INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

REDES DE COMPUTADORAS

PROF. AXEL ERNESTO MORENO CERVANTES

GRUPO: 3CM7

PRÁCTICA 2 – TIENDA VIRTUAL

ALEJANDRO DE JESÚS ZEPEDA FLORES

28 de abril de 2020

Prácticas: https://drive.google.com/drive/folders/12lUUJImgU72oXJfukVcbrB8OjSsMSkb2?usp=sharing

Práctica 2: https://drive.google.com/drive/folders/1awvIsKhA7fDnjwq1920C\_3w0pbvwtqzi?usp=sharing

OBJETIVO

Implementar un servicio de compra en línea que permita al usuario ver un catálogo de artículos, agregar a su carrito de compras varios artículos (modificar las cantidades desde el carrito de compras, guardar el estado del carrito si es que no se desea efectuar la compra en ese momento) y realizar la compra. Después de realizada la compra el usuario recibirá un ticket de compra y en el catálogo del servidor se actualizarán las existencias de artículos.

INTRODUCCIÓN

Ambas formas de comunicación (UDP y TCP) utilizan la abstracción de sockets, que proporciona los puntos extremos de la comunicación entre procesos. Los sockets (conectores) están presentes en la mayoría de las versiones de UNIX, incluido Linux y también Windows.

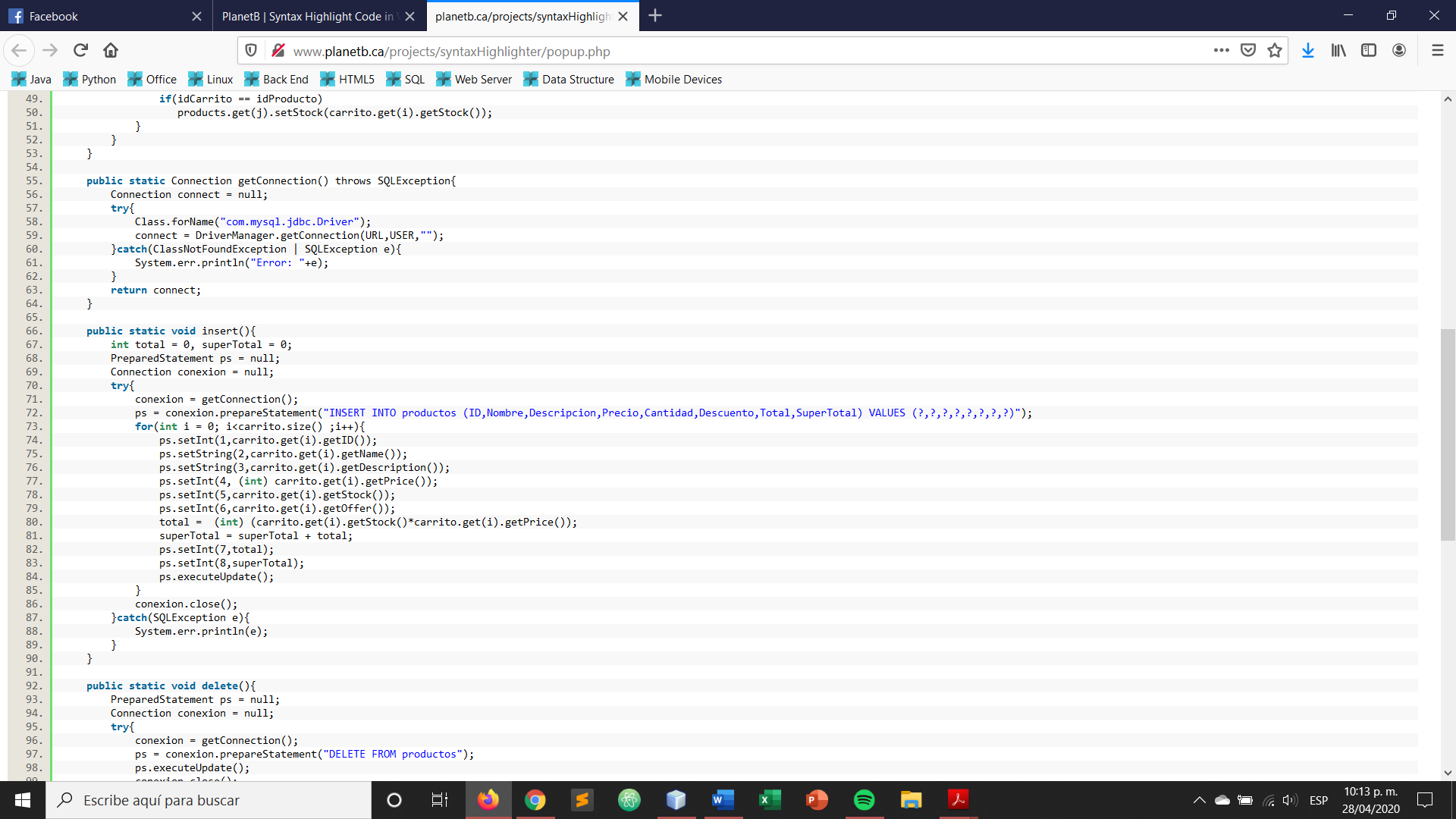
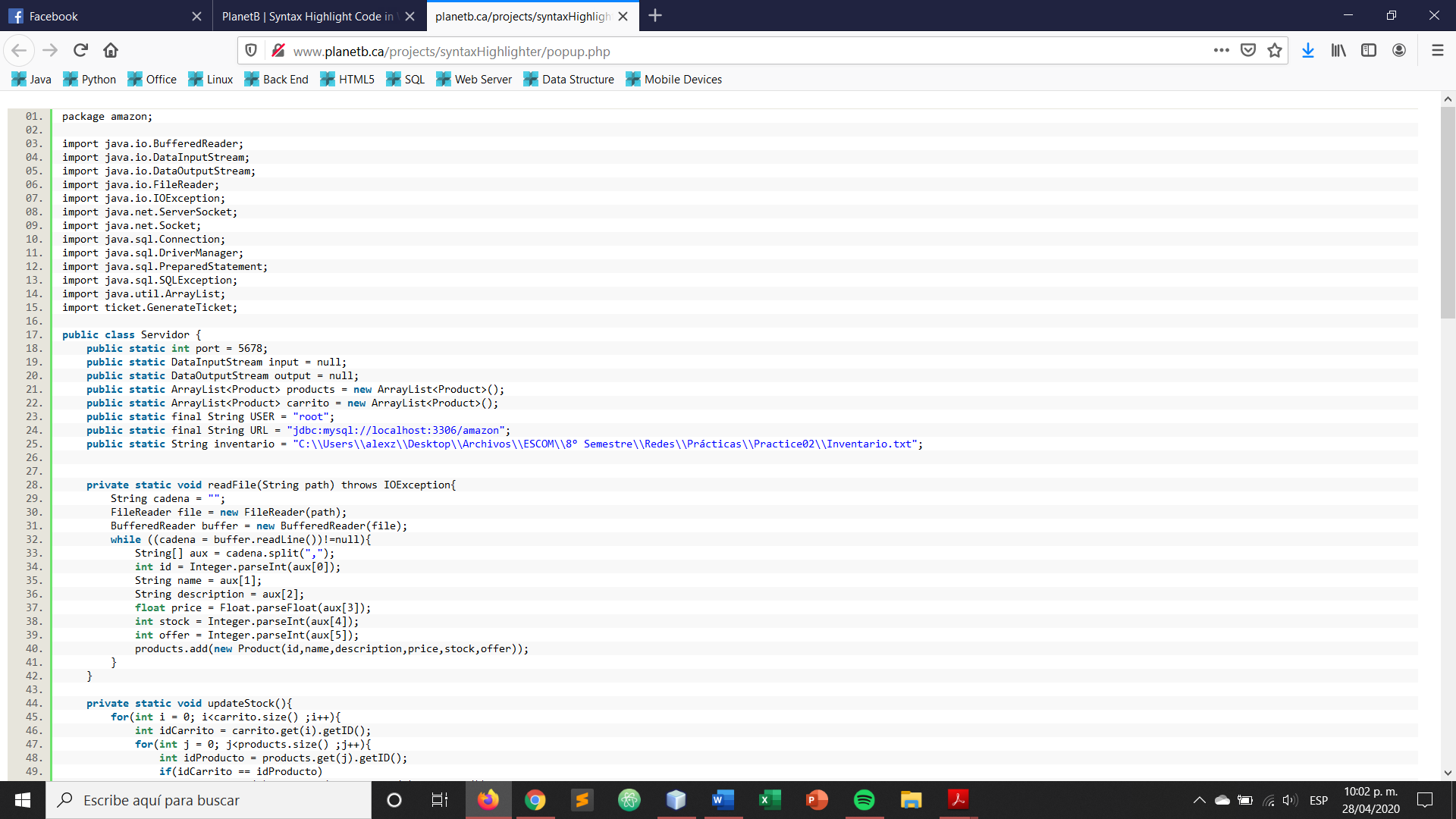
Los sockets permiten conectar dos programas en red para que se puedan intercambiar datos. Los sockets están basados en una arquitectura cliente/servidor. En esta arquitectura uno de los programas debe estar siempre arrancado y pendiente de que alguien establezca conexión con él. Este programa se denomina servidor. El otro programa lo arranca el usuario cuando lo necesita y es el programa que da el primer paso en el establecimiento de la comunicación. Este programa se llama cliente.

