|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Guadalupe Lizeth Parrales Romay |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 04 |
| *No de Práctica(s):* | 6 |
| *Integrante(s):* | Dávila Ortega Jesús Eduardo - No. Cuenta: 317199860  Díaz Hernández Marcos Bryan - No. Cuenta: 317027253  Pareja Ávila Emiliano - No. Cuenta: 317081345  Vázquez Zavala Oliver Alexis - No. Cuenta: 317202263 |
| *No. de Equipo de cómputo empleado:* |  |
| *No. de Lista o Brigada:* | 12,14, 30, 37 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | 14 de noviembre del 2020 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_

***Objetivo:***

*Organizar adecuadamente las clases según su funcionalidad o propósito bajo un namespace o paquete.*

***Introducción:***

***La Herencia:***

*La herencia o relación es-un, es la relación existente entre dos clases: una es la derivada que se crea a partir de otra ya existente, denominada base, la nueva hereda de la ya existente. Como la clase base y la clase derivada tienen código y datos comunes, es evidente que si se crea la clase derivada de modo independiente, se duplicaría mucho de lo que ya se escribió para la clase base; entonces la herencia en la clase derivada es una ampliación de la base pues aquella dispone de sus atributos y métodos propios más los atributos y métodos heredados, para la herencia java dispone de la palabra reservada* ***extends*** *para crear clases derivadas o clases que son extensión de otra, de modo que la nueva hereda todos los miembros de datos y los métodos que pertenecen a la ya existente, para crear una herencia se debe de seguir el siguiente formato:*

***class*** *nombre\_clase* ***extends*** *nombre\_clase\_base{*

*//atributos de la clase heredada*

*//constructor(opcional)*

*//metodos especificos de la clase*

*}*

*Los atributos con el modificador de acceso* ***private*** *no se pueden acceder a ellos desde métodos de las clases derivadas, sólo se pueden a aquellos que tengan el modificador* ***public, protected, o friendly,*** *la clase base a su vez no puede acceder a los metodos especificos de la clase hijas, ya que para esta no es visible.*

*En la herencia existe la sobrecarga de métodos y se produce cuando se define uno con el mismo nombre que otro de la misma clase pero con distinto número o tipo de argumentos; en la sobrecarga no interviene el tipo de retorno.*

*Una clase derivada puede redefinir un método de la clase sin tener exactamente la misma cabecera o signatura, teniendo el mismo nombre pero distinta lista de argumentos; esta redefinición no oculta el método de la clase base, sino que da lugar a una sobrecarga del método heredado en la derivada.*

*Para determinar qué métodos ejecutar en cada llamada, el compilador realiza el proceso denominado* ***resolución de la sobrecarga,*** *es decir, enumera todos los métodos que se llamen igual de la clase heredada, ya sean propios o heredados; a continuación determina el tipo de los parámetros proporcionados en la llamada al método, si hay una coincidencia exacta en número y tipo, ese es el método a ejecutar; en caso contrario se pone en marcha el proceso de conversiones de tipos para determinar la mejor coincidencia.*

*Java considera que la herencia siempre es pública; esto significa que una derivada tiene acceso a los elementos públicos y protegidos de su base. Los elementos con visibilidad amigable(****friendly)*** *son accesibles desde cualquier clase del mismo paquete pero no son visibles en derivadas de otros paquetes. Una clase derivada no puede acceder a variables y métodos privados de su base; está utiliza elementos protegidos para ocultar sus detalles respecto a clases no derivadas de otros paquetes.*

***Constructores en herencia***

*Al construir un objeto de una clase derivada, primero se construye la parte de su base llamando a su constructor y, a continuación, se inicializan los miembros propios de la derivada, cuando el constructor de la base tiene argumentos, la primera sentencia del constructor de la derivada debe incluir una llamada al constructor de la base, esta se hace atreves de la palabra reservada* ***super();*** *los argumentos a transmitir se incluyen en la lista de argumentos de la base.*

***nombre\_clase(tipo argumento, tipo argumento,){***

***super(tipo argumento);***

***atributo\_clase\_hija = argumento;***

***}***

***Referencia a la clase base o padre***

*Se puede llamar a los métodos heredados de la clase base desde cualquier método de la clase derivada, simplemente se escribe su nombre y la lista de argumentos; puede ocurrir que haya métodos de la base que no interesa heredar en la derivada, debido a que se busca una funcionalidad adicional: para ello se sobreescribe el método en la derivada. Los métodos en las clases derivadas con la misma signatura, es decir, igual nombre, tipo de retorno y número y tipo de argumentos que los de la base, anulan o reemplazan al método de la base. El método ocultado de la base puede llamarse desde cualquier método de la derivada mediante la referencia* ***super*** *seguida de un punto, el nombre del método y la lista de argumentos, la palabra reservada* ***super*** *permite acceder a cualquier miembro de la clase base siempre que no sea privado.*

***super****.nombre\_metodo(argumentos);*

*Para evitar la herencia de clases se utiliza la palabra reservada* ***final***

*modificador* ***final*** *class nombre\_clase extends clase\_base{*

*…*

*}*

***Paquetes (Package)***

*Los paquetes son una forma de organizar grupos de clases. Un paquete contiene un conjunto de clases relacionadas bien por finalidad, por ámbito, o por herencia.*

*Los paquetes son el mecanismo que usa java para facilitar el modularidad del código. Un paquete puede contener una o más definiciones de interfaces y clases, distribuyendose habitualmente como un archivo. Para utilizar los elementos de un paquete es necesario importar este en el módulo de codigo en curso, usando para ello la sentencia* ***import.***

*Un paquete Java se genera sencillamente incluyendo la palabra* ***package*** *al inicio de los módulos de código en los que se definen las clases que formarán parte del mismo. Un paquete puede contener, además de definiciones de tipos como las clases e interfaces, otros paquetes, dando lugar a estructuras jerárquicas de contenedores.*

***Sintaxis***

***package*** *nombre\_paquete;*

*//importar paquetes propios o de la API de Java*

*class nombre\_clase{*

*//atributos*

*//métodos*

*}*

*.*

*.*

*.*

***Importación de paquetes***

*Para poder utilizar los paquetes en Java estos deben de ser importados hasta el bloque de código donde se va a utilizar, para este fin utilizaremos la cláusula* ***import,***  *a fin de importar en el ámbito actual las definiciones de otro paquete y poder usarlas, la cláusula* ***import*** *puede utilizarse para importar un elemento concreto de un paquete, facilitando el nombre de este seguido de un punto y el identificador de dicho elemento, un ejemplo es al importar la clase Math del paquete java.lang.*

***Ejemplo:***

*import java.lang.Math;*

*Es habitual que al importar un paquete nos interesen muchas de las clases definidas en el mismo. En este caso podríamos recurrir a utilizar un asterisco el cual indica que se van a importar todas las clases con sus métodos para poder utilizarse.*

*Ejemplo*

*import.java.lang.\*;*

***Desarrollo:***

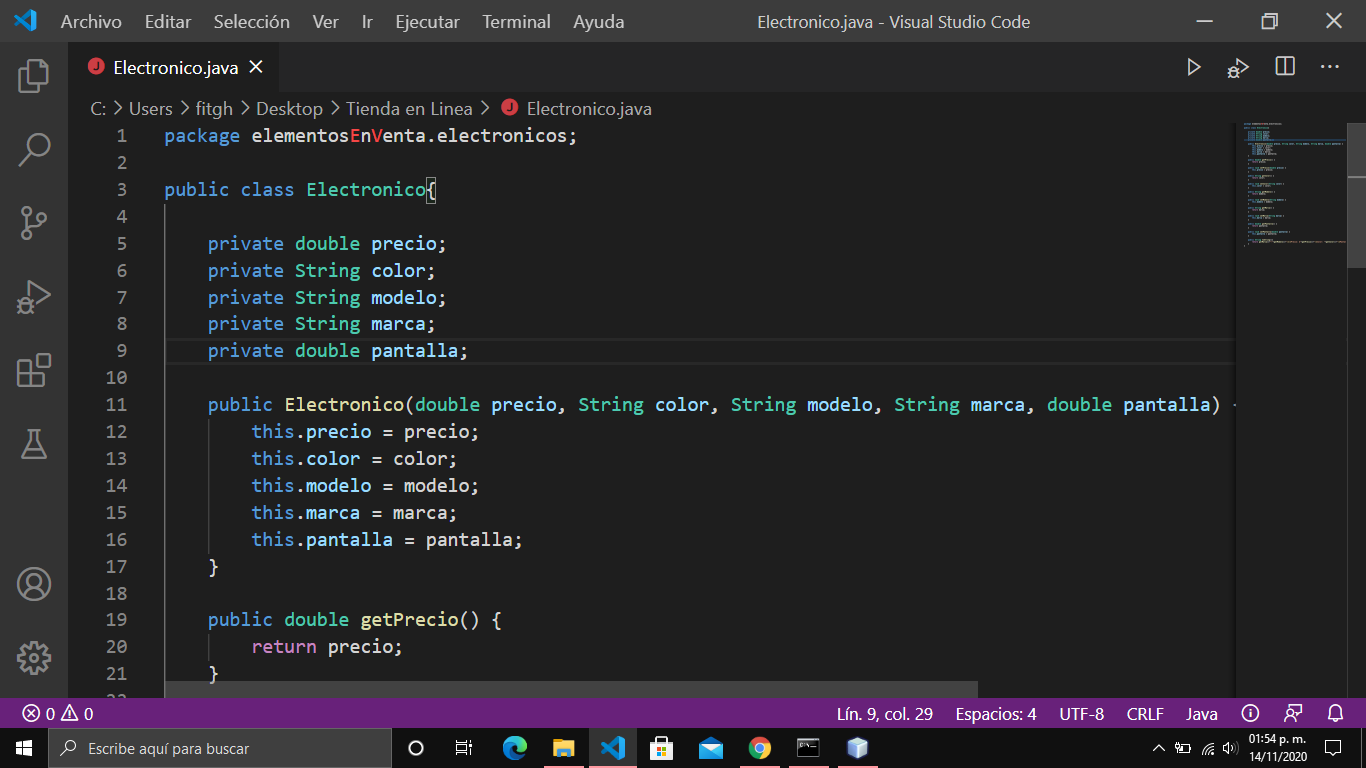
Análisis Previo:

