                        boolean insercionExitosa = productoDAO.insertarProducto(nuevoProducto);
                        // Imprime un mensaje indicando si la inserción fue exitosa
                        if (insercionExitosa) {
```

```
                            System.out.println("Producto añadido
correctamente.");
                        } else {
                            System.out.println("Error al añadir el
producto.");
                        }
                        break;

                    case 2: // Si el usuario elige "Ver productos"
                        // Muestra todos los productos almacenados en la base
de datos
                        System.out.println("\nProductos:");
                        productoDAO.mostrarProductos();
                        break;

                    case 3: // Si el usuario elige "Eliminar producto"
                        // Solicita al usuario que ingrese el ID del producto
a eliminar
                        System.out.print("Ingrese el ID del producto a
eliminar: ");
                        int idEliminar = scanner.nextInt();
                        // Intenta eliminar el producto de la base de datos
                        boolean eliminacionExitosa =
productoDAO.eliminarProducto(idEliminar);
                        // Imprime un mensaje indicando si la eliminación fue
exitosa
                        if (eliminacionExitosa) {
                            System.out.println("Producto eliminado
correctamente.");
                        } else {
                            System.out.println("Error al eliminar el
producto.");
                        }
                        break;

                    case 4: // Si el usuario elige "Actualizar producto"
                        // Solicita al usuario que ingrese el ID del producto
a actualizar
                        System.out.print("Ingrese el ID del producto a
actualizar: ");
                        int idActualizar = scanner.nextInt();
                        scanner.nextLine(); // Limpia el buffer del scanner

                        // Solicita al usuario que ingrese los nuevos
detalles del producto
                        System.out.print("Ingrese el nuevo nombre del
producto: ");
                        String nuevoNombre = scanner.nextLine();
                        System.out.print("Ingrese la nueva fecha de envasado
(YYYY-MM-DD): ");
                        String nuevaFechaEnvasado = scanner.nextLine();
                        System.out.print("Ingrese la nueva cantidad de
unidades: ");
                        int nuevasUnidades = scanner.nextInt();
                        System.out.print("Ingrese el nuevo precio: ");
                        double nuevoPrecio = scanner.nextDouble();
                        scanner.nextLine(); // Limpia el buffer del scanner
                        System.out.print("¿Está disponible? (true/false): ");
                        boolean nuevoDisponible = scanner.nextBoolean();
```

```
                        scanner.nextLine(); // Limpia el buffer del scanner

                        // Crea un nuevo objeto Producto con los nuevos
detalles
                        Producto productoActualizado = new
Producto(nuevoNombre, nuevaFechaEnvasado, nuevasUnidades, nuevoPrecio,
nuevoDisponible);
                        productoActualizado.setIdProducto(idActualizar); //
Establece el ID del producto a actualizar
                        // Actualiza el producto en la base de datos
                        boolean actualizacionExitosa =
productoDAO.actualizarProducto(productoActualizado);
                        // Imprime un mensaje indicando si la actualización
fue exitosa
                        if (actualizacionExitosa) {
                            System.out.println("Producto actualizado
correctamente.");
                        } else {
                            System.out.println("Error al actualizar el
producto.");
                        }
                        break;

                    case 5: // Si el usuario elige "Salir"
                        System.out.println("Saliendo del programa..."); //
Imprime un mensaje de despedida
                        return; // Sale del bucle y termina el programa

                    default: // Si el usuario elige una opción inválida
                        System.out.println("Opción inválida."); // Imprime un
mensaje de error
                }
            }

        } catch (SQLException e) { // Captura cualquier excepción de tipo
SQLException
            e.printStackTrace(); // Imprime el seguimiento de la pila para
identificar el error
        }
    }
}
```

CLASE PRODUCTO

```
package Hitoo; // Define un paquete llamado Hitoo

public class Producto { // Define una clase llamada Producto

    // Atributos de la clase Producto
    private int idProducto; // El identificador único del producto
    private String nombre; // El nombre del producto
    private String fechaEnvasado; // La fecha en la que el producto fue
envasado
    private int unidades; // El número de unidades disponibles
    private double precio; // El precio del producto
    private boolean disponible; // Indica si el producto está disponible o no