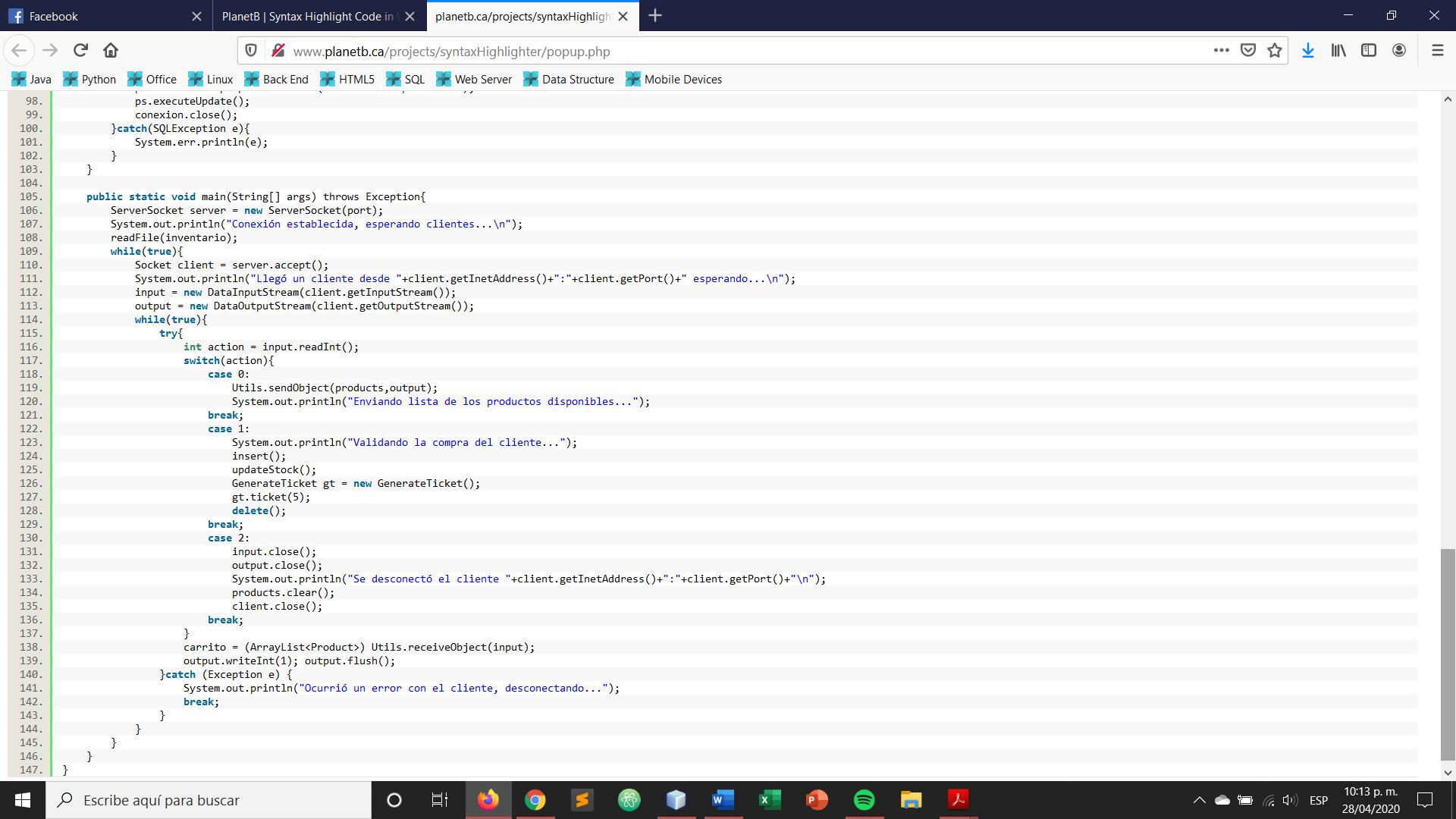
DESARROLLO

Iniciamos con la programación del Servidor, ya que este es el que estará enviando y recibiendo los datos con los clientes. Realmente las únicas funciones que se pueden observar no realizan un funcionamiento complicado.

* **readFile(String path):** esta función es la encargada de leer el archivo de texto donde el usuario especifico los productos disponibles y las características de cada uno. Para esta práctica, utilizamos un ArrayList products para almacenarlos y por la facilidad que representa enviarlo como Objeto.
* **updateStock():** esta función actualiza el inventario que se encuentra en el servidor para que no importa la cantidad de clientes que se conecten al servidor, no se tenga problemas con la cantidad de productos disponibles, ya que, de no actualizar el inventario, cada vez que se conecte un cliente, la cantidad de productos siempre sería la misma y nunca se terminarían.
* **getConnection():** la utilización de bases de datos no era necesario para está práctica, sin embargo, con la finalidad de brindar una mejor interacción entre el usuario y la aplicación, se implementó el uso de Bases de datos y lo único que realiza la función es la conexión con la base.

A continuación, se muestra el código implementado en el Servidor.





Ahora, vamos a explicar la parte del cliente.