Lo primero que se atendió fueron los requerimientos de la práctica, los cuales indican que deben tener una clasificación para cada tipo de elemento en venta, en este caso se establecen 3, electrónicos, alimentos, prendas de vestir, debe soportar dos tipos de usuarios administradores y usuarios normales y las formas de entrega de los productos, por lo cual se debe de establecer un paquete para usuarios, otro para elementos en venta y otro para las formas de entrega, en el paquete usuarios se deben de crear dos clases una para usuarios normales y otra para administradores, en el paquete de elementos en ventas se debe crear una clase por cada tipo de elementos en venta (electrónicos, alimentos, prendas de vestir) de las cuales heredarán a subclases de productos definidos, en el paquete formas de entrega se deben crear dos clases que ayuden a modelar las compras y las formas de entrega, por su parte en la clase domicilio, incluirá los datos de dirección y la tarjeta de pago del usuario y la clase para la entrega en tienda contendrá información sobre la fecha y la tienda a recoger el producto.

Además de lo anterior se implementará una clase catalogo encargada de inicializar y proporcionar a los productos disponibles a comprar en la tienda, proporcionando sus características y disponibilidad y de igual manera de una clase principal que muestre las opciones en pantalla que tienen los administradores y los usuarios.

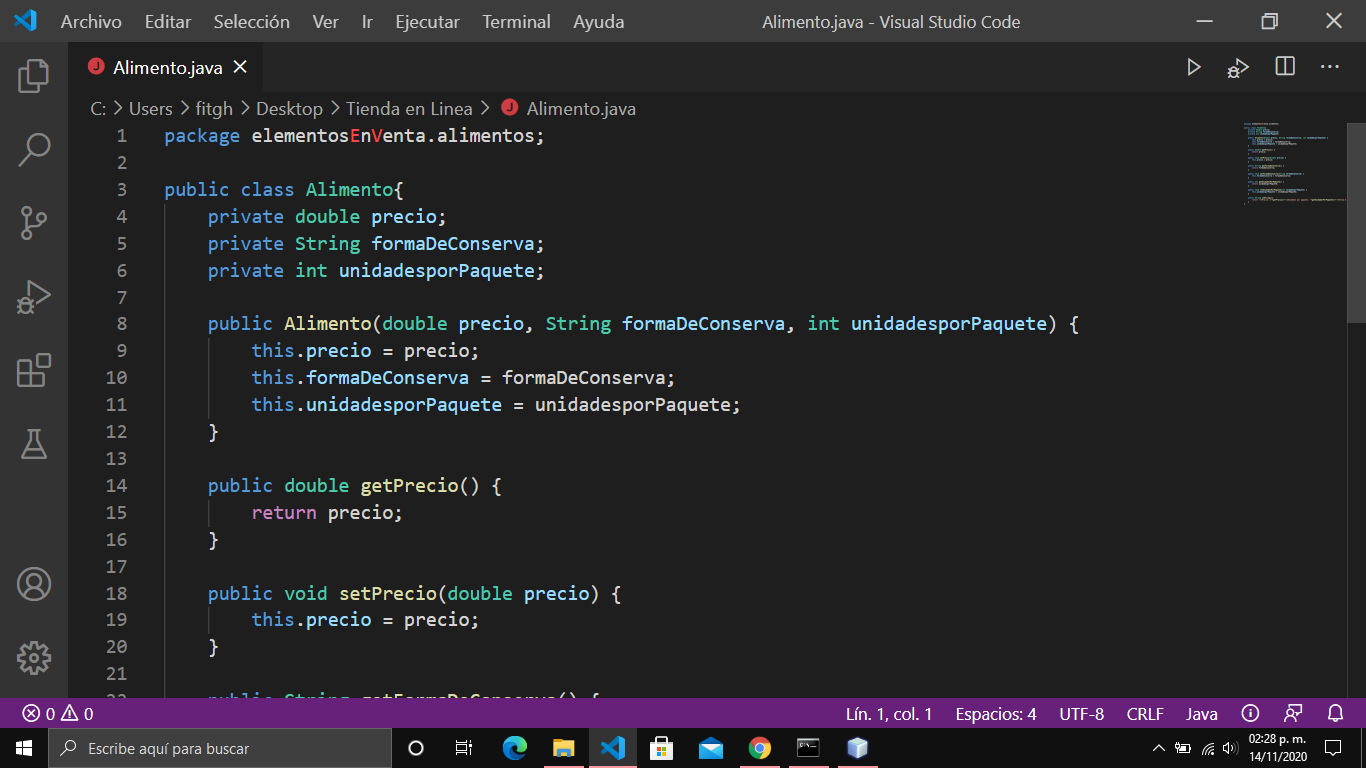
Desarrollo:

Primero se crearon las clases correspondientes a elementos en venta electrónicos, alimentos, prendas de vestir, por su parte en la clase Electrónico es un objeto que posee(atributos) un precio, color, modelo, una marca y dispone de un tamaño de pantalla.



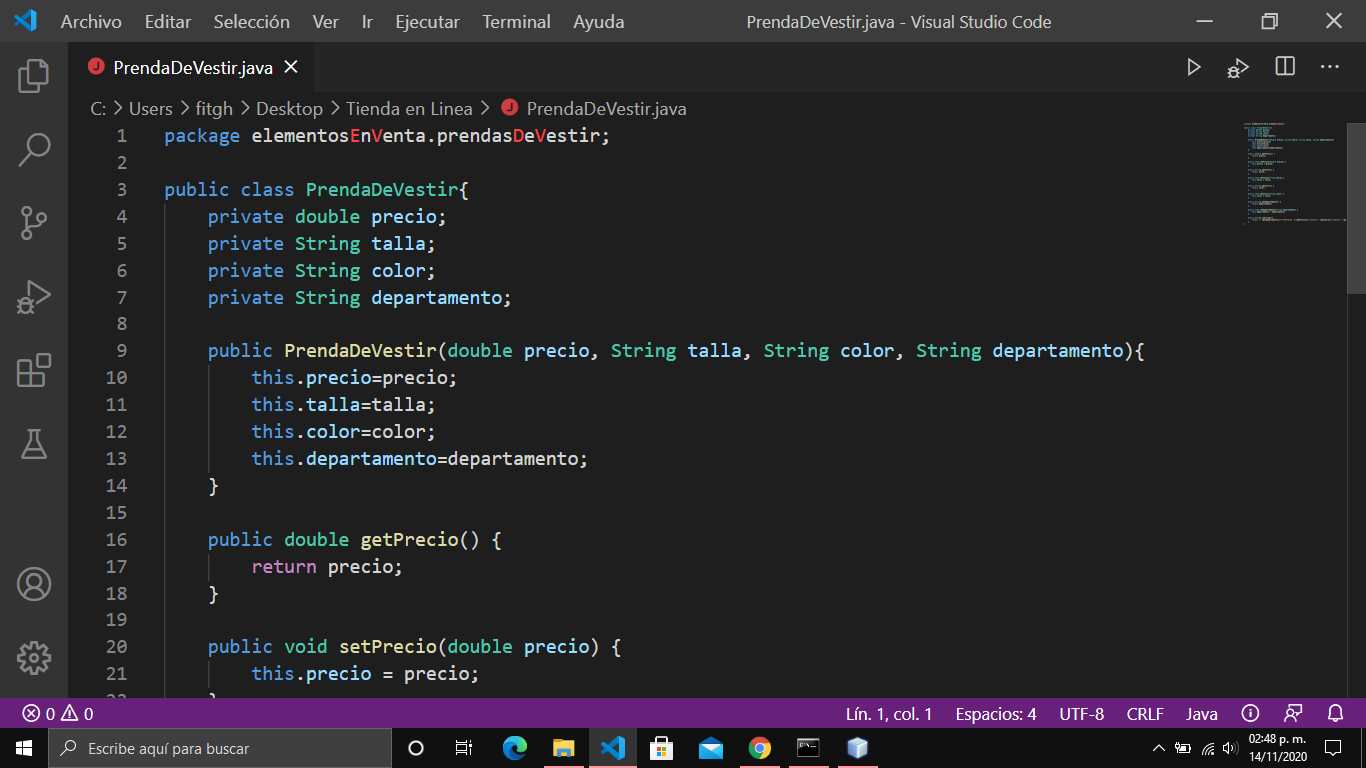
De dicha clase se establece por relación de herencia tres clases, que son Celular, Computadora y TV, y por ende las tres comparten los atributos de la clase Electrónico, por un lado celular posee de atributos la memoria, Cámara, batería y un sistema operativo, la clase computadora contiene un tipo, que indica si el dispositivo es portátil o de escritorio, memoria, almacenamiento, un procesador y el sistema operativo y la clase TV un tipo (Smart por ej.), resolución y su tecnología de pantalla, cada clase con sus respectivos métodos consultores, para crear objetos de estos tipos.