    // Constructor vacío
```

```
    public Producto() {}

    // Constructor con parámetros para inicializar los atributos
    public Producto(String nombre, String fechaEnvasado, int unidades, double
precio, boolean disponible) {
        this.nombre = nombre; // Asigna el nombre del producto
        this.fechaEnvasado = fechaEnvasado; // Asigna la fecha de envasado
del producto
        this.unidades = unidades; // Asigna el número de unidades disponibles
del producto
        this.precio = precio; // Asigna el precio del producto
        this.disponible = disponible; // Asigna la disponibilidad del
producto
    }

    // Métodos getters y setters para acceder y modificar los atributos de la
clase

    public int getIdProducto() {
        return idProducto;
    }

    public void setIdProducto(int idProducto) {
        this.idProducto = idProducto;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getFechaEnvasado() {
        return fechaEnvasado;
    }

    public void setFechaEnvasado(String fechaEnvasado) {
        this.fechaEnvasado = fechaEnvasado;
    }

    public int getUnidades() {
        return unidades;
    }

    public void setUnidades(int unidades) {
        this.unidades = unidades;
    }

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public boolean isDisponible() {
```

```
        return disponible;
    }

    public void setDisponible(boolean disponible) {
        this.disponible = disponible;
    }
}
```

CLASE PRODUCTODAO

```
package Hitoo; // Define un paquete llamado Hitoo

import java.sql.*; // Importa las clases necesarias para interactuar con
bases de datos SQL

public class ProductoDAO { // Define una clase llamada ProductoDAO

    private Connection con; // Establece una conexión a la base de datos

    public ProductoDAO(Connection con) { // Constructor que recibe una
conexión como parámetro
        this.con = con; // Asigna la conexión a la variable de instancia
    }

    public boolean insertarProducto(Producto producto) { // Método para
insertar un nuevo producto en la base de datos
        String query = "INSERT INTO productos (nombre, fechaEnvasado,
unidades, precio, disponible) VALUES (?, ?, ?, ?, ?)"; // Consulta SQL para
insertar un producto
        try (PreparedStatement stmt = con.prepareStatement(query)) { //
Intenta preparar la consulta
            // Establece los valores de los parámetros en la consulta
            stmt.setString(1, producto.getNombre());
            stmt.setString(2, producto.getFechaEnvasado());
            stmt.setInt(3, producto.getUnidades());
            stmt.setDouble(4, producto.getPrecio());
            stmt.setBoolean(5, producto.isDisponible());
            int filasInsertadas = stmt.executeUpdate(); // Ejecuta la
consulta y devuelve el número de filas afectadas
            return filasInsertadas > 0; // Devuelve true si se insertó al
menos una fila, false en caso contrario
        } catch (SQLException e) { // Captura cualquier excepción de tipo
SQLException
            e.printStackTrace(); // Imprime el seguimiento de la pila para
identificar el error
            return false; // Devuelve false en caso de error
        }
    }

    public void mostrarProductos() { // Método para mostrar todos los
productos almacenados en la base de datos
        String query = "SELECT * FROM productos"; // Consulta SQL para
seleccionar todos los productos
        try (Statement stmt = con.createStatement(); // Crea un objeto
Statement para ejecutar la consulta
             ResultSet rs = stmt.executeQuery(query)) { // Ejecuta la
consulta y obtiene un conjunto de resultados
```

```
            while (rs.next()) { // Itera sobre cada fila del conjunto de
resultados
                // Imprime los detalles de cada producto
                System.out.println("ID: " + rs.getInt("idProducto") +
                                   ", Nombre: " + rs.getString("nombre") +
                                   ", Fecha Envasado: " +
rs.getString("fechaEnvasado") +
                                   ", Unidades: " + rs.getInt("unidades") +
                                   ", Precio: " + rs.getDouble("precio") +
                                   ", Disponible: " +
rs.getBoolean("disponible"));
            }
        } catch (SQLException e) { // Captura cualquier excepción de tipo
SQLException
            e.printStackTrace(); // Imprime el seguimiento de la pila para
identificar el error
        }
    }