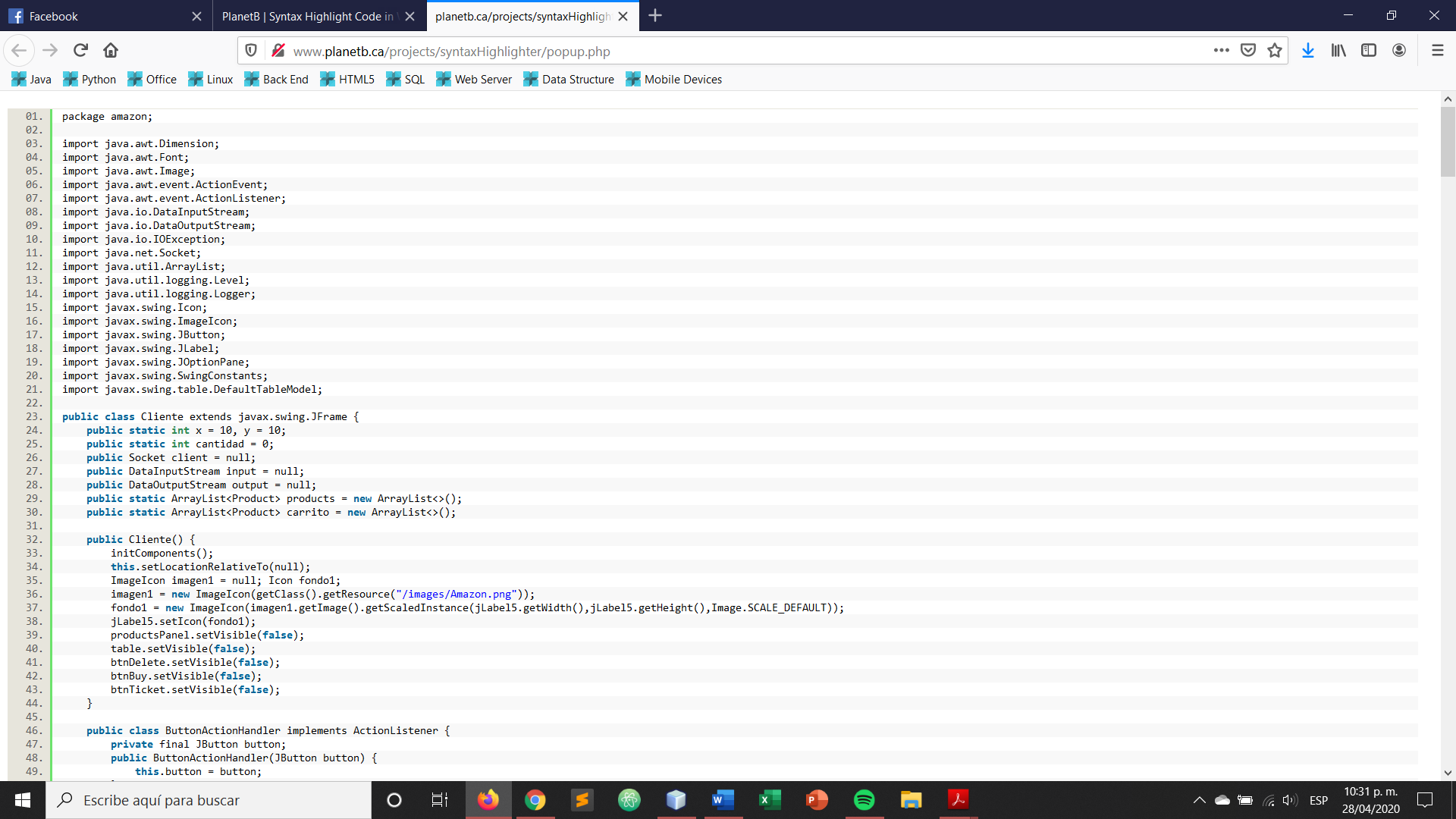
* **ButtonActionHandler implements ActionListener:** esta es una función dinámica, ya que, por cada producto, en la interfaz de usuario se crea un botón con la imagen de este, entonces, para su posterior manejo, se agrega un ActionListener para identificar el botón que fue presionado y recolectar los datos de dicho botón.

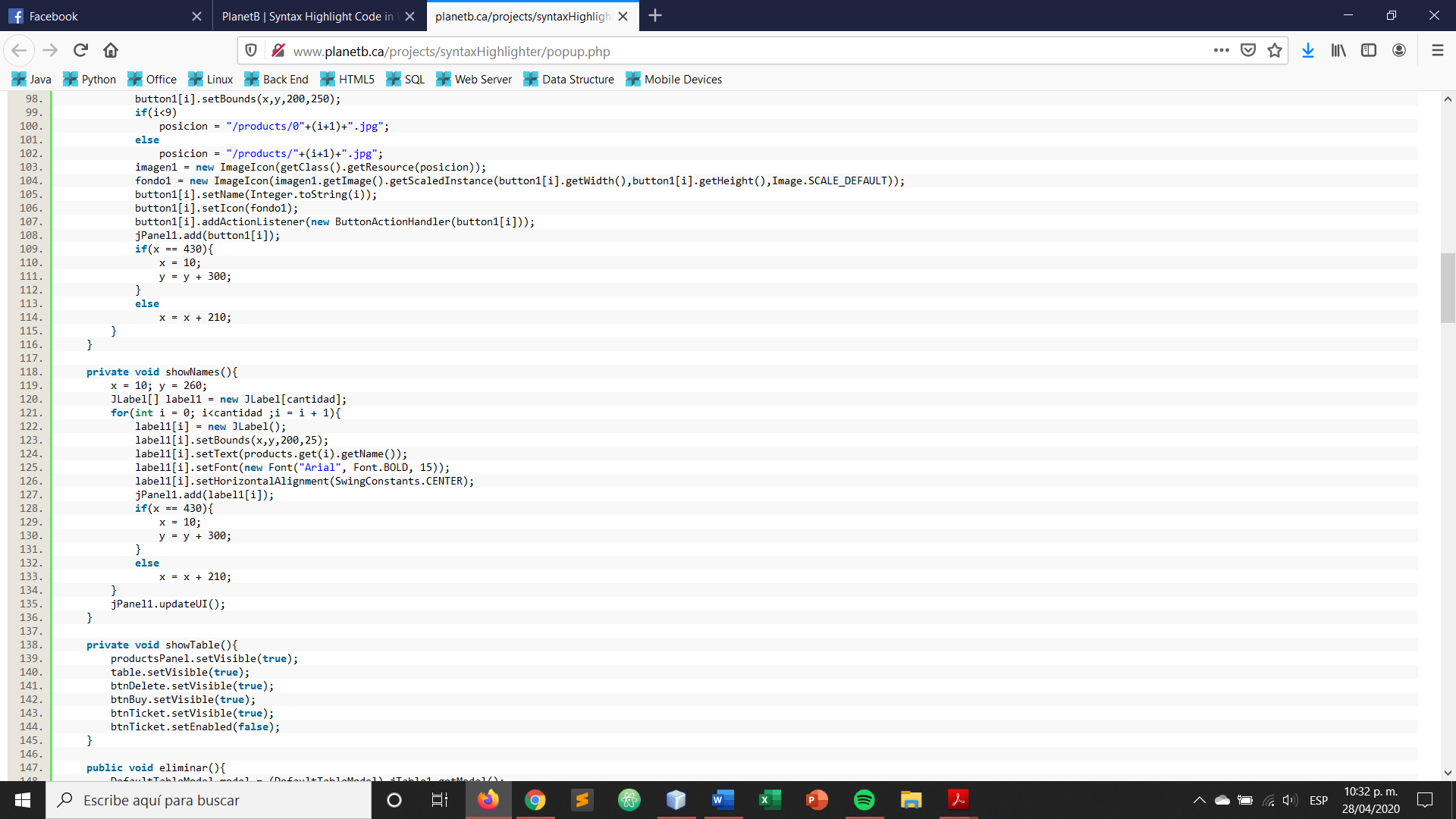
El resto de las funciones tienen que ver con cuestiones de diseño, por lo que son irrelevantes.