Para los alimentos se establece una clase padre llamada alimento con los siguientes atributos:



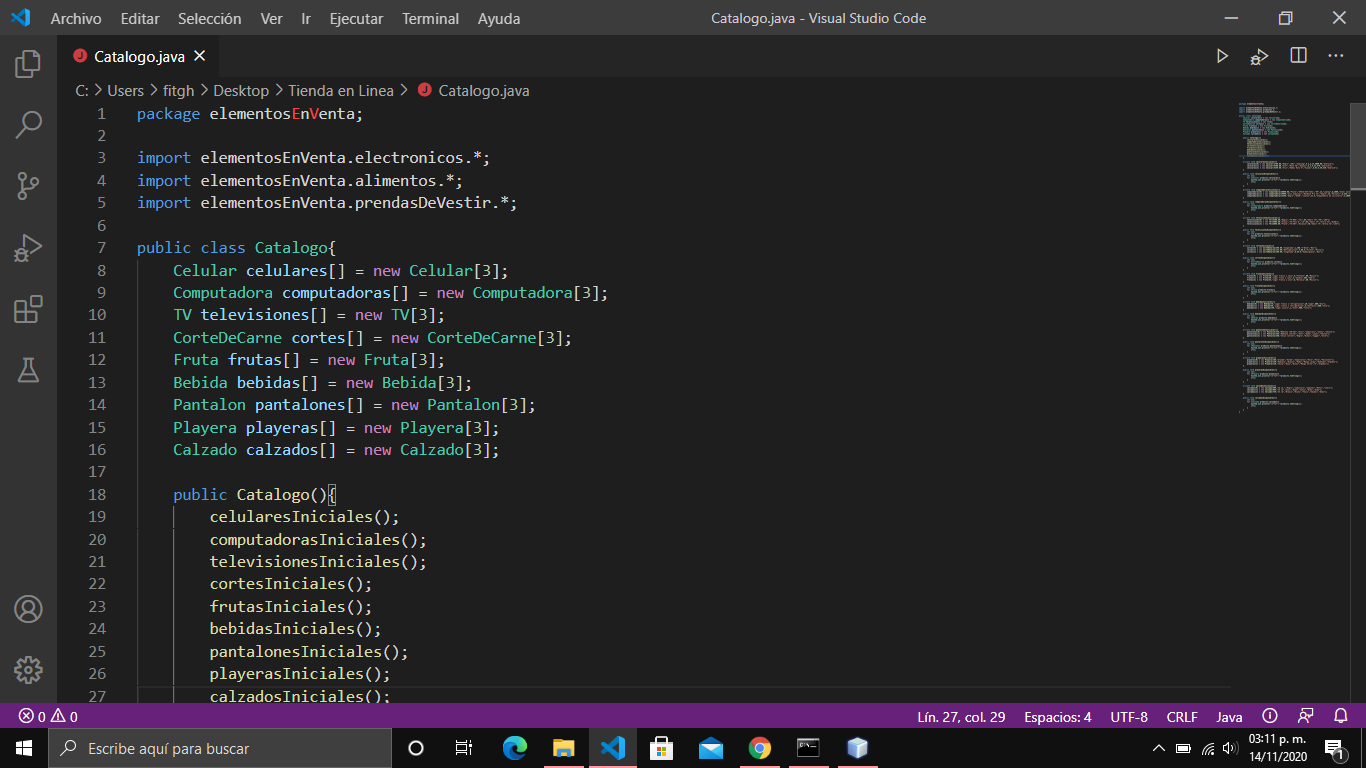
De la cual se crean tres subclases, la primera CorteDeCarne con atributos para el peso por porción, el tipo de corte, y el animal de origen, Fruta con atributos del peso por porción, tipo de fruta y país de procedencia y una tercera clase para la creación de Bebidas con atributos del sabor, volumen y tipo de bebidas, instanciando objetos con las características de los elementos mencionados.

Posteriormente se desarrolló una tercera clase padre correspondiente a prendas de vestir de la siguiente manera:

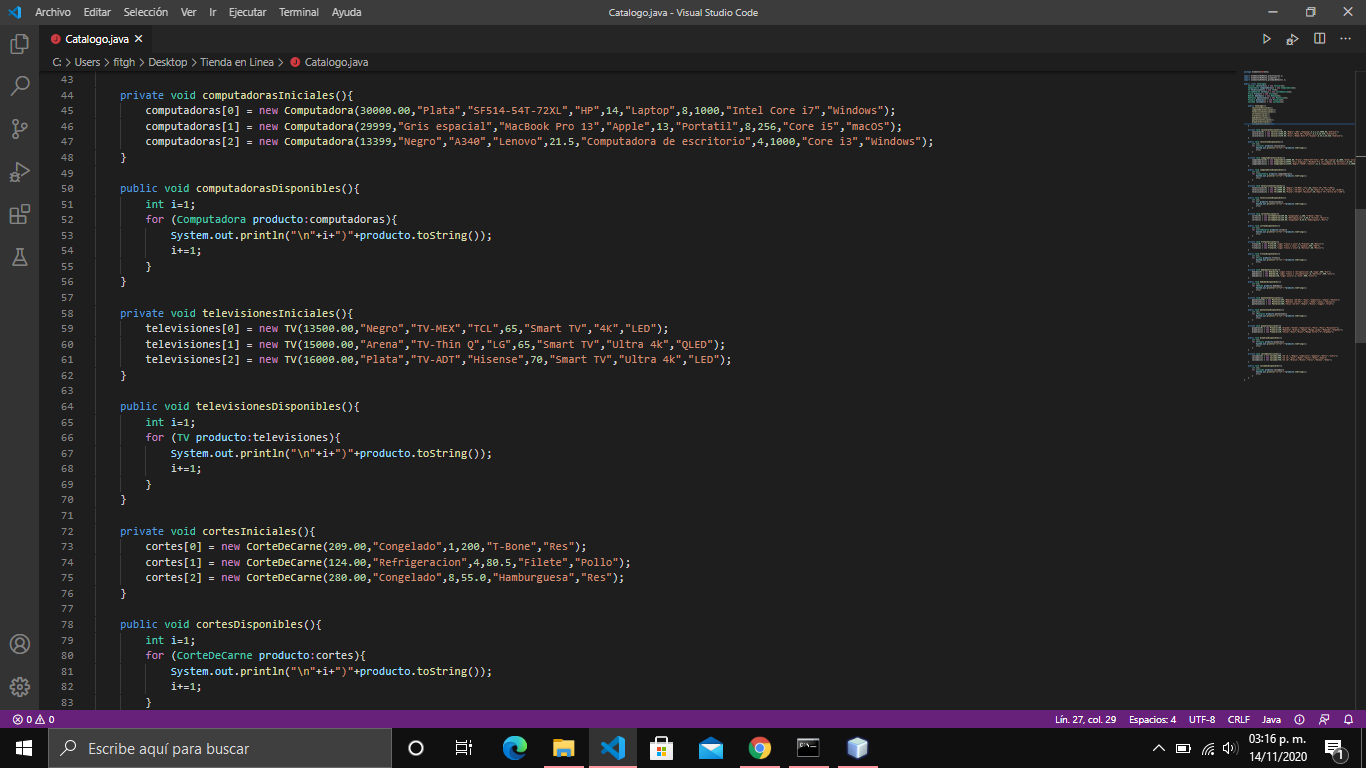


De la cual por relación de herencia se establecen las clases, Pantalon, Playera, y Calzado, la clase Pantalon es usada para prendas de este tipo y dispone de los atributos de tipo y corte que son aspectos a resaltar en este tipo de prendas, por otro lado la clase Playera cuenta con los atributos de tipo, cuello y material y la clase Calzado con tipo, marca y suela, atributos que ayudan a modelar a las prendas correspondientes a cada subclase.