    public boolean eliminarProducto(int idProducto) { // Método para eliminar
un producto de la base de datos por su ID
        String query = "DELETE FROM productos WHERE idProducto = ?"; //
Consulta SQL para eliminar un producto
        try (PreparedStatement stmt = con.prepareStatement(query)) { //
Intenta preparar la consulta
            stmt.setInt(1, idProducto); // Establece el valor del parámetro
en la consulta
            int filasEliminadas = stmt.executeUpdate(); // Ejecuta la
consulta y devuelve el número de filas afectadas
            return filasEliminadas > 0; // Devuelve true si se eliminó al
menos una fila, false en caso contrario
        } catch (SQLException e) { // Captura cualquier excepción de tipo
SQLException
            e.printStackTrace(); // Imprime el seguimiento de la pila para
identificar el error
            return false; // Devuelve false en caso de error
        }
    }

    public boolean actualizarProducto(Producto producto) { // Método para
actualizar un producto en la base de datos
        String query = "UPDATE productos SET nombre = ?, fechaEnvasado = ?,
unidades = ?, precio = ?, disponible = ? WHERE idProducto = ?"; // Consulta
SQL para actualizar un producto
        try (PreparedStatement stmt = con.prepareStatement(query)) { //
Intenta preparar la consulta
            // Establece los valores de los parámetros en la consulta
            stmt.setString(1, producto.getNombre());
            stmt.setString(2, producto.getFechaEnvasado());
            stmt.setInt(3, producto.getUnidades());
            stmt.setDouble(4, producto.getPrecio());
            stmt.setBoolean(5, producto.isDisponible());
            stmt.setInt(6, producto.getIdProducto());
            int filasActualizadas = stmt.executeUpdate(); // Ejecuta la
consulta y devuelve el número de filas afectadas
            return filasActualizadas > 0; // Devuelve true si se actualizó al
menos una fila, false en caso contrario
        } catch (SQLException e) { // Captura cualquier excepción de tipo
SQLException
```

```
            e.printStackTrace(); // Imprime el seguimiento de la pila para
identificar el error
            return false; // Devuelve false en caso de error
        }
    }
}
```

COMPROBACION DEL FUNCIONAMIENTO

AGREGAR Y VER

```
Problems  Javadoc  Declaration  Console
Menu (1) [Java Application] C:\Users\Franchutii\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v2024

Menú:
1. Añadir producto
2. Ver productos
3. Eliminar producto
4. Actualizar producto
5. Salir
Seleccione una opción: 1
Ingrese el nombre del producto: mueble
Ingrese la fecha de envasado (YYYY-MM-DD): 1899-10-25
Ingrese la cantidad de unidades: 4
Ingrese el precio: 12
¿Está disponible? (true/false): true
Producto añadido correctamente.

Menú:
1. Añadir producto
2. Ver productos
3. Eliminar producto
4. Actualizar producto
5. Salir
Seleccione una opción: 2

Productos:
ID: 3, Nombre: mueble, Fecha Envasado: 1899-10-25, Unidades: 4, Precio: 12.0, Disponible: true
```

←T→	idProducto	nombre	fechaEnvasado	unidades	precio	disponible
☐ Editar Copiar Borrar	3	mueble	1899-10-25	4	12.00	1

ELIMINAR

```
Menú:
1. Añadir producto
2. Ver productos
3. Eliminar producto
4. Actualizar producto
5. Salir
Seleccione una opción: 3
Ingrese el ID del producto a eliminar: 3
Producto eliminado correctamente.
```

idProducto	nombre	fechaEnvasado	unidades	precio	disponible

Operaciones sobre los resultados de la consulta

ACTUALIZAR

```
Menú:
1. Añadir producto
2. Ver productos
3. Eliminar producto
4. Actualizar producto
5. Salir
Seleccione una opción: 4
Ingrese el ID del producto a actualizar: 4
Ingrese el nuevo nombre del producto: silla
Ingrese la nueva fecha de envasado (YYYY-MM-DD): 1895-03-14
Ingrese la nueva cantidad de unidades: 8
Ingrese el nuevo precio: 45
¿Está disponible? (true/false): true
Producto actualizado correctamente.
```

←T→	idProducto	nombre	fechaEnvasado	unidades	precio	disponible
☐ Editar Copiar Borrar	4	silla	1895-03-14	8	45.00	1