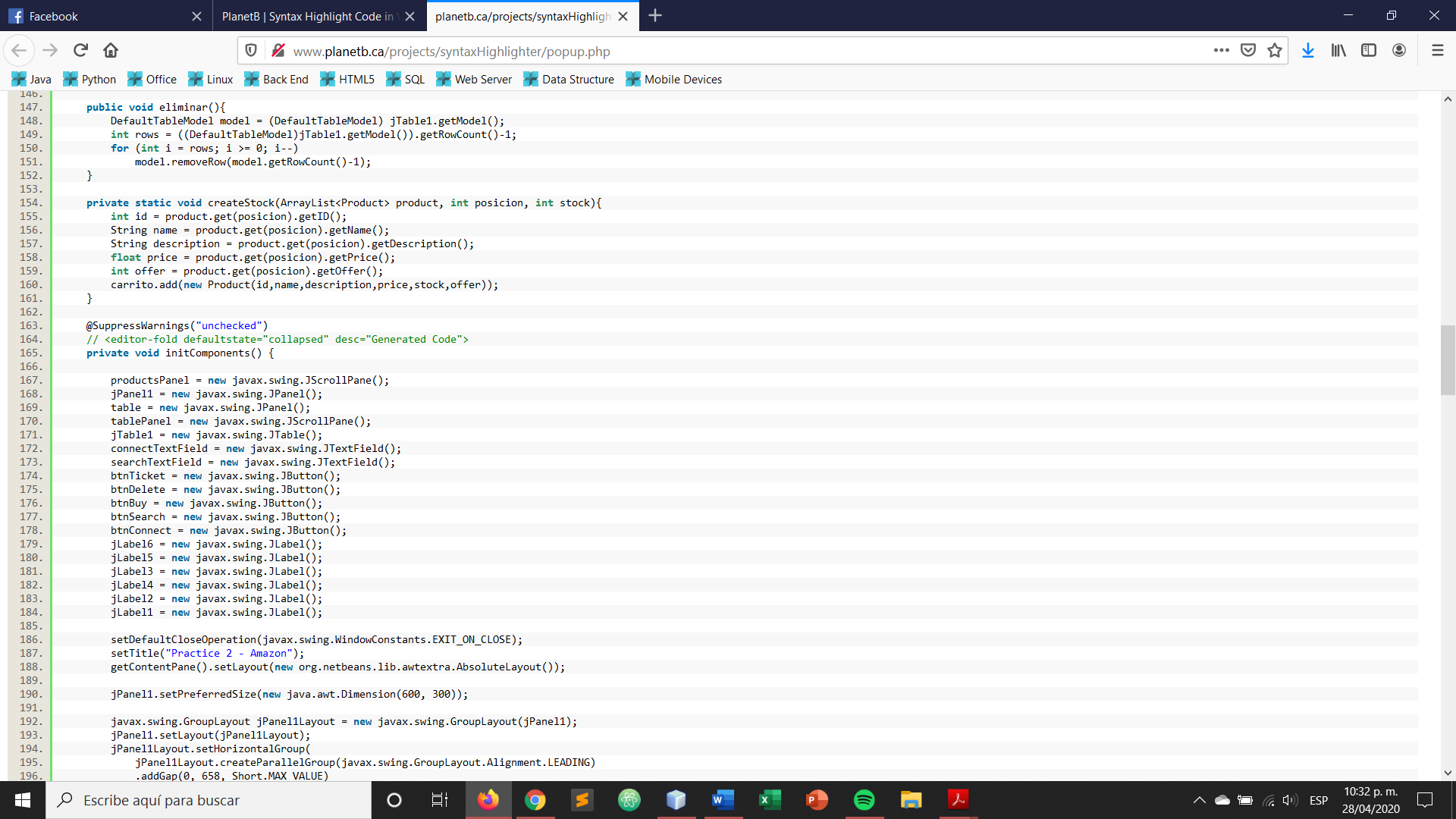
Lo que, si debemos explicar, es el comportamiento de los botones de dicha interfaz:

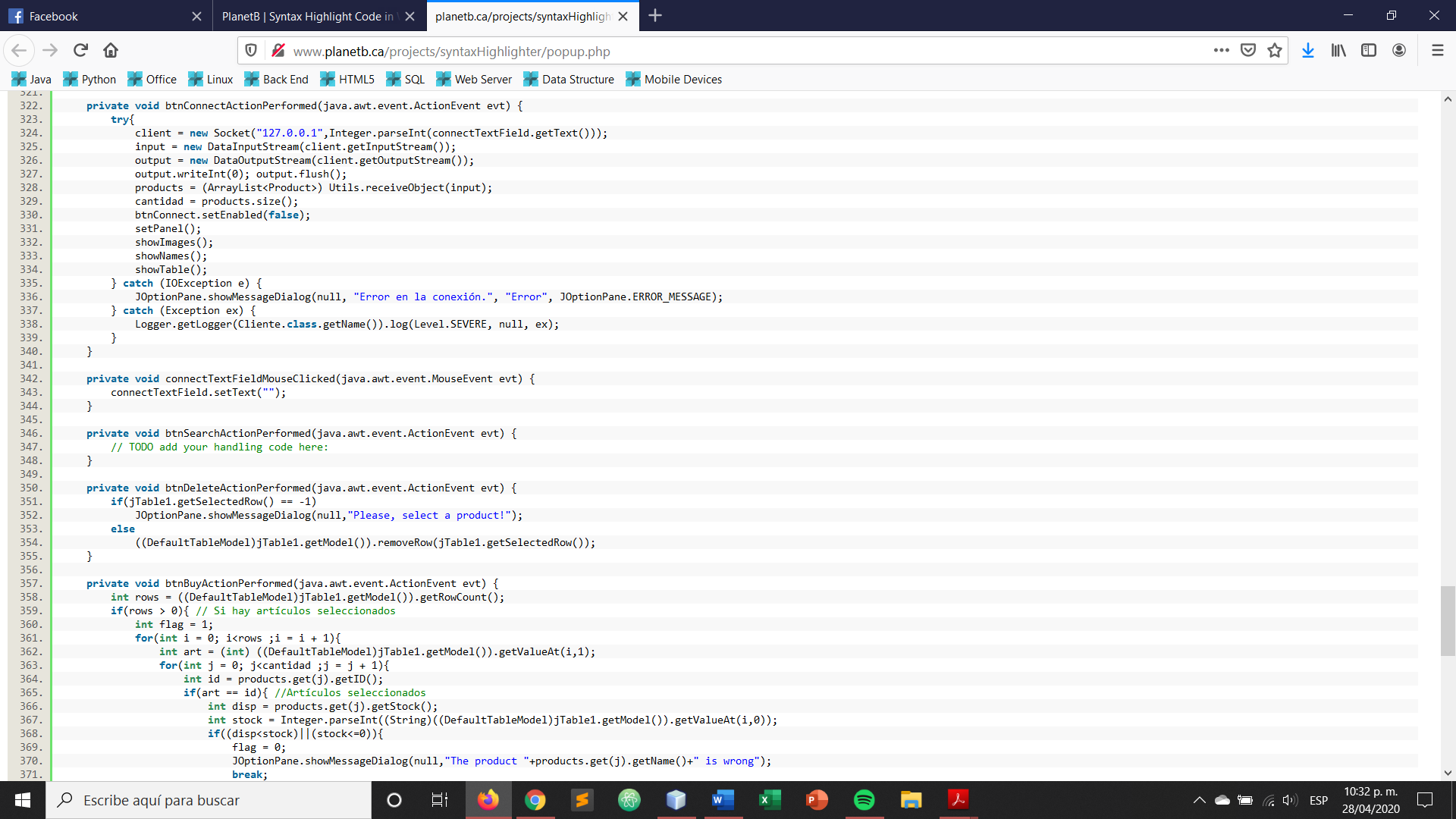
1. **Connect:** este es el botón que se encarga de realizar la conexión con el servidor. En él, se crea el socket correspondiente y los flujos de entrada y salida de datos. Además, al momento de realizar la conexión, se recibe el inventario de productos y se cargan las funciones de diseño.
2. **Delete:** este botón sólo funciona cuando ya seleccionamos un producto y se agregó al carrito. Su función, es eliminar el producto seleccionado y devolverlo al inventario para no alterar la cantidad.
3. **Buy:** este botón lo que hace es enviar al servidor el carrito con los productos seleccionados para que este, actualice el inventario global y devuelva una notificación de aprobación de productos.