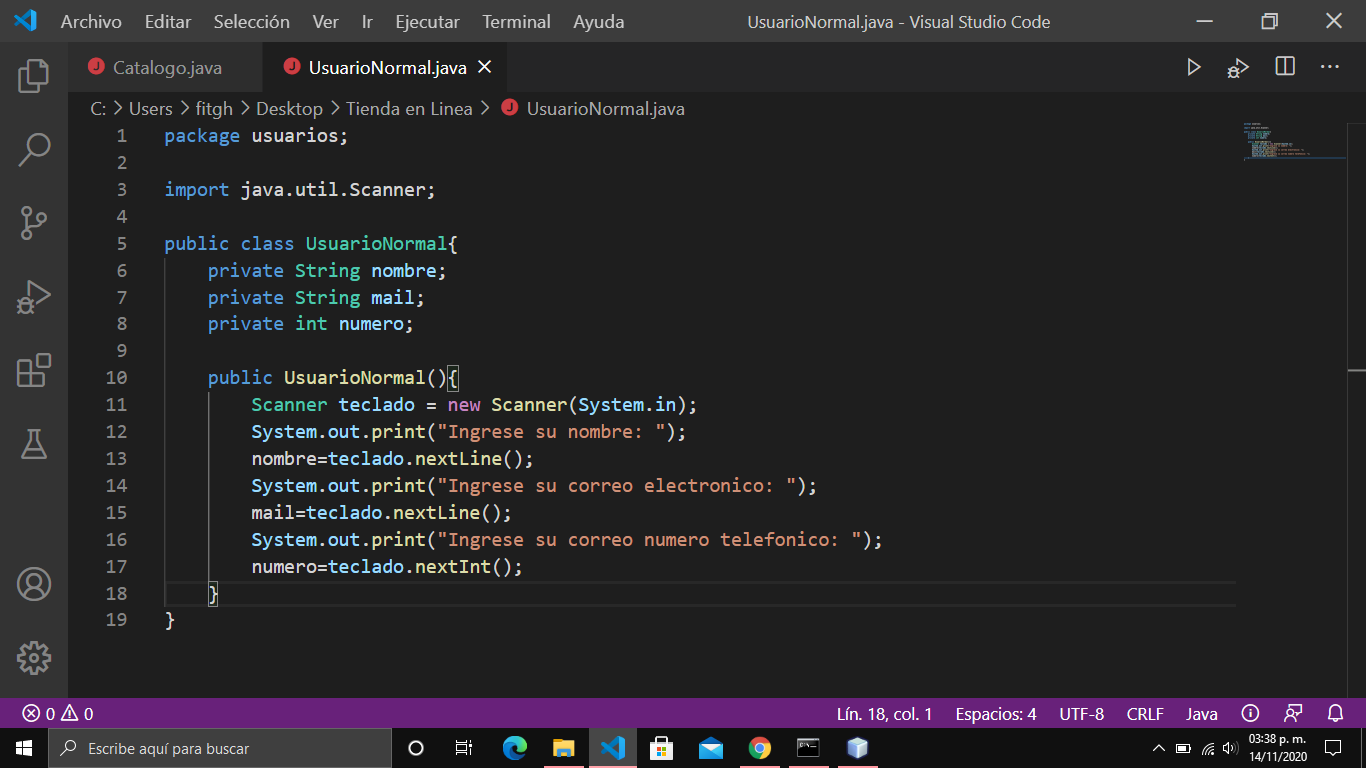
Para la creación de los objetos de cada producto disponible se desarrolló la clase Catalogo:



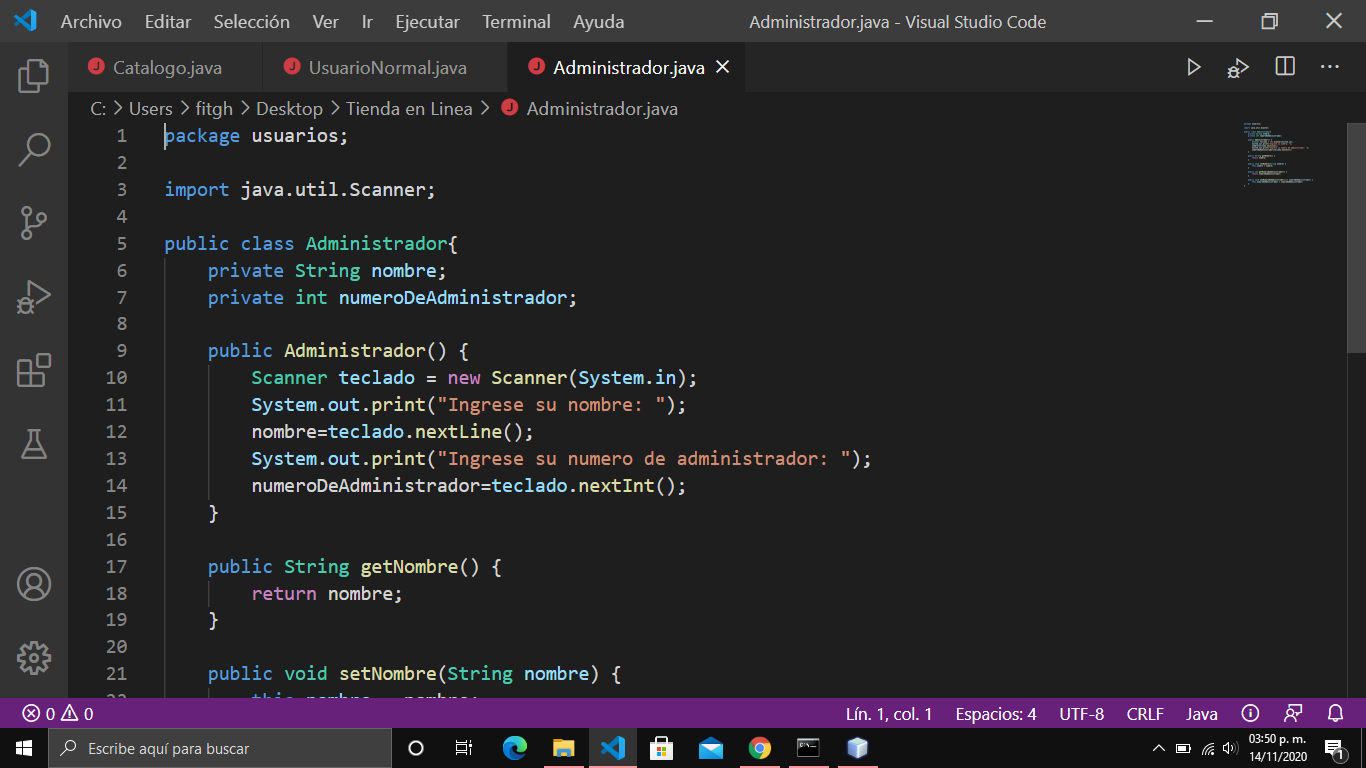
La cual tiene definida un arreglo de tres elementos para cada uno de los artículos en venta, y sus respectivos métodos para crear e inicializar cada artículo y otro método para poder mostrar en pantalla los artículos de cada tipo, por ejemplo para las computadoras, televisiones y cortes de carne se tiene:



