

HITO 1 3T Programación

David Ronaldo Garcia



CUESTIÓN 1. POO en Java y colecciones

CriaturasMagicas.java

```
package Cuestion1;
public class CriaturasMagicas {
    String nombre;

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

    public void volar() {
        System.out.println("¡" + nombre + " está volando!");
    }
}

class Dragon extends CriaturasMagicas {
    public Dragon(String nombre) {
        super(nombre);
    }

    public void lanzarFuego() {
        System.out.println("¡" + nombre + " lanza fuego!");
    }
}

class Hada extends CriaturasMagicas {
    public Hada(String nombre) {
        super(nombre);
    }

    public void lanzarPolvoDeHadas() {
        System.out.println("¡" + nombre + " lanza polvo de hadas!");
    }
}
```

Main.java

```
package Cuestion1;
public class Main {
    public static void main(String[] args) {
        Dragon smaug = new Dragon("El escamas");
        smaug.volar();
        smaug.lanzarFuego();

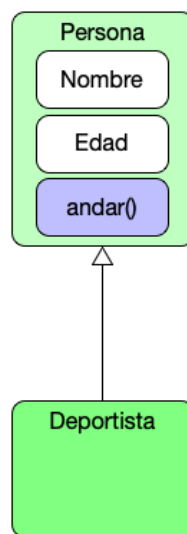
        Hada tinkerbelle = new Hada("Campanilla");
        tinkerbelle.volar();
        tinkerbelle.lanzarPolvoDeHadas();
    }
}
```

Resultado:

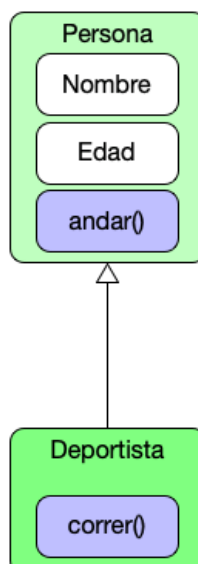
```
¡El escamas está volando!  
¡El escamas lanza fuego!  
¡Campanilla está volando!  
¡Campanilla lanza polvo de hadas!
```

Explicación:

Java no soporta la herencia múltiple ya que una clase concreta solo puede tener una clase padre. Lamentablemente esta no es una respuesta totalmente correcta o final. Vamos a verlo más a detalle ya que se trata de un tema bastante interesante. Supongamos que tenemos la clase Persona con nombre y edad y el método andar .



De esta clase podemos heredar la clase de Deportista que añadirá el método correr y si nos apetece sobrescribirá el método andar para que el deportista ande más rápido hasta aquí todo es muy muy clásico.



CriaturaMagica.java

```
package Cuestion1;
//Clase base (superclase)
class CriaturaMagica {
    String nombre;
    // Constructor de la superclase
    public CriaturaMagica(String nombre) {
        this.nombre = nombre;
    }
    // Sobrecarga del método emitirSonido()
    public void emitirSonido() {
        System.out.println(nombre + " emite un sonido misterioso.");
    }
    // Sobrecarga del método emitirSonido() con un parámetro adicional
    public void emitirSonido(String intensidad) {
        System.out.println(nombre + " emite un sonido " + intensidad + ".");
    }
}
//Subclase que hereda de CriaturaMagica
class Dragon extends CriaturaMagica {
    // Constructor de la subclase Dragon
    public Dragon(String nombre) {
        super(nombre);
    }
    // Sobreescritura del método emitirSonido() para el dragón
    @Override
    public void emitirSonido() {
        System.out.println(nombre + " ruge con furia.");
    }
}
//Subclase adicional
class Hada extends CriaturaMagica {
    // Constructor de la subclase Hada
    public Hada(String nombre) {
        super(nombre);
    }
    // Sobreescritura del método emitirSonido() para el hada
    @Override
    public void emitirSonido() {
        System.out.println(nombre + " canta una melodía encantadora.");
    }
}
```

Main.java

```
package Cuestion1;
public class Main {
    public static void main(String[] args) {
        CriaturaMagica criatura = new CriaturaMagica("Criatura misteriosa");
        criatura.emitirSonido();
        criatura.emitirSonido("fuerte");

        Dragon smaug = new Dragon("El peligroso");
        smaug.emitirSonido();

        Hada tinkerbelle = new Hada("Campanilla");
        tinkerbelle.emitirSonido();
    }
}
```

```
Criatura misteriosa emite un sonido fuerte.
El peligroso ruge con furia.
Campanilla canta una melodía encantadora.
```

- Almacenar datos en colecciones.