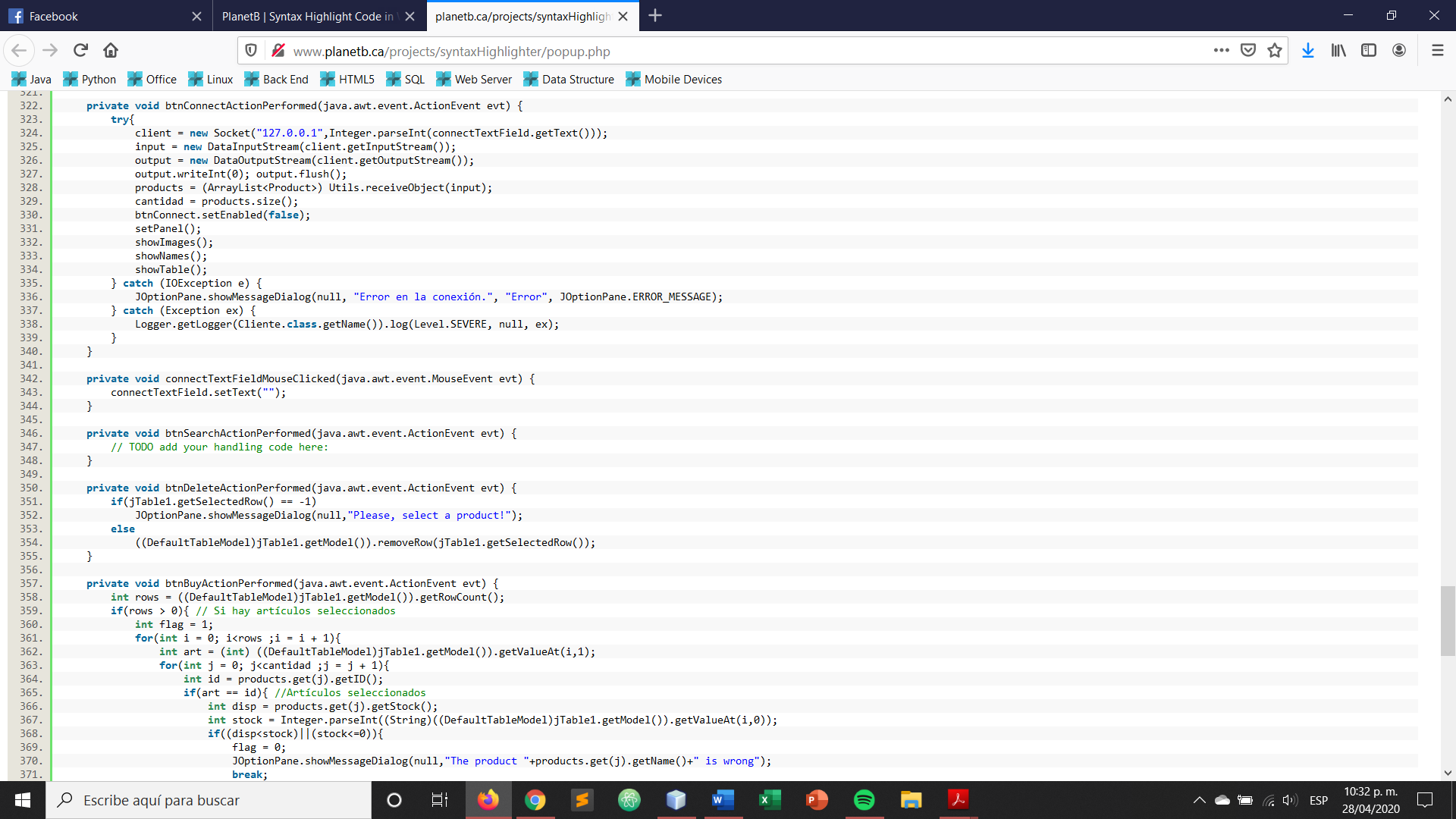
A continuación, se muestra el código del Cliente.