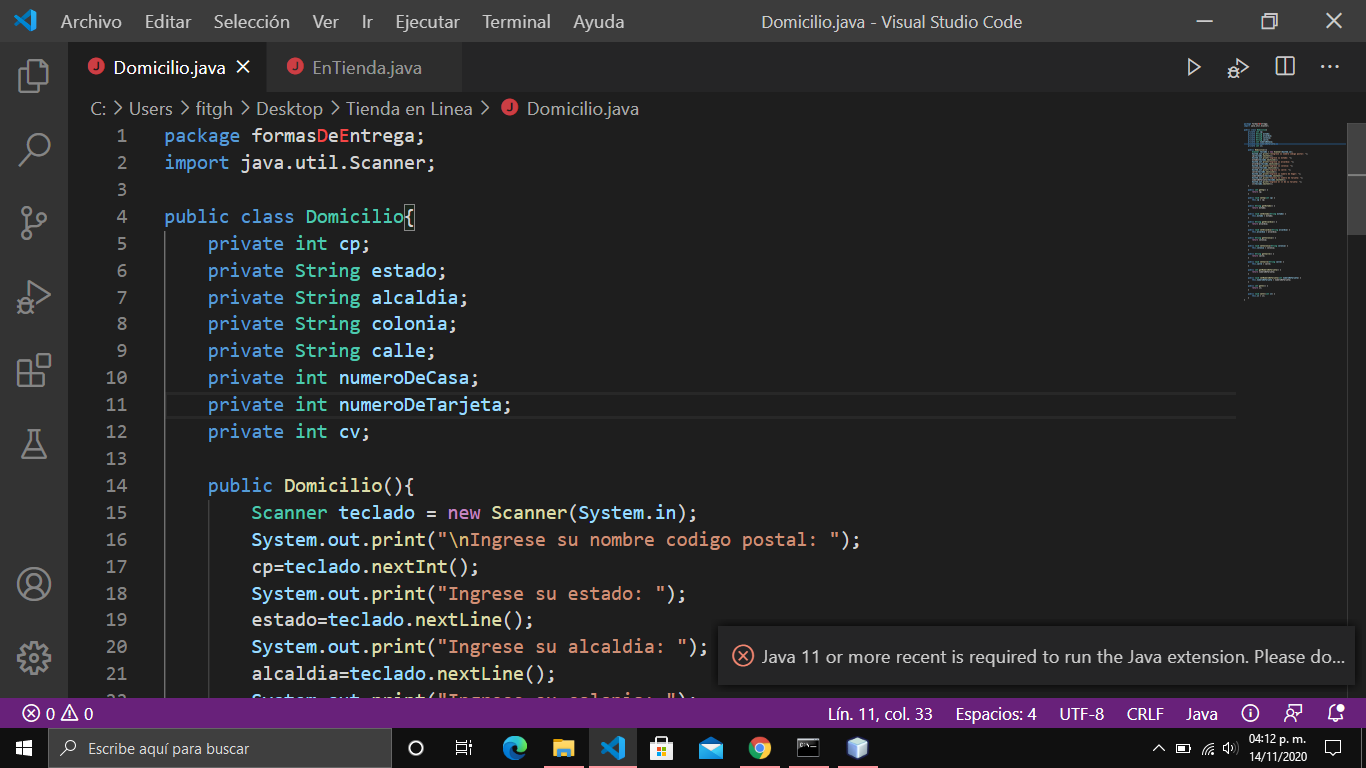
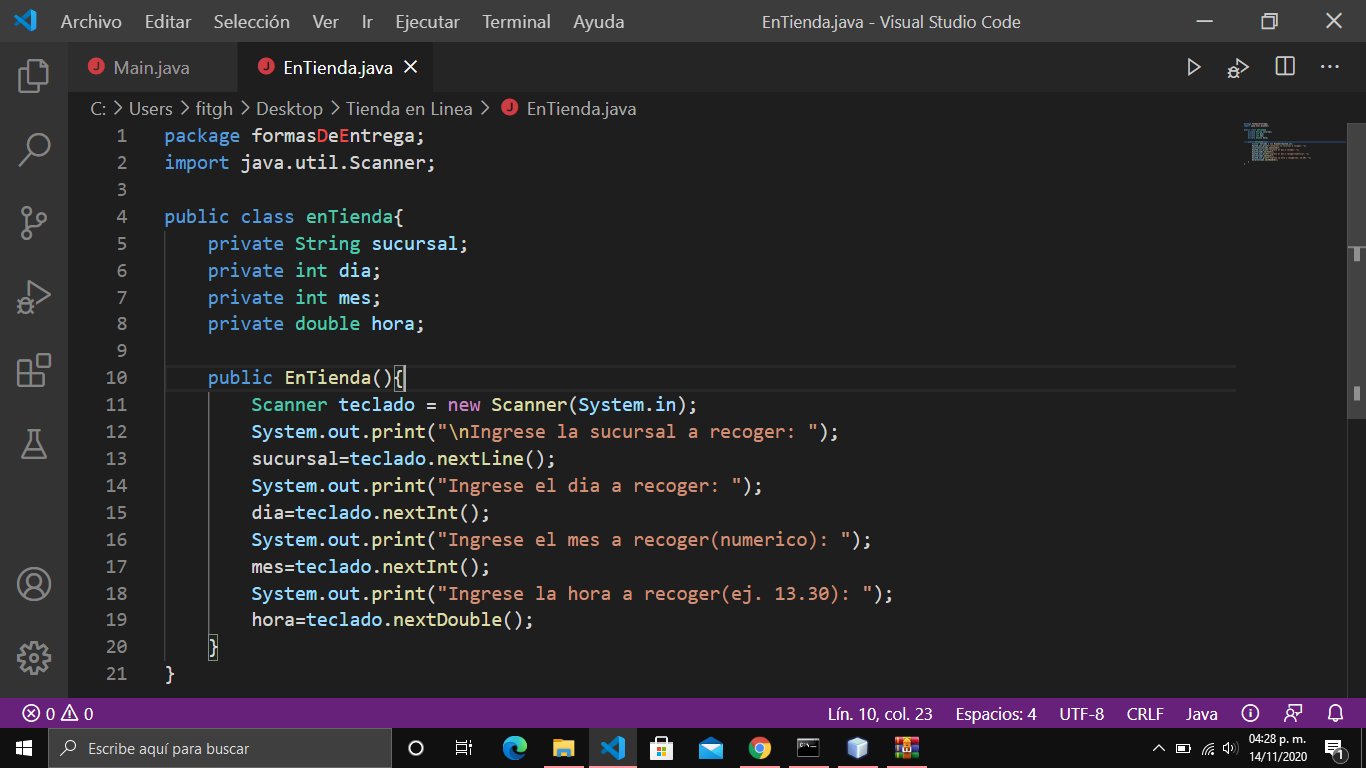
En el paquete correspondiente a los usuarios se desarrollaron dos clases una correspondiente a los Usuarios normales y otra que se encarga de los Administradores de la tienda, para el caso de los usuarios normales se tiene:



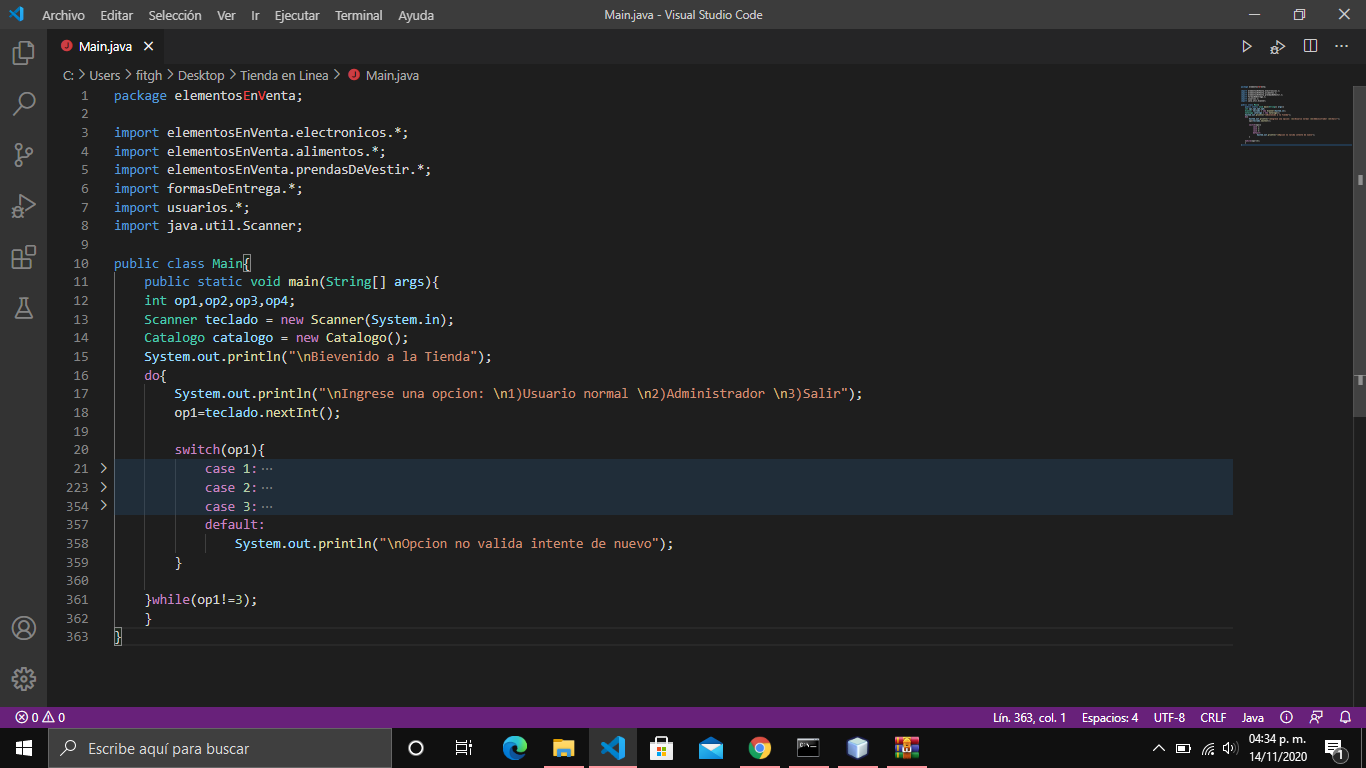
Que modela a un usuario con su nombre, mail y número de celular, en el caso de los administradores la clase tiene los siguientes atributos:



Para la sección de las formas de entrega se diseñó una clase para la entrega a domicilio y otra clase para la entrega en tienda, dado que ya se tiene los datos de nombre y contacto del usuario por parte de la entrega a domicilio se requieren los datos correspondientes al domicilio del usuario, tales como código postal, estado, alcaldía, colonia, calle, número de hogar, número de tarjeta y el código de verificación, y por el lado de entrega en tienda, se requiere de la sucursal a recoger, la fecha y la hora.