Lista:

```
package Cuestion1;
import java.util.LinkedList;
import java.util.Queue;
public class Main {
    public static void main(String[] args) {
        Queue<String> cola = new LinkedList<>();
        cola.offer("Enero");
        cola.offer("Febrero");
        cola.offer("Marzo");
        cola.offer("Abril");
        System.out.println("Meses:");
        while (!cola.isEmpty()) {
            System.out.println(cola.poll());
        }
    }
}
```

Resultado:

```
Meses:
Enero
Febrero
Marzo
Abril
<
```

Explicación:

Una lista es una colección ordenada que permite duplicar elementos.

Los elementos en una lista están indexados por posición.

Las implementaciones comunes incluyen **ArrayList** y **LinkedList**.

Pila:

```
package Cuestion1;
import java.util.Stack;
public class Main {
    public static void main(String[] args) {
        Stack<String> pila = new Stack<>();
        pila.push("Rojo");
        pila.push("Azul");
        pila.push("Verde");

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

Resultado:

```
Colores:
Verde
Azul
Rojo
```

Explicación:

Una pila es una colección LIFO (**Last-In-First-Out**), lo que significa que el último elemento agregado es el primero en ser eliminado.

Las operaciones principales son **push()** para agregar un elemento y **pop()** para eliminar el último elemento agregado.

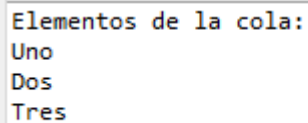
La clase Stack proporciona una implementación de pila en Java.

Cola:

```
package Cuestion1;
import java.util.LinkedList;
import java.util.Queue;
public class Main {
    public static void main(String[] args) {
        Queue<String> cola = new LinkedList<>();
        cola.offer("Uno");
        cola.offer("Dos");
        cola.offer("Tres");

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

Resultado:



```
Elementos de la cola:
Uno
Dos
Tres
```

Explicación:

Una cola es una colección FIFO (**First-In-First-Out**), lo que significa que el primer elemento agregado es el primero en ser eliminado.

Las operaciones principales son **offer()** para agregar un elemento y **poll()** para eliminar y devolver el primer elemento.

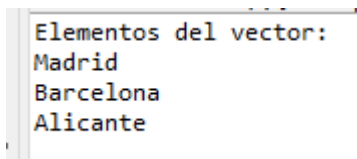
La interfaz Queue proporciona una abstracción de cola en Java.

Vector:

```
package Cuestion1;
import java.util.Vector;
public class Main {
    public static void main(String[] args) {
        Vector<String> vector = new Vector<>();
        vector.add("Madrid");
        vector.add("Barcelona");
        vector.add("Alicante");

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

Resultado:



```
Elementos del vector:
Madrid
Barcelona
Alicante
```

Explicación:

Un vector es una colección dinámica similar a un arreglo que se puede redimensionar automáticamente.

Los vectores son seguros para la concurrencia (**thread-safe**), pero menos eficientes que **ArrayList**.

La clase Vector proporciona una implementación de vector en Java.

CUESTIÓN 2. Acceso a información con ficheros

Main.java

```
package CuestionII;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        String nombreArchivo = "preguntasHito.txt";

        try (BufferedReader br = new BufferedReader(new
FileReader(nombreArchivo))) {
            Scanner scanner = new Scanner(System.in);
            String pregunta;

            while ((pregunta = br.readLine()) != null) {
                System.out.println(pregunta);
                scanner.nextLine();
            }

            System.out.println("¡Eso es todo! Gracias por responder.");
        } catch (IOException e) {
            System.err.println("Error al leer el archivo: " + e.getMessage());
        }
    }
}
```

Resultado:

ficheroHito.txt