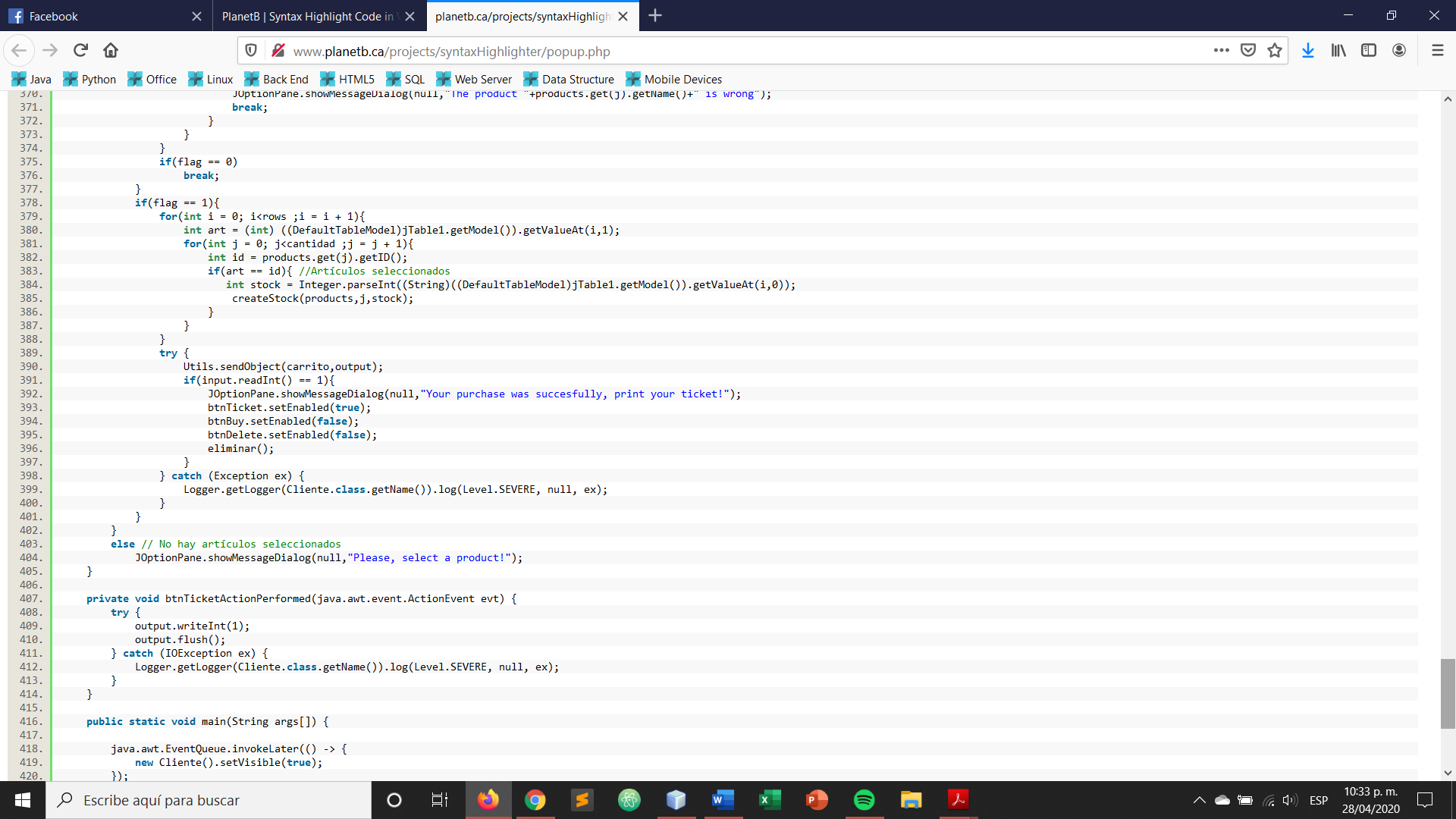


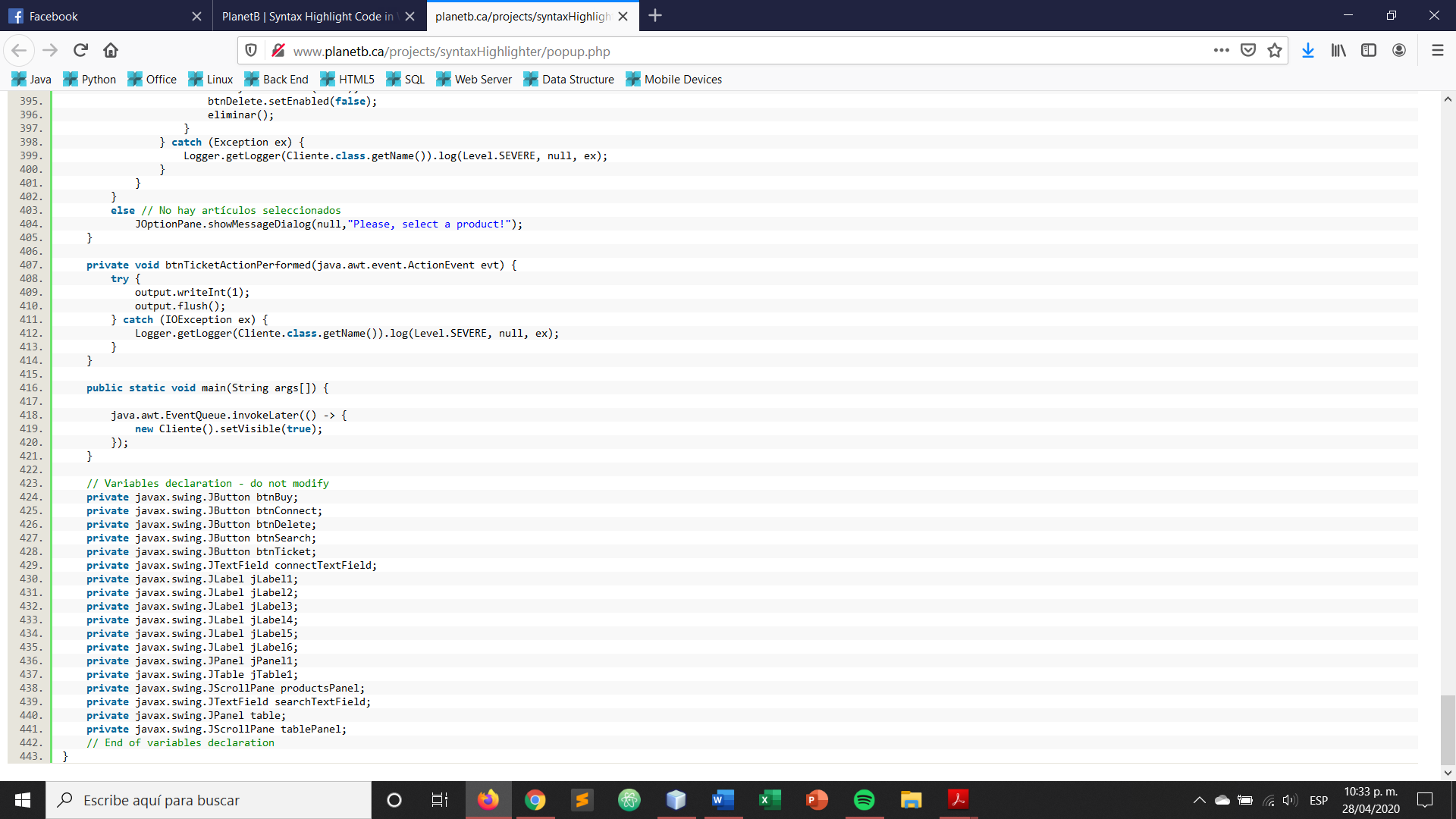








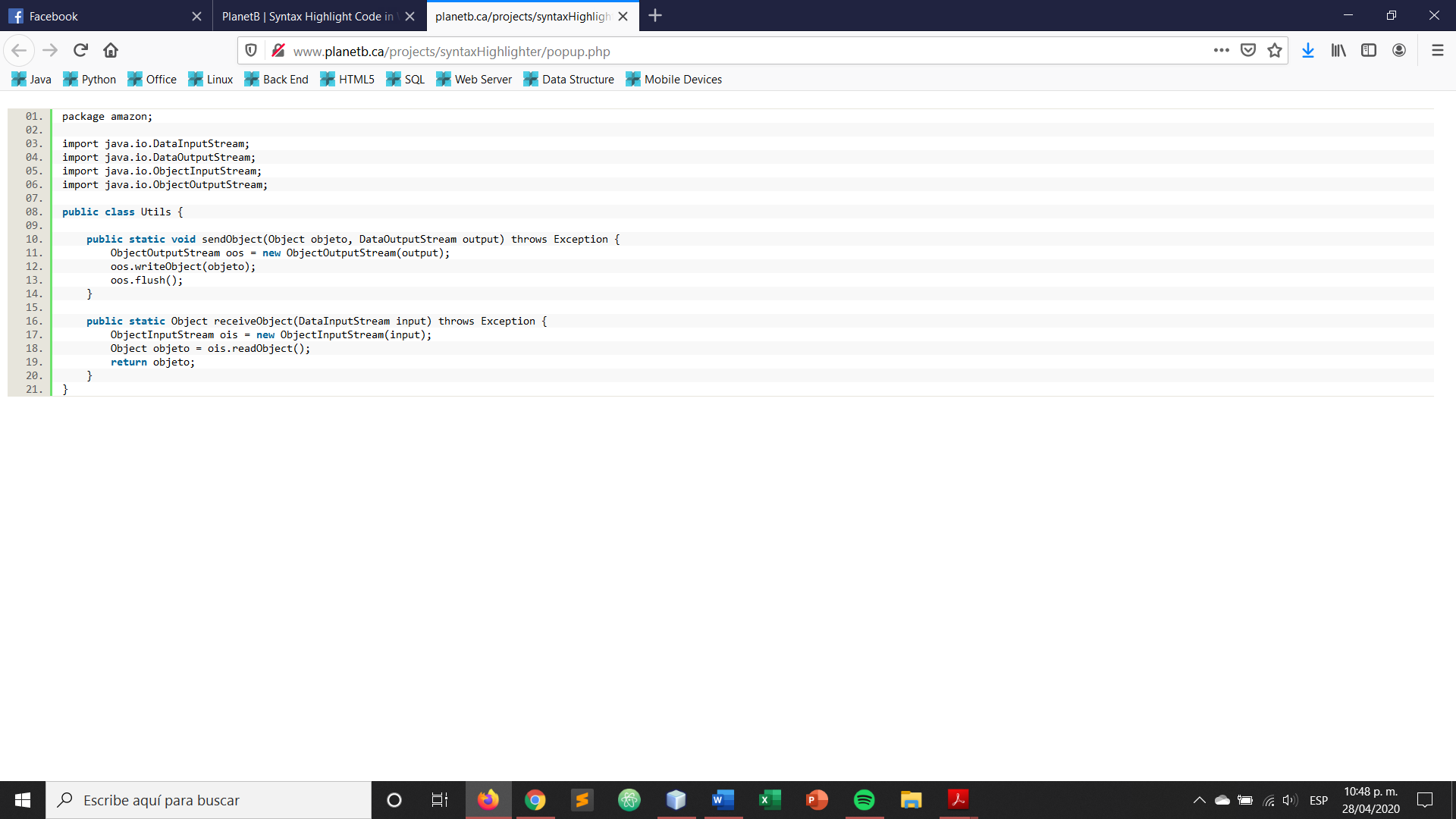




Para esta práctica, como su elaboración era un poco más complicada que la anterior, implementa otras clases para hacer un poco más general el envío de objetos, el manejo de productos y la creación del tipo de compra; por lo que a continuación se mencionará una breve explicación de dichas clases.

**Utils.java**

La clase Utils, únicamente contiene dos funciones y lo que realizan, como su nombre lo dice, es enviar y recibir objetos. Esto para facilitar la comunicación entre el cliente y el servidor. Además, permite que el envío de objetos entre ambas partes se realicé de una manera rápida y sencilla.