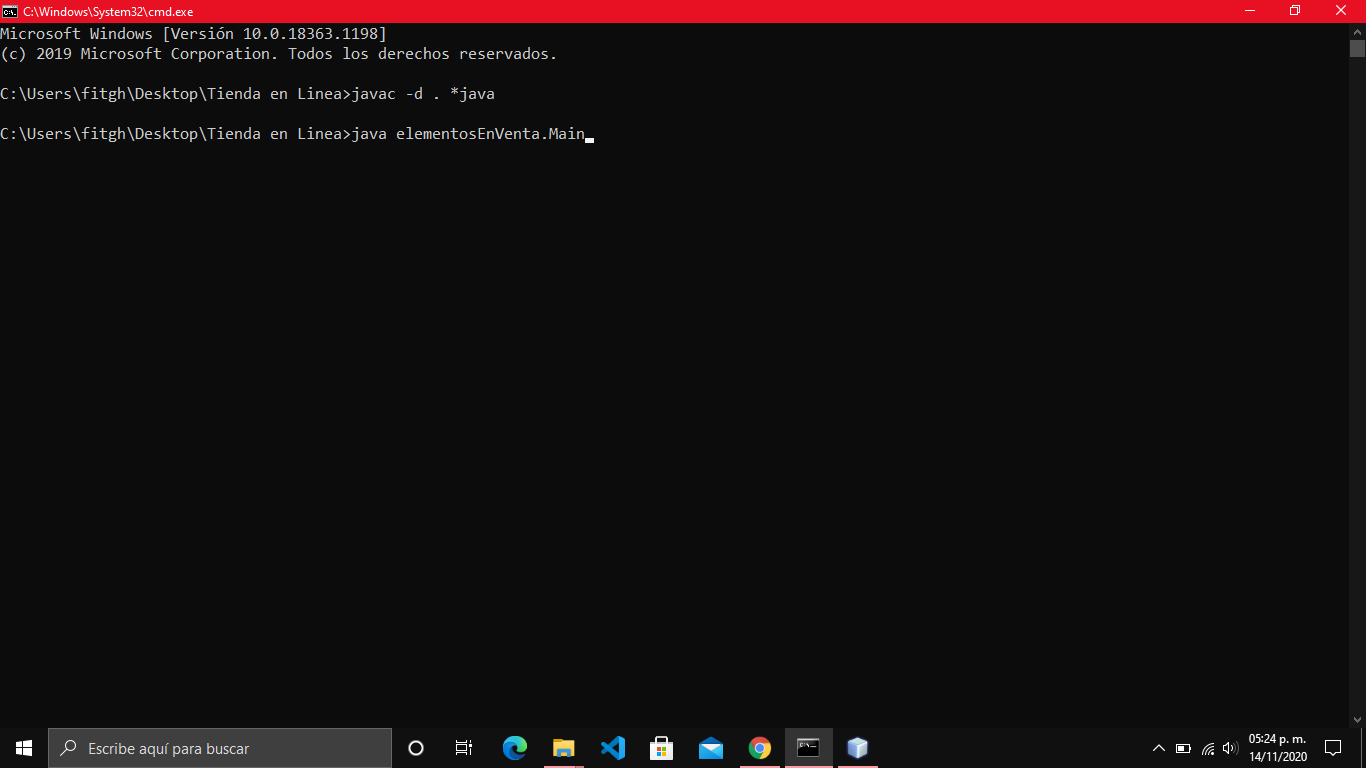
La clase desarrollada Main recopila a la gran mayoría de las clases descritas anteriormente, en esta clase se instancian objetos de las clases, Catalogo, UsarioNormal, Administrador, Domicilio, enTienda, entre otros y en ella se despliega un menú con las opciones para los usuarios normales y los administradores.



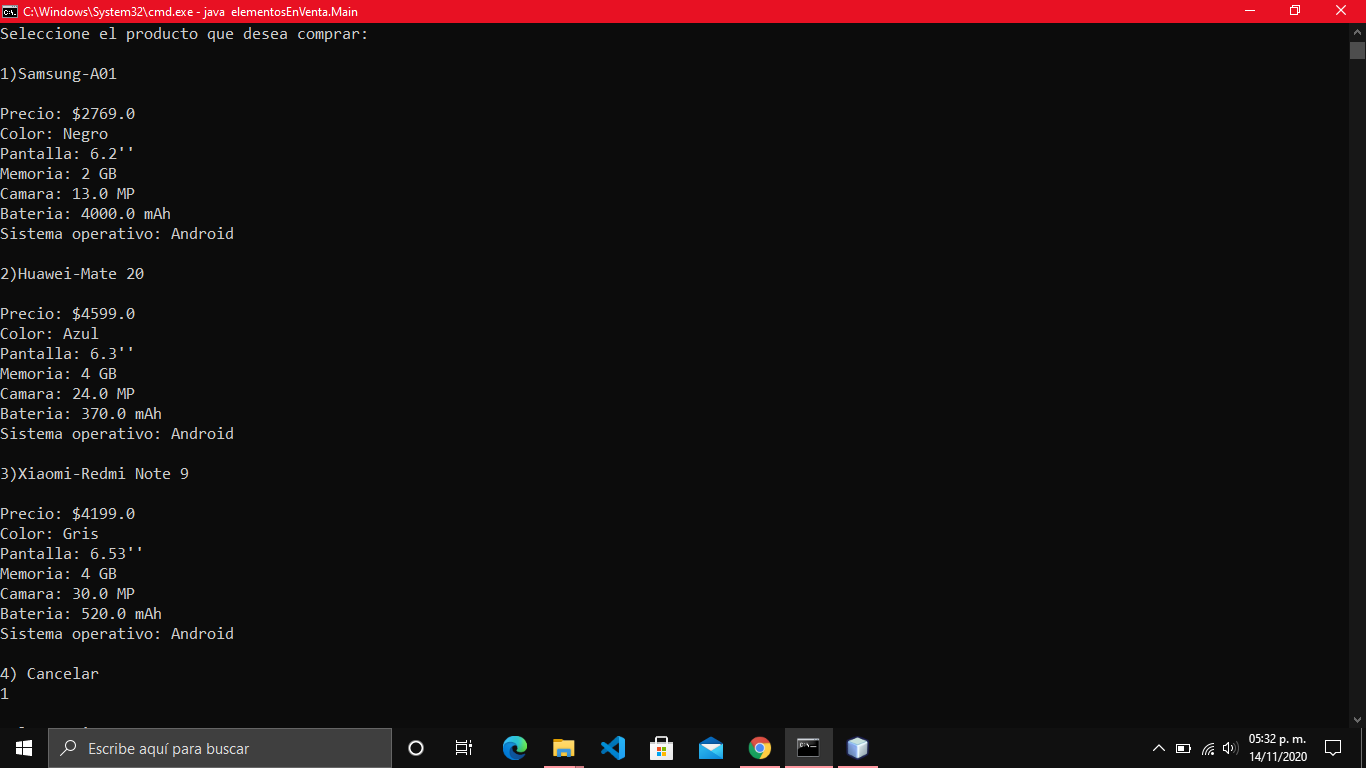
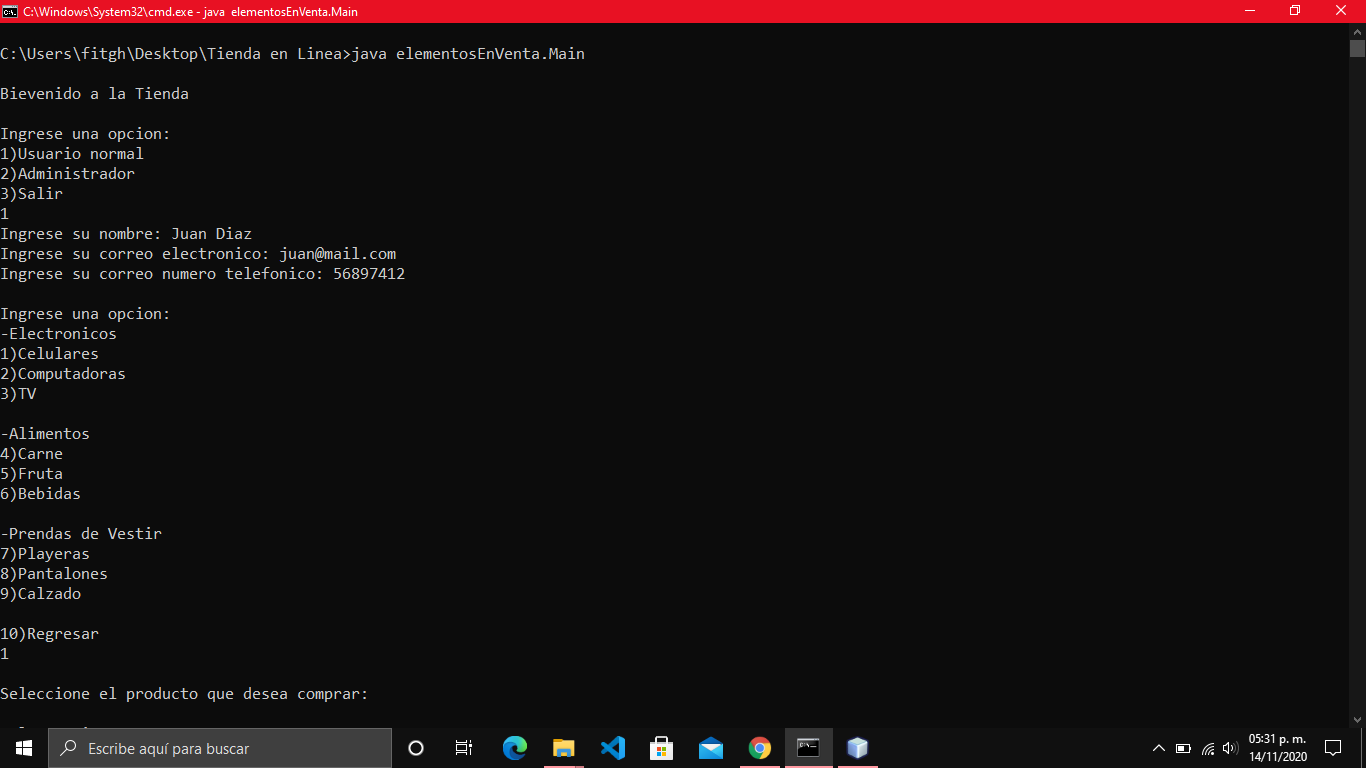
**Clase principal.**