```
1. ¿Cuál equipo ganó la Copa Mundial de la FIFA 2018?  
Francia  
2. ¿Quién es el máximo goleador en la historia de la Liga de Campeones de la UEFA?  
Cristiano Ronaldo  
3. ¿En qué país se juega el Clásico Español entre el Barcelona y el Real Madrid?  
España  
4. ¿Cuántos jugadores conforman un equipo de fútbol en el campo durante un partido?  
22  
5. ¿Quién es conocido como "La Pulga" y es considerado uno de los mejores futbolistas de todos los tiempos?  
Messi  
6. ¿Cuál es el estadio de fútbol más grande del mundo por capacidad de espectadores?  
Maracanã  
7. ¿Qué selección nacional ha ganado la Copa Mundial de la FIFA en más ocasiones?  
Brasil  
8. ¿Cuál es el equipo de fútbol más exitoso en la historia de la Liga Premier de Inglaterra?  
Manchester United  
9. ¿Qué país ganó la primera Copa Mundial de la FIFA en 1930?  
Uruguay  
10. ¿Cuál es el jugador de fútbol que ha ganado el mayor número de premios Balón de Oro?  
Messi  
¡Eso fue todo! Gracias por responder.
```

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.

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.

Respuestas.java

```
package CuestionII;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class Respuestas {
    public static void main(String[] args) {
        String preguntasArchivo = "preguntasHito.txt";
        String respuestasArchivo = "respuestasHito.txt";

        try (
            BufferedReader preguntasReader = new BufferedReader(new
            FileReader(preguntasArchivo));
            BufferedReader respuestasReader = new BufferedReader(new
            FileReader(respuestasArchivo));
            Scanner scanner = new Scanner(System.in)
        ) {
            int puntuacion = 0;
            String pregunta;
            String respuesta;

            while ((pregunta = preguntasReader.readLine()) != null && (respuesta =
            respuestasReader.readLine()) != null) {
                System.out.println(pregunta);
                String respuestaUsuario = scanner.nextLine().trim();

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

```

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

    System.out.println("¡Eso fue todo! Tu puntuación final es: " + puntuacion);
    if (puntuacion >= 5) {
        System.out.println("¡Felicidades, has aprobado!");
    } else {
        System.out.println("Lo siento, no has alcanzado la nota mínima para
aprobar.");
    }
} catch (IOException e) {
    System.err.println("Error al leer el archivo: " + e.getMessage());
}
}
}

```

Resultado:

```

1. ¿Cuál equipo ganó la Copa Mundial de la FIFA 2018?
Francia
¡Respuesta correcta!
2. ¿Quién es el máximo goleador en la historia de la Liga de Campeones de la UEFA?
Cristiano Ronaldo
¡Respuesta correcta!
3. ¿En qué país se juega el Clásico Español entre el Barcelona y el Real Madrid?
España
¡Respuesta correcta!
4. ¿Cuántos jugadores conforman un equipo de fútbol en el campo durante un partido?
22
Respuesta incorrecta. La respuesta correcta es: 11
5. ¿Quién es conocido como "La Pulga" y es considerado uno de los mejores futbolistas de todos los tiempos?
Lionel Messi
¡Respuesta correcta!
6. ¿Cuál es el estadio de fútbol más grande del mundo por capacidad de espectadores?
Maracanã
Respuesta incorrecta. La respuesta correcta es: Estadio Rungrado 1º de Mayo
7. ¿Qué selección nacional ha ganado la Copa Mundial de la FIFA en más ocasiones?
Brasil
¡Respuesta correcta!
8. ¿Cuál es el equipo de fútbol más exitoso en la historia de la Liga Premier de Inglaterra?
Manchester United
¡Respuesta correcta!
9. ¿Qué país ganó la primera Copa Mundial de la FIFA en 1930?
Uruguay
¡Respuesta correcta!
10. ¿Cuál es el jugador de fútbol que ha ganado el mayor número de premios Balón de Oro?
Lionel Messi
¡Respuesta correcta!
¡Eso fue todo! Tu puntuación final es: 6
¡Felicidades, has aprobado!