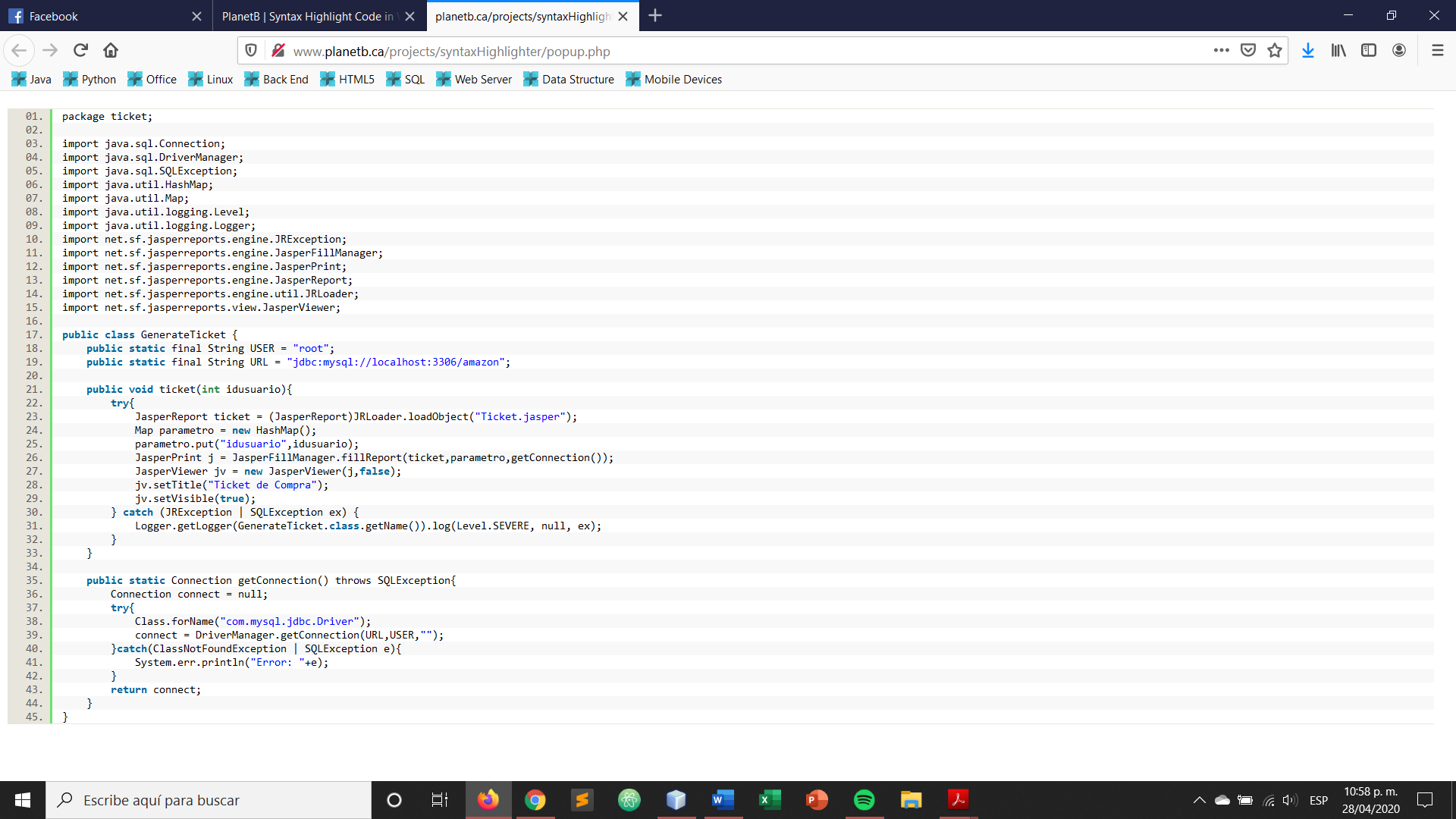
**Product.java**

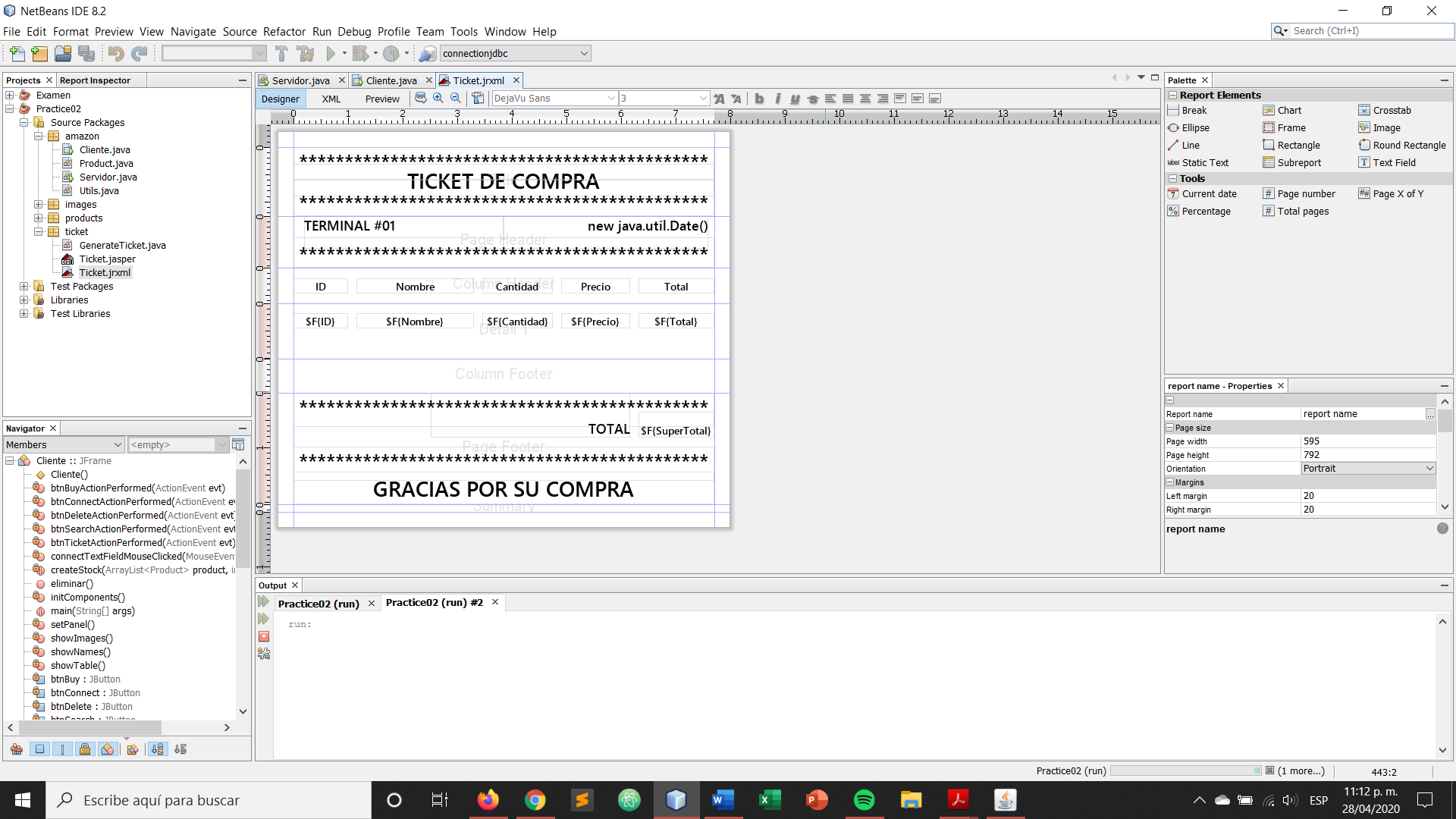
Esta clase esta implementada con la finalidad de manejar de una manera más sencilla los productos, ya que, como se mencionó en el Servidor, se crea un ArrayList de tipo Product por cada producto que el usuario ingresó. Únicamente se implementaron métodos getters con excepción del stock, ya que es el único que puede cambiar en tiempo de ejecución; el resto se mantiene hasta el final del programa.



**Ticket.java**

Esta clase se utiliza para la generación del ticket de compra. Utilizamos una herramienta llamada Jasper para manejar PDF dinámicos y lo que hace esta clase es llamar al objeto que creamos en Jasper.





FUNCIONAMIENTO DE LA APLICACIÓN

A continuación, se muestra una explicación del funcionamiento general de la aplicación.

Como se puede observar en la imagen 1, del lado izquierdo aparece un panel con los productos disponibles, estas imágenes son botones realizados dinámicamente para que no importe el número de productos, siempre se acomoden al tamaño del panel. Del lado izquierdo aparece un JTable, que representa el carrito de compras y es en donde se irán agregando los productos.