AA3.Diseño de diagrama de clases I

DAM - Programación Orientada a Objetos

Gonzalo Nieto Maneu - UOC

1.Preguntas del caso práctico **1**
2. Diagrama de clases: **2**
3.Implementación **3**
4.Javadoc **5**
5.Video mostrando la ejecución: **5**
6. Problemas encontrados **5**
Autoevaluación **6**

1.Preguntas del caso práctico

En este caso práctico se pueden identificar tres enumeraciones distintas.

1.¿Qué es una enumeración?
Es un tipo de dato que consiste en un conjunto fijo de constantes en ellas se representan las opciones y variantes posibles. Este tipo de dato se utiliza cuando se sabe de antemano cuáles son todas las posibles opciones que puede tomar la variable.

2.¿Cuáles son las tres enumeraciones?
TipoTablero: En ella representamos los diferentes tipos de tablero. Ejemplo: Aglomerado,Contrachapado…

ColorBarniz: En ella representamos los diferentes colores de barniz. Ejemplo:Incoloro,Caoba,Nogal…

TipoArticulo: En ella representamos los diferentes tipos de artículos.Ejemplo:Estantería,Mesa,Silla.

3.¿Cómo se representan en el diagrama de clases?
Se representan como un caso de valores fijos constantes , estos valores son las opciones posibles. Esas clases de enumeración se representan en cursiva su nombre y los valores de las constantes enumerados abajo.

4.¿Cómo se codifican en Java?
```
public enum TipoTablero {
   AGLOMERADO,
   CONTRACHAPADO,
   MDF
}
public enum ColorBarniz {
   INCOLORO,
   CAOBA,
   NOGAL
}
public enum TipoArticulo {
   ESTANTERIA,
   MESA,
   SILLA,
   ARMARIO
}
```

2. Diagrama de clases:

He representado un diagrama de clases inicial especificando el nombre de la clase , los atributos que lo componen y sus métodos según las indicaciones que nos ha proporcionado el enunciado.

WoodShops:
Atributos:
tiendas: ArrayList<Tienda>
proveedores: ArrayList<Proveedor>
Métodos:
añadirTienda(ti: Tienda): void
listarProductos(tienda: Tienda,tipoProducto: TipoProducto): void
mostrarStockProducto(codigo: String)

Tienda:
Atributos:
nombre: String
almacenes: ArrayList<Almacen>
Métodos:
añadirAlmacen(almacen: Almacen):void
listarProductos(tipoProducto:TipoProducto): void
obtenerStock(codigo: String): void

Almacén:
Atributos:
productos: HashMap<String, Producto>|
Métodos:
añadirProducto(p: Producto): void
listarProductos(tipoProducto:
TipoProducto): void
obtenerStock(codigo: String): void

Producto:
Atributos:
codigo: String
descripcion: String
proveedor: Proveedor
precioVenta: double
stock: int
Métodos:
Producto(codigo: String, descripcion: String,proveedor: Proveedor, precioVenta:
double, stock: int): void
getStock(): int
setStock(stock: int): void
Proveedor
Atributos:
- nif: String

- nombre: String

Métodos:

Proveedor(nif: String,nombre: String): void

TipoProducto (Enumeración)

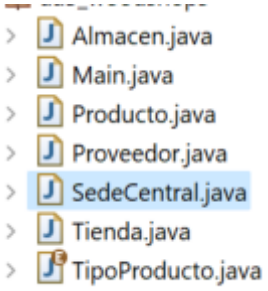
TABLERO
BARNIZ
ARTICULO

Relacionaría WoodShops y Tienda, Tienda y Almacén , Producto y Proveedor.

3.Implementación

A medida que he ido programando el ejercicio he decidido hacer unos cambios respecto al diagrama de clases original que he hecho en el apartado anterior. Uno de los cambios más notables ha sido crear la clase "Main.java" donde he querido solamente utilizarlo para implementar el método "main" y la hacer la llamada a las funciones principales y la clase "WoodShops" la he transformado en "SedeCentral.java". En "SedeCentral.java" reside varias funciones entre ellas la función de "ejecutarMenu" y también es ahí donde creo los objetos con el método "cargarDatosIniciales". Esto ha sido algunos de los cambios respecto al diagrama original anteriormente realizado, he creído que de esta manera el ejercicio me quedaba más entendible y depurado.