```

CUESTIÓN 3. Java CRUD

Creamos la base de datos:

```
1 • CREATE DATABASE IF NOT EXISTS TiendaHito;
2 • USE TiendaHito;
3 • CREATE TABLE IF NOT EXISTS Productos (
4     idProducto INT AUTO_INCREMENT PRIMARY KEY,
5     nombre VARCHAR(255) NOT NULL,
6     fechaEnvasado DATE NOT NULL,
7     unidades INT NOT NULL,
8     precio DECIMAL(10, 2) NOT NULL,
9     disponible BOOLEAN NOT NULL
10 );
11
```

Main.java

```
package CuestionIII;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/TiendaHito", "root", "");
            ProductoDAO productoDAO = new ProductoDAO(con);
            Scanner scanner = new Scanner(System.in);
            int opcion;
            do {
                System.out.println("Menú:");
                System.out.println("1. Añadir producto");
                System.out.println("2. Mostrar productos");
                System.out.println("3. Actualizar producto");
                System.out.println("4. Eliminar producto");
                System.out.println("5. Salir");
                System.out.print("Ingresa la opción deseada: ");
                opcion = scanner.nextInt();
                switch (opcion) {
                    case 1:
                        scanner.nextLine(); // Limpiar el buffer del scanner
                        System.out.print("Ingresa el nombre del producto: ");
                        String nombre = scanner.nextLine();
                        System.out.print("Ingresa la fecha de envasado (YYYY-MM-DD): ");
                        String fechaEnvasado = scanner.nextLine();
                        System.out.print("Ingresa la cantidad de unidades: ");
```

```

        int unidades = scanner.nextInt();
        System.out.print("Ingrese el precio: ");
        double precio = scanner.nextDouble();
        System.out.print("¿Está disponible? (true/false): ");
        boolean disponible = scanner.nextBoolean();
        Producto nuevoProducto = new Producto(nombre, fechaEnvasado, unidades,
precio, disponible);
        if (productoDAO.insertarProducto(nuevoProducto)) {
            System.out.println("Producto agregado correctamente.");
        } else {
            System.out.println("Error al agregar el producto.");
        }
        break;
    case 2:
        System.out.println("Lista de productos:");
        productoDAO.mostrarProductos();
        break;
    case 3:
        System.out.print("Ingrese el ID del producto a actualizar: ");
        int idActualizar = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer del scanner
        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): ");
        fechaEnvasado = scanner.nextLine(); // Asignamos el valor a la variable
        System.out.print("Ingrese la nueva cantidad de unidades: ");
        int nuevasUnidades = scanner.nextInt();
        System.out.print("Ingrese el nuevo precio: ");
        double nuevoPrecio = scanner.nextDouble();
        System.out.print("¿Está disponible? (true/false): ");
        boolean nuevoDisponible = scanner.nextBoolean();
        Producto productoActualizar = new Producto(nuevoNombre, fechaEnvasado,
nuevasUnidades, nuevoPrecio, nuevoDisponible);
        productoActualizar.setIdProducto(idActualizar);
        if (productoDAO.actualizarProducto(productoActualizar)) {
            System.out.println("Producto actualizado correctamente.");
        } else {
            System.out.println("Error al actualizar el producto.");
        }
        break;
    case 4:
        System.out.print("Ingrese el ID del producto a eliminar: ");
        int idEliminar = scanner.nextInt();
        if (productoDAO.eliminarProducto(idEliminar)) {
            System.out.println("Producto eliminado correctamente.");
        } else {
            System.out.println("Error al eliminar el producto.");
        }
        break;
    case 5:
        System.out.println("Saliendo del programa...");
        break;

```

```

        default:
            System.out.println("Opción inválida. Por favor, seleccione una opción válida.");
        }
    } while (opcion != 5);
    // Cerrar el scanner y la conexión a la base de datos
    scanner.close();
    con.close();
} catch (ClassNotFoundException | SQLException e) {
    e.printStackTrace();
}
}
}
}

```

Se conecta a una base de datos MySQL para gestionar productos de una tienda. Utiliza la API JDBC para interactuar con la base de datos. La clase Main contiene la lógica principal de la aplicación, incluyendo un menú con diferentes opciones que el usuario puede seleccionar. Dependiendo de la opción elegida, se ejecutan diferentes acciones como añadir, mostrar, actualizar o eliminar productos. Cada opción del menú invoca métodos de la clase ProductoDAO para realizar operaciones en la base de datos, como inserción, actualización, eliminación y consulta de productos. La clase Producto representa la estructura de datos de un producto y la clase ProductoDAO contiene métodos para interactuar con la tabla de productos en la base de datos.