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Imagen 1. Interfaz principal.

Una característica que agregue a la práctica fue la validación del carrito de compras. Puede suceder que el usuario intente agregar el mismo producto más de una vez, por lo que realizo una búsqueda rápida para ver cuales ya han sido seleccionados y mandar el error.

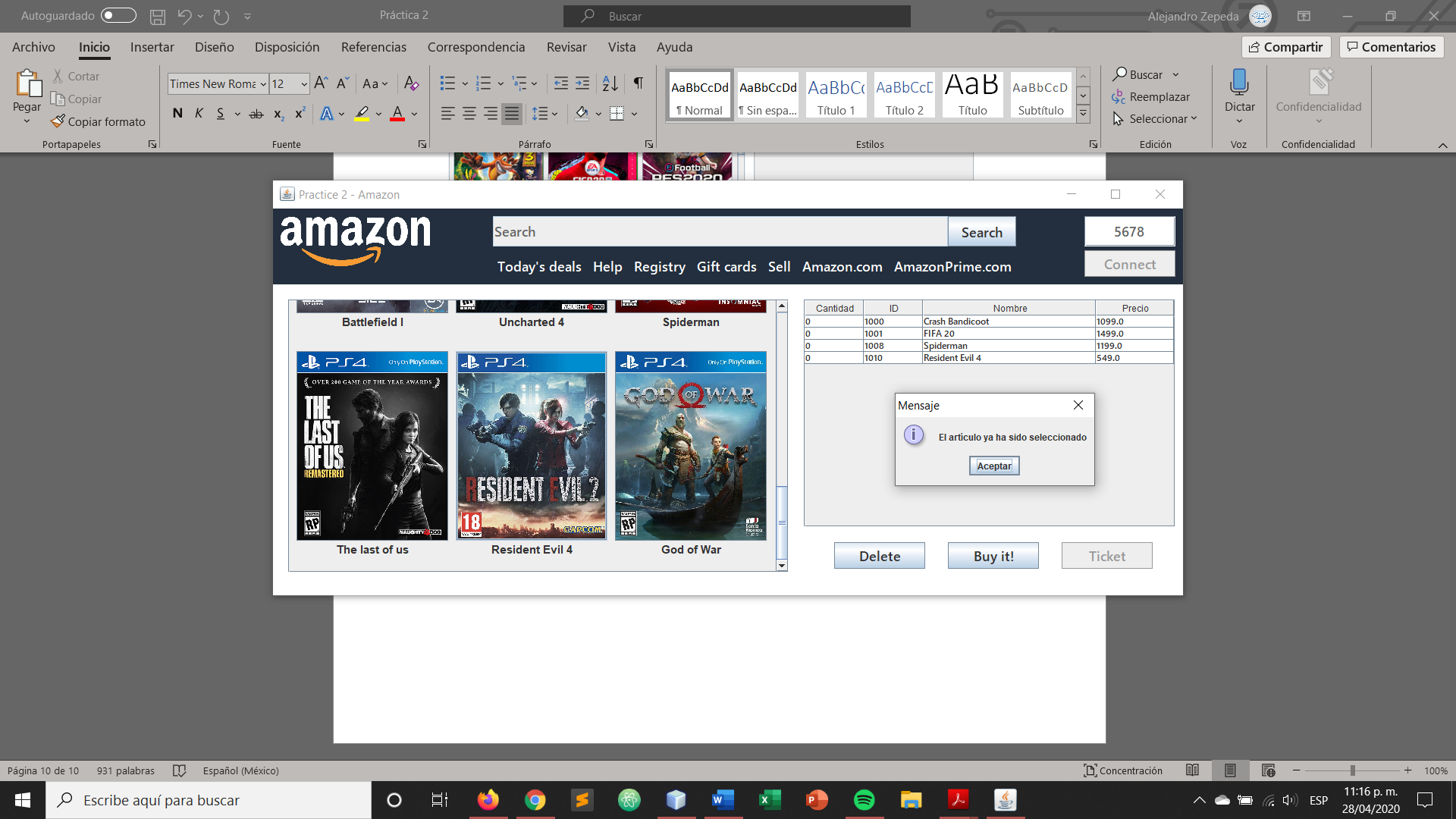


Imagen 2. Validación del carrito de compras.

De igual manera, el programa cuenta con una validación interna del número de productos seleccionados. Como se aprecia en la imagen 3, se muestra un mensaje indicando el producto cuyo número de artículos supera al disponible.

Captura de pantalla de computadora

Descripción generada automáticamente

Imagen 3. Validación del número de artículos.

Ahora, en la imagen 4, se observa que la compra fue realizada con éxito. Esto quiere decir que el inventario del servidor ya fue actualizado con éxito y está a la espera de que un nuevo cliente intente realizar otra compra.