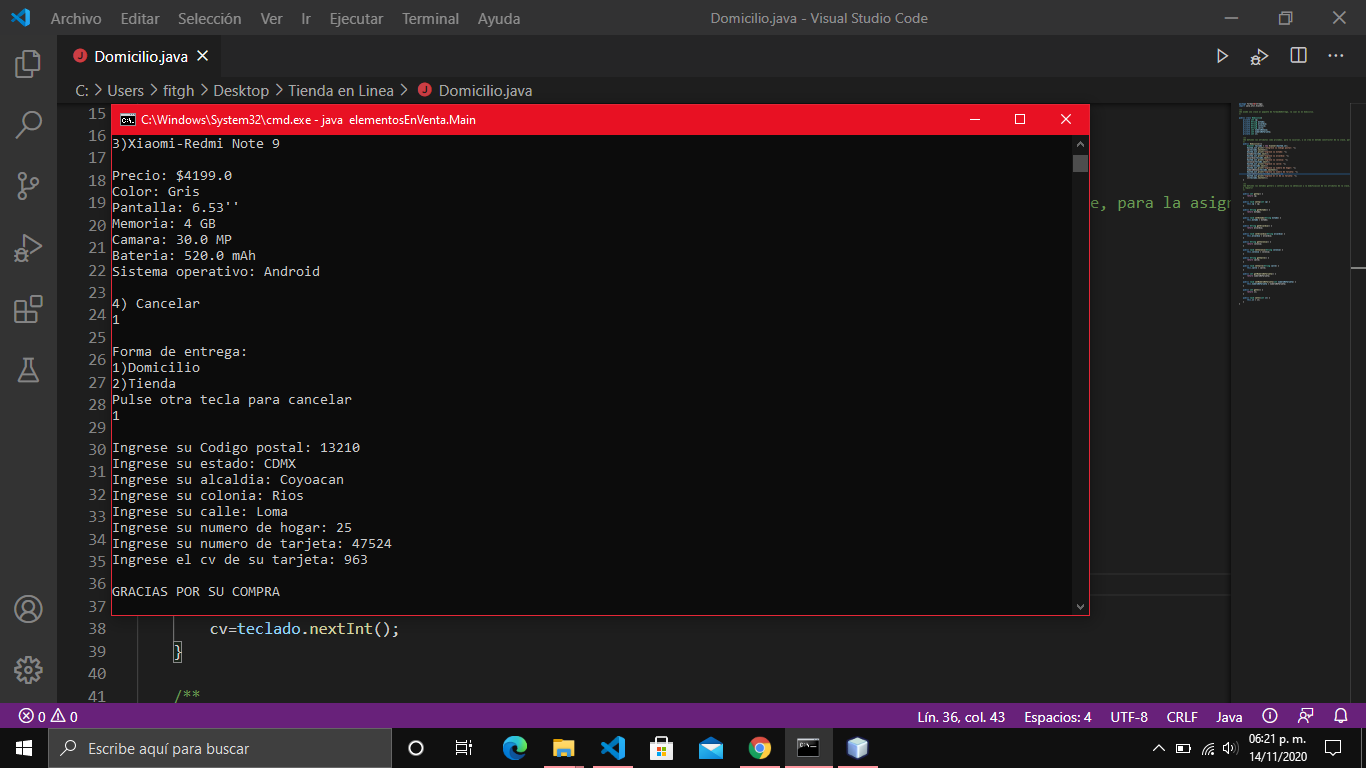
**Ejecución de la aplicación:**

Para ejecutar la aplicación:

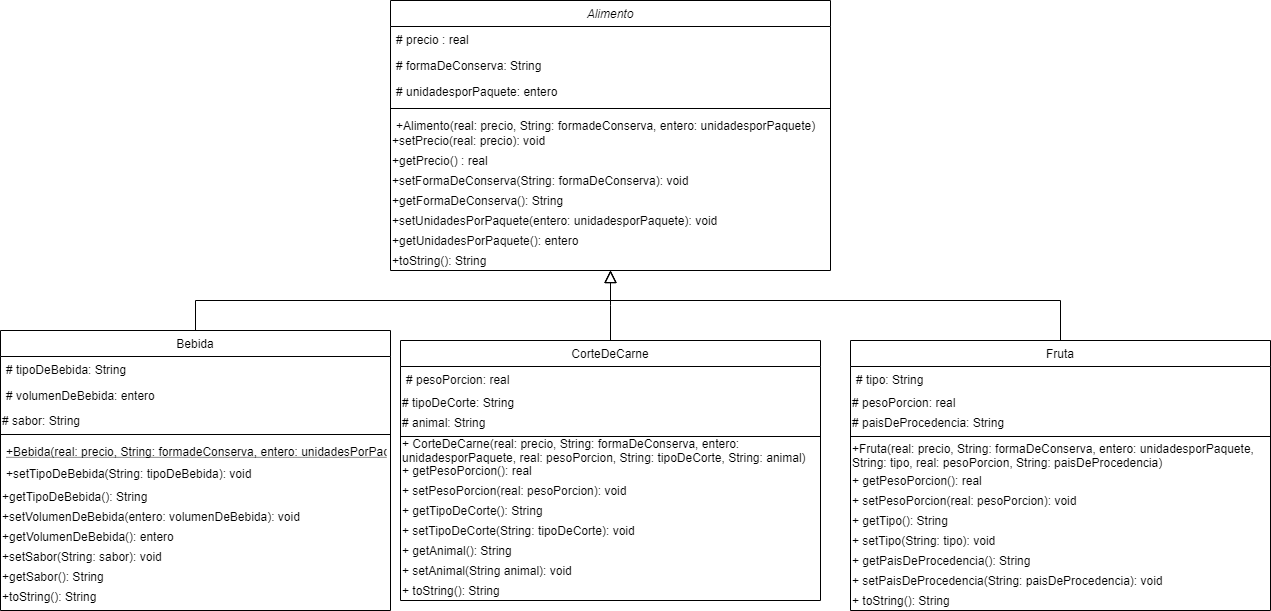
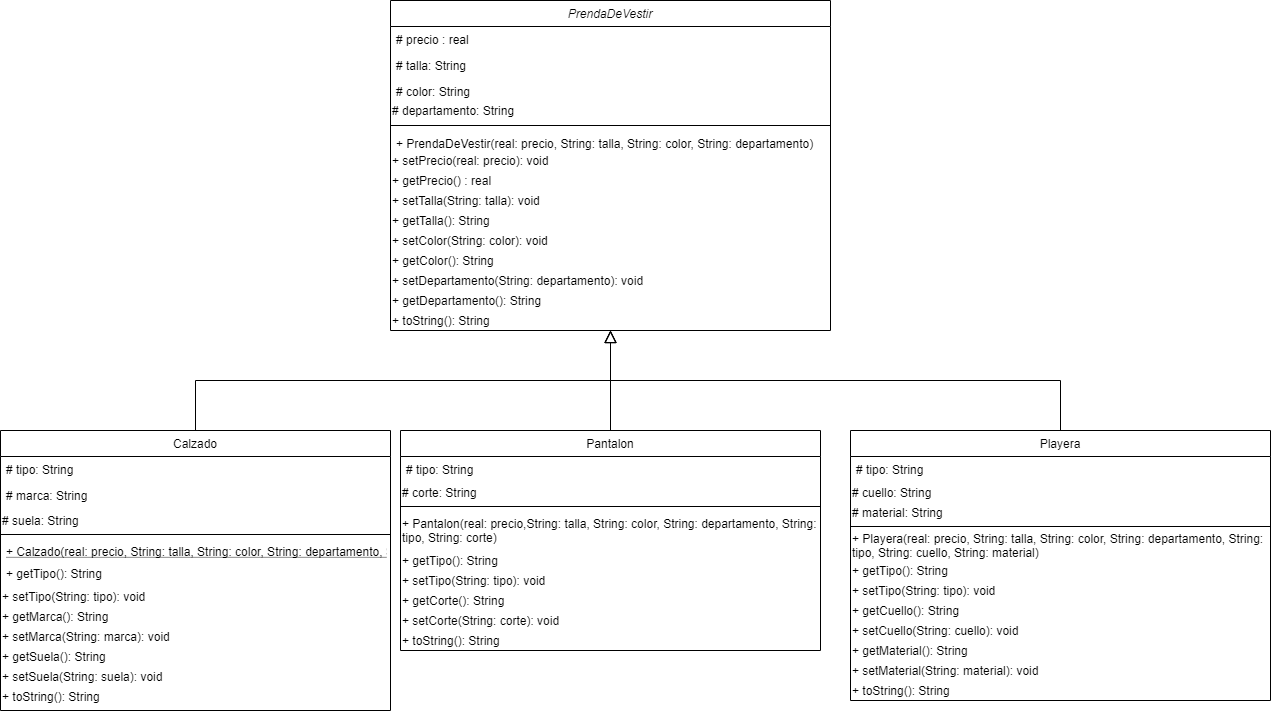