ProductoDAO:

```
package CuestionIII;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class ProductoDAO {
    private Connection conexion;
    public ProductoDAO(Connection conexion) {
        this.conexion = conexion;
    }
    public boolean insertarProducto(Producto producto) {
        try {
            PreparedStatement statement = conexion.prepareStatement(
                "INSERT INTO Productos (nombre, fechaEnvasado, unidades, precio, disponible)
VALUES (?, ?, ?, ?, ?)");
            statement.setString(1, producto.getNombre());
            statement.setDate(2, Date.valueOf(producto.getFechaEnvasado()));
            statement.setInt(3, producto.getUnidades());
            statement.setDouble(4, producto.getPrecio());
            statement.setBoolean(5, producto.isDisponible());
            return statement.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    public boolean actualizarProducto(Producto producto) {
        try {
            PreparedStatement statement = conexion.prepareStatement(
                "UPDATE Productos SET nombre = ?, fechaEnvasado = ?, unidades = ?, precio = ?,
disponible = ? WHERE idProducto = ?");
            statement.setString(1, producto.getNombre());
            statement.setDate(2, Date.valueOf(producto.getFechaEnvasado()));
            statement.setInt(3, producto.getUnidades());
            statement.setDouble(4, producto.getPrecio());
            statement.setBoolean(5, producto.isDisponible());
            statement.setInt(6, producto.getIdProducto());
            return statement.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    public boolean eliminarProducto(int idProducto) {
        try {
            PreparedStatement statement = conexion.prepareStatement("DELETE FROM Productos
WHERE idProducto = ?");
            statement.setInt(1, idProducto);
            return statement.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```



```

    }
}
public void mostrarProductos() {
    try {
        Statement statement = conexion.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM Productos");
        while (resultSet.next()) {
            System.out.println(
                resultSet.getInt("idProducto") + ", " +
                resultSet.getString("nombre") + ", " +
                resultSet.getDate("fechaEnvasado") + ", " +
                resultSet.getInt("unidades") + ", " +
                resultSet.getDouble("precio") + ", " +
                resultSet.getBoolean("disponible")
            );
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

Define una clase llamada ProductoDAO, que se encarga de realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en una tabla de productos en una base de datos MySQL. Utiliza la API JDBC para interactuar con la base de datos. La clase contiene métodos para insertar, actualizar, eliminar y mostrar productos en la base de datos. Cada método utiliza consultas SQL preparadas para ejecutar las operaciones correspondientes en la base de datos.

Producto.java

```
package CuestionIII;
import java.time.LocalDate;
public class Producto {
    private int idProducto;
    private String nombre;
    private LocalDate fechaEnvasado;
    private int unidades;
    private double precio;
    private boolean disponible;
    public Producto(String nombre, String fechaEnvasado, int unidades, double precio, boolean
disponible) {
        this.nombre = nombre;
        this.fechaEnvasado = LocalDate.parse(fechaEnvasado);
        this.unidades = unidades;
        this.precio = precio;
        this.disponible = disponible;
    }
    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 LocalDate getFechaEnvasado() {
        return fechaEnvasado;
    }
    public void setFechaEnvasado(LocalDate 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;
    }
}

```

La clase tiene seis atributos privados: **idProducto**, **nombre**, **fechaEnvasado**, **unidades**, **precio**, y **disponibilidad**.

El constructor Producto toma cinco parámetros para inicializar los atributos del producto: **nombre**, **fechaEnvasado**, **unidades**, **precio**, y **disponible**. La fecha de envasado se convierte de String a **LocalDate** usando el método **LocalDate.parse**.

Se proporcionan métodos getter y setter para acceder y modificar los atributos del producto. Los getters son **getIdProducto**, **getNombre**, **getFechaEnvasado**, **getUnidades**, **getPrecio**, y **isDisponible**. Los setters son **setIdProducto**, **setNombre**, **setFechaEnvasado**, **setUnidades**, **setPrecio**, y **setDisponible**, respectivamente.

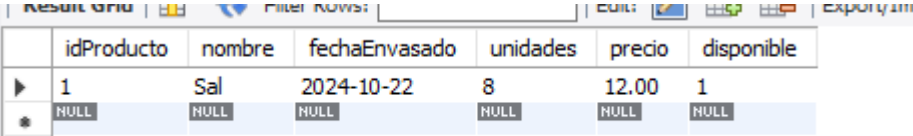
Resultado:

Añadir producto:

```

Menú:
1. Añadir producto
2. Mostrar productos
3. Actualizar producto
4. Eliminar producto
5. Salir
Ingrese la opción deseada: 1
Ingrese el nombre del producto: Sal
Ingrese la fecha de envasado (YYYY-MM-DD): 2024-10-22
Ingrese la cantidad de unidades: 8
Ingrese el precio: 12
¿Está disponible? (true/false): true
Producto agregado correctamente.
Menú:

```



	idProducto	nombre	fechaEnvasado	unidades	precio	disponible
▶	1	Sal	2024-10-22	8	12.00	1
*	NULL	NULL	NULL	NULL	NULL	NULL

Mostrar productos:

```
Menú:
1. Añadir producto
2. Mostrar productos
3. Actualizar producto
4. Eliminar producto
5. Salir
Ingrese la opción deseada: 2
Lista de productos:
1, Sal, 2024-10-22, 8, 12.0, true
Menú:
```

Actualizar producto:

```
Menú:
1. Añadir producto
2. Mostrar productos
3. Actualizar producto
4. Eliminar producto
5. Salir
Ingrese la opción deseada: 3
Ingrese el ID del producto a actualizar: 1
Ingrese el nuevo nombre del producto: Azucar
Ingrese la nueva fecha de envasado (YYYY-MM-DD): 2025-11-12
Ingrese la nueva cantidad de unidades: 14
Ingrese el nuevo precio: 10
¿Está disponible? (true/false): false
Producto actualizado correctamente.
Menú:
```

Result Grid

Filter Rows:

Edit:

Export/Imp

	idProducto	nombre	fechaEnvasado	unidades	precio	disponible
▶	1	Azucar	2025-11-12	14	10.00	0
*	NULL	NULL	NULL	NULL	NULL	NULL

Eliminar producto:

```
Menú:
1. Añadir producto
2. Mostrar productos
3. Actualizar producto
4. Eliminar producto
5. Salir
Ingrese la opción deseada: 1
Ingrese el nombre del producto: prueba
Ingrese la fecha de envasado (YYYY-MM-DD): 2025-10-10
Ingrese la cantidad de unidades: 8
Ingrese el precio: 40
¿Está disponible? (true/false): false
Producto agregado correctamente.
Menú:
1. Añadir producto
2. Mostrar productos
3. Actualizar producto
4. Eliminar producto
5. Salir
Ingrese la opción deseada: 4
Ingrese el ID del producto a eliminar: 1
Producto eliminado correctamente.
Menú:
```

	idProducto	nombre	fechaEnvasado	unidades	precio	disponible
▶	2	prueba	2025-10-10	8	40.00	0
*	NULL	NULL	NULL	NULL	NULL	NULL

“Entregado en github,cada carpeta numerada con su respectiva cuestión,los archivos .txt dentro de su cuestión (cuestionII) , la estructura de la base de datos también y este documento en formato pdf en el Classroom”

Bibliografía:

Cuestión 1:

<https://www.arquitecturajava.com/java-herencia-multiple-y-sus-opciones/>

https://www.grycap.upv.es/gmolto/docs/eda/EDA_Tema_8_gmolto.pdf

Cuestión 2:

<https://doctortecnologico.com/web/cuales-son-las-formas-de-acceder-a-un-fichero-java/>

Cuestión 3:

[Ejercicio realizado en clase](#)