3.1 Clases:



3.2 Implementar un método para la carga de datos iniciales:

```java
/**
 * Método para cargar datos iniciales.
 */
public void cargarDatosIniciales() {
    // Creamos algunos proveedores
    Proveedor proveedor1 = new Proveedor("001", "Proveedor1");
    Proveedor proveedor2 = new Proveedor("002", "Proveedor2");

    // Añadimos los proveedores a la sede central
    añadirProveedor(proveedor1);
    añadirProveedor(proveedor2);

    // Creamos algunas tiendas
    Tienda tienda1 = new Tienda("Tienda1");
    Tienda tienda2 = new Tienda("Tienda2");

    // Añadimos las tiendas a la sede central
    añadirTienda(tienda1);
    añadirTienda(tienda2);

    // Creamos algunos productos de ejemplo
    Producto producto1 = new Producto("001", "Tablero Aglomerado", proveedor1, 10.50, 50, TipoProducto.TABLERO);
    Producto producto2 = new Producto("002", "Barniz Incoloro", proveedor2, 5.20, 100, TipoProducto.BARNIZ);
    Producto producto3 = new Producto("003", "Mesa de Madera", proveedor1, 75.00, 20, TipoProducto.ARTICULO);

    Producto producto4 = new Producto("004", "Tablero Aglomerado2", proveedor1, 10.50, 50, TipoProducto.TABLERO);
    Producto producto5 = new Producto("005", "Barniz Incoloro2", proveedor2, 5.20, 100, TipoProducto.BARNIZ);
    Producto producto6 = new Producto("006", "Mesa de Madera2", proveedor1, 75.00, 20, TipoProducto.ARTICULO);


    // Creamos almacenes y añadimos productos a las tiendas
    Almacen almacen1 = new Almacen();
    almacen1.añadirProducto(producto1);
    almacen1.añadirProducto(producto2);
    almacen1.añadirProducto(producto4);
    almacen1.añadirProducto(producto5);
    almacen1.añadirProducto(producto6);
    tienda1.añadirAlmacen(almacen1);

    Almacen almacen2 = new Almacen();
    almacen2.añadirProducto(producto3);
    almacen1.añadirProducto(producto4);
    almacen1.añadirProducto(producto5);
    almacen1.añadirProducto(producto6);
    tienda2.añadirAlmacen(almacen2);
}
```

3.3 Crear las opciones del menú

```java
public static void ejecutarMenu(SedeCentral sedeCentral) {
    Scanner scanner = new Scanner(System.in);
    int opcion;
    do {
        System.out.println("\nMenú:");
        System.out.println("1. Listar productos de una tienda");
        System.out.println("2. Mostrar stock de un producto");
        System.out.println("3. Anyadir un nuevo producto");
        System.out.println("0. Salir");
        System.out.print("Ingrese la opción: ");
        opcion = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer del scanner

        switch (opcion) {
            case 1:
                listarProductosDeTienda(sedeCentral);
                break;
            case 2:
                System.out.print("Ingrese el código del producto: ");
                String codigoProducto = scanner.nextLine();
                sedeCentral.mostrarStockDeProducto(codigoProducto);
                break;
            /*case 3:
                anyadirNuevoProducto(sedeCentral);
                break;*/
            case 0:
                System.out.println("Saliendo del programa...");
                break;
            default:
                System.out.println("Opción no válida. Intente de nuevo.");
        }
    } while (opcion != 0);
    scanner.close();
}
```

4.Javadoc

Para generar documentación Javadoc es necesario comentar y añadir las etiquetas correspondientes al código.
Etiquetas:

- @param: Descripción de un parámetro del método.
- @return: Descripción del valor del return.
- @throws: Descripción de las excepciones que puede lanzar el método.
- @see: Referencia a otros elementos (clases, métodos ...).

javadoc -d carpeta_destino MiClase.java

Para generar el javadoc utilizaré la siguiente comanda:

```
javadoc -d doc -sourcepath src -subpackages aa3_woodshops
```

5.Video mostrando la ejecución:

Enlace: https://youtu.be/dEb7_FhpX8Y

6. Problemas encontrados

El principal problema que me he encontrado ha sido a la hora de implementar que el usuario pueda añadir un nuevo producto , si es verdad que podemos crear objetos manualmente, pero he querido implementar que se pueda hacer de forma interactiva por consola para que el usuario pueda crear fácilmente un producto , el caso es que después de haberle dedicado un extenso tiempo a intentar realizar esta implementación al final no he sido capaz de hacerlo, he probado de escribir este método de diferentes maneras , le he dado varias y varias vueltas pero al final no he sido capaz de hacerlo correctamente… He dejado algunos de estos intentos comentados en la práctica , se puede encontrar en la clase “SedeCentral.java”, también la opción del menú de “Añadir producto” se encuentra comentada en esta misma clase , dentro del método “ejecutarMenu”.

Otro problema que he tenido ha sido con los scanners, todo el programa va bien pero a la hora de salir del menú me sale una excepción de tipo scanner. Si dejo los scanners sin cerrar me sale warning en el archivo explicándome que tengo que cerrar los scanners después de usarlos (es lo que he hecho para quitar los 2 warnings) pero el programa me peta al salir de él , si no cierro los scanners en las funciones el problema funciona perfectamente , lo único malo que aparecen los dos warnings en el fichero.

Autoevaluación

FP056_AA3

Criterios	Calificaciones				Puntos
Realización de diagramas de clases a partir de las especificaciones	**3 pts** Realiza diagramas de clases a partir de las especificaciones de forma optimizada, siendo capaz de proponer diversos diseños indicando las mejoras que representa cada uno.	**2,25 pts** Realiza diagramas de clases a partir de las especificaciones	**1,5 pts** Realiza diagramas de clases a partir de las especificaciones con algún error.	**0,75 pts** Presenta dificultades en la realización de diagramas de clases a partir de las especificaciones	3 puntos
Conocimiento de la sintaxis, estructura y componentes típicos de una clase	**3 pts** Conoce la sintaxis, estructura y componentes de una clase, así como las nomenclaturas estándar utilizadas en POO y la organización correcta de los mismos.	**2,25 pts** Conoce la sintaxis, estructura y componentes típicos de una clase.	**1,5 pts** Conoce con algunos errores la sintaxis, estructura y componentes de una clase.	**0,75 pts** Presenta dificultades en el conocimiento de la sintaxis, estructura y componentes de una clase.	3 puntos
Definición de clases con sus atributos y métodos	**3 pts** Define las clases con sus atributos y métodos de forma optimizada.	**2,25 pts** Define las clases con sus atributos y métodos.	**1,5 pts** Define las clases con sus atributos y métodos con algún error.	**0,75 pts** Presenta dificultades en la definición de clases con sus atributos y métodos.	3 puntos
Creación de constructores	**3 pts** Crea constructores con sobrecarga.	**2,25 pts** Crea constructores.	**1,5 pts** Crea constructores con algún error.	**0,75 pts** Presenta dificultades en la creación de constructores.	3 puntos
Interpretación del significado de diagramas de clases	**4 pts** Interpreta el significado del diagrama de clases y su desarrollo. Es capaz de proponer mejoras en el diseño OO.	**3 pts** Interpreta el significado del diagrama de clases.	**2 pts** Interpreta el significado del diagrama de clases con algún error.	**1 pts** Presenta dificultades en la interpretación del significado del diagrama de clases.	4 puntos
Generación de código a partir de un diagrama de clases	**4 pts** Genera el código adecuado a partir de un diagrama de clases, siendo capaz de escoger la mejor opción posible en la implementación.	**3 pts** Genera el código adecuado a partir de un diagrama de clases.	**2 pts** Genera código a partir de un diagrama de clases con algún error.	**1 pts** Presenta dificultades en la generación del código a partir de un diagrama de clases.	4 puntos
					Puntos totales: 20