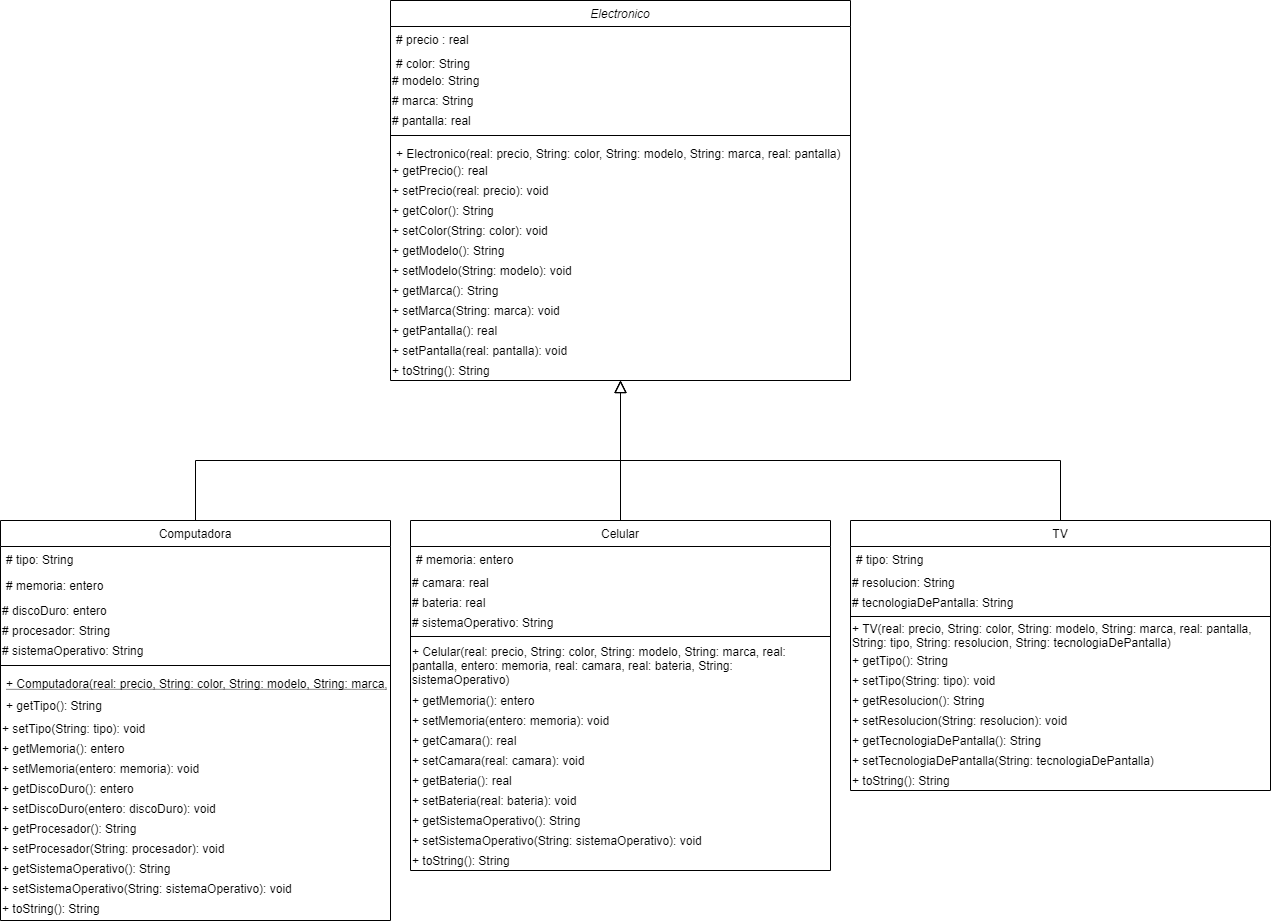
Capturas de ejecución:





***Diagramas de las clases desarrolladas:***

* **Clase Alimento:**
* **Clase PrendaDeVestir**
* **Clase Electrónicos:**



**Conclusiones:**

* Dávila Ortega Jesús Eduardo:

Durante la práctica entendí el concepto de encapsulamiento, esto gracias a la retroalimentación del equipo y la explicación de cómo es que funciona el programa, este programa en lo personal me fue algo difícil al momento de entender cómo iba a ser el funcionamiento y las relaciones que iban a tener las clases para poder interactuar ya sea de una manera casi completa a una manera reducida pero gracias a la interacción del equipo logre entender cómo relacionar las cosas y cómo funcionaba lo que planteamos realizar

* Díaz Hernández Marcos Bryan:

En esta práctica aprendí a como poder encapsular de forma más concreta a las clases que utilizamos, para poder llevar a cabo una tienda, donde los productos estuviesen agrupados dentro de un paquete, que en cierto punto me confundió demasiado pero la práctica me llevó a entender mejor el funcionamiento de estos y como permiten el ordenar las clases, cumpliendo con el objetivo de la práctica.

* Pareja Ávila Emiliano:

Aunque al principio del tema se me complicó un poco entender el tema de paquetes, conforme las clases fueron pasando este concepto lo entendí mucho mejor, y sé cómo organizar y realizar jerarquía de clases en paquetes, así como compilar y ejecutar desde terminal todas las clases y archivos contenidos dentro de uno. También tengo un buen manejo de la jerarquía de clases, y con esta práctica, gracias a que se modelaban varios objetos y se aplicaba la herencia, pude comprender mejor este concepto que es fundamental en la Programación Orientada a Objetos. Entonces, puedo decir que se cumplieron los objetivos de la práctica.

* Vázquez Zavala Oliver Alex:

Esta práctica me ha dejado de aprendizaje el uso y la importancia de los paquetes en la programación orientada a objetos y en particular en Java, con el uso de paquetes en la práctica me di cuenta que son una herramienta bastante útil en la organización de un programa en java, ya que ayuda a organizar y hacer un mejor manejo de las clases, ayudando a no trabajar todo en un mismo archivo o en una misma carpeta proporcionando una forma más organizada de trabajar, e incluso al hacer revisiones y modificaciones de un código o al trabajar en equipo dividir una aplicación en paquetes es una muy buena opción.

**Bibliografía:**

* Joyanes Aguilar, Luis, Zahonero Martínez, Ignacio(2011) “Programación en Java: Algoritmos y programación orientada a objetos” pág. (321 - 340).