**Proyecto Tienda de café.**

La famosa tienda de café “duCOFFEE” ha encargado un sistema de facturación para reemplazar el actual. La Sirena Verde sólo expende el mejor café, en **dos variedades**: **Tostado Suave** y **Tostado Intenso**. Los clientes pueden elegir entre **tres agregados a sus cafés**: **Leche,** **Crema Batida** y **Jarabe de Vainilla**, y **combinarlos** como quieran. Una vez ingresada la bebida y los agregados, el **sistema calculará el costo** y **emitirá el ticket**.

**Se pide:**

* Implementar la solución
* Acompañar el código de un diagrama UML con la solución.

**Solución:**

**UML**

Table

Description automatically generated with medium confidence

**CODIGO**:

**package** Ejercicio\_1;

**public** **class** ProgramaPrincipal {

**public** **static** **void** main(String[] args) {

Producto Cafe1 = **new** Combo();

Cafe1.setNombre("Pedido 1: ");

Producto Cafe2 = **new** Combo();

Cafe2.setNombre("Pedido 2: ");

Producto CafeSuave = **new** ProductoConcreto();

CafeSuave.setNombre("Cafe Suave");

CafeSuave.setPrecio(145);

Producto CafeIntenso = **new** ProductoConcreto();

CafeIntenso.setNombre("Cafe Suave");

CafeIntenso.setPrecio(132);

Producto Leche = **new** ProductoConcreto();

Leche.setNombre("Leche");

Leche.setPrecio(5);

Producto CremaBatida = **new** ProductoConcreto();

CremaBatida.setNombre("Crema Batidata");

CremaBatida.setPrecio(8);

Producto JarabedeVainilla = **new** ProductoConcreto();

JarabedeVainilla.setNombre("Jarabe de Vainilla");

JarabedeVainilla.setPrecio(7);

//Armar producto:

Cafe1.agregar(CafeIntenso);

Cafe1.agregar(Leche);

Cafe1.mostrar();

System.***out***.println("-----------------------------------------");

Cafe2.agregar(CafeSuave);

Cafe2.agregar(CremaBatida);

Cafe2.mostrar();

}

}

**package** Ejercicio\_1;

**public** **class** ProductoConcreto **extends** Producto {

**public** **void** agregar(Producto producto) {

**throw** **new** UnsupportedOperationException("No es posible procesar esta operacion.");

}

**public** **void** mostrar() {

System.***out***.println("Adicional: " + getNombre() + " Precio: " + getPrecio());

}

}

**package** Ejercicio\_1;

**public** **abstract** **class** Producto {

**private** String Nombre;

**private** **double** precio;

**public** String getNombre() {

**return** Nombre;

}

**public** **void** setNombre(String nombre) {

Nombre = nombre;

}

**public** **double** getPrecio() {

**return** precio;

}

**public** **void** setPrecio(**double** precio) {

**this**.precio = precio;

}

**public** **abstract** **void** agregar(Producto producto);

**public** **abstract** **void** mostrar();

}

**package** Ejercicio\_1;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Combo **extends** Producto {

**private** ArrayList productos = **new** ArrayList();

**public** **void** agregar(Producto producto) {

productos.add(producto);

}

**public** **void** mostrar() {

System.***out***.println("Combo: " + getNombre());

Iterator iterator = productos.iterator();

**while** (iterator.hasNext()) {

Producto element = (Producto) iterator.next();

element.mostrar();

}

System.***out***.println("Total " + getNombre()+ " :" + getPrecio());

}

**public** **double** getPrecio() {

**double** total = 0;

Iterator iterator = productos.iterator();

**while** (iterator.hasNext()) {

Producto element = (Producto) iterator.next();

total+=element.getPrecio();

}

**return** total;

}

**public** **void** setPrecio(**double** precio) {

**throw** **new** UnsupportedOperationException("No es posible realizar este combo");

}

}

Graphical user interface, text, application

Description automatically generated🡪 Ejecución.

**Ejercicio 2**

Se debe implementar el módulo **de navegación de archivos para** un novedoso sistema operativo hecho en Java que promete revolucionar el mercado. Como es sabido, los archivos son entidades que permanecen en el disco rígido u otro soporte y tienen un nombre y un tamaño. A su vez, es posible agrupar archivos en directorios o “carpetas”. Las carpetas tienen prácticamente las mismas características que los archivos: tienen un nombre y un tamaño que corresponde a la suma de todos los archivos que tienen dentro.

**Se pide:**

* Implementar una solución que permita tratar a archivos y directorios por igual, mostrar su nombre y su tamaño.
* Acompañar el código de un diagrama UML con la solución.

**Solución:**

**UML**

A screenshot of a computer

Description automatically generated with medium confidence

**CODIGO**:

**package** Ejercico\_2;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Directorio Directorio = **new** Directorio("C:/Diseño e Sistemas/MODULO12",120);

Archivo Archivo = **new** Archivo("Ejercicio numero 2.JAVA", 20);

Directorio.mostrarInfo();

Archivo.mostrarInfo();

System.***out***.println("------------------------------------");

DiscoRigido DiscoRigido = **new** DiscoRigido(Directorio);

DiscoRigido.Guardar(Directorio,Archivo);

DiscoRigido.mostrarInfo();

System.***out***.println("------------------------------------");

Archivo Archivo2 = **new** Archivo("Ejercicio numero 3.JAVA", 30);

DiscoRigido.Guardar(Directorio,Archivo2);

DiscoRigido.mostrarInfo();

}

}

**package** Ejercico\_2;

**public** **abstract** **class** Navegacion {

**private** **int** tamanio = 0;

**public** **int** getTamanio() {

**return** tamanio;

}

**public** **void** setTamanio(**int** tamanio) {

**this**.tamanio = tamanio;

}

**public** **abstract** **void** mostrarInfo();

}

**package** Ejercico\_2;

**public** **abstract** **class** Decorador **extends** Navegacion {

**private** Navegacion Navegacion;

**public** Decorador(Navegacion Navegacion) {

**this**.Navegacion = Navegacion;

}

**public** Navegacion getNavegacion() {

**return** Navegacion;

}

**public** **void** setNavegacion(Navegacion Navegacion) {

**this**.Navegacion = Navegacion;

}

}

**package** Ejercico\_2;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** DiscoRigido **extends** Decorador {

**private** ArrayList lista = **new** ArrayList();

**public** DiscoRigido(Navegacion Navegacion) {

**super**(Navegacion);

}

**public** **void** Guardar(Directorio Directorio, Archivo Archivo) {

lista.add(Directorio.getDirectorio()+"/"+Archivo.getArchivo());

getNavegacion().setTamanio(getNavegacion().getTamanio() + Archivo.getTamanio());

}

**public** **void** mostrarInfo() {

Iterator Iterator = lista.iterator();

**while** (Iterator.hasNext()) {

String elemento = (String)Iterator.next();

System.***out***.println("El nombre del archivo es: " + elemento);

System.***out***.println("El Tamaño Total es: "+getNavegacion().getTamanio()+"MB");

}

}

}

**package** Ejercico\_2;

**public** **class** Directorio **extends** Navegacion {

**private** String Nombre;

**public** Directorio(String Nombre,**int** tamanio) {

**this**.Nombre = Nombre;

setTamanio(tamanio);

}

**public** String getDirectorio() {

**return** Nombre;

}

**public** **void** mostrarInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Nomre del directorio a utilizar: " + Nombre );

}

}

**package** Ejercico\_2;

**public** **class** Archivo **extends** Navegacion {

**private** String Nombre;

**public** Archivo(String Nombre,**int** tamanio) {

**this**.Nombre = Nombre;

setTamanio(tamanio);

}

**public** String getArchivo() {

**return** Nombre;

}

**public** **void** mostrarInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Nomre del Archivo a utilizar: " + Nombre);

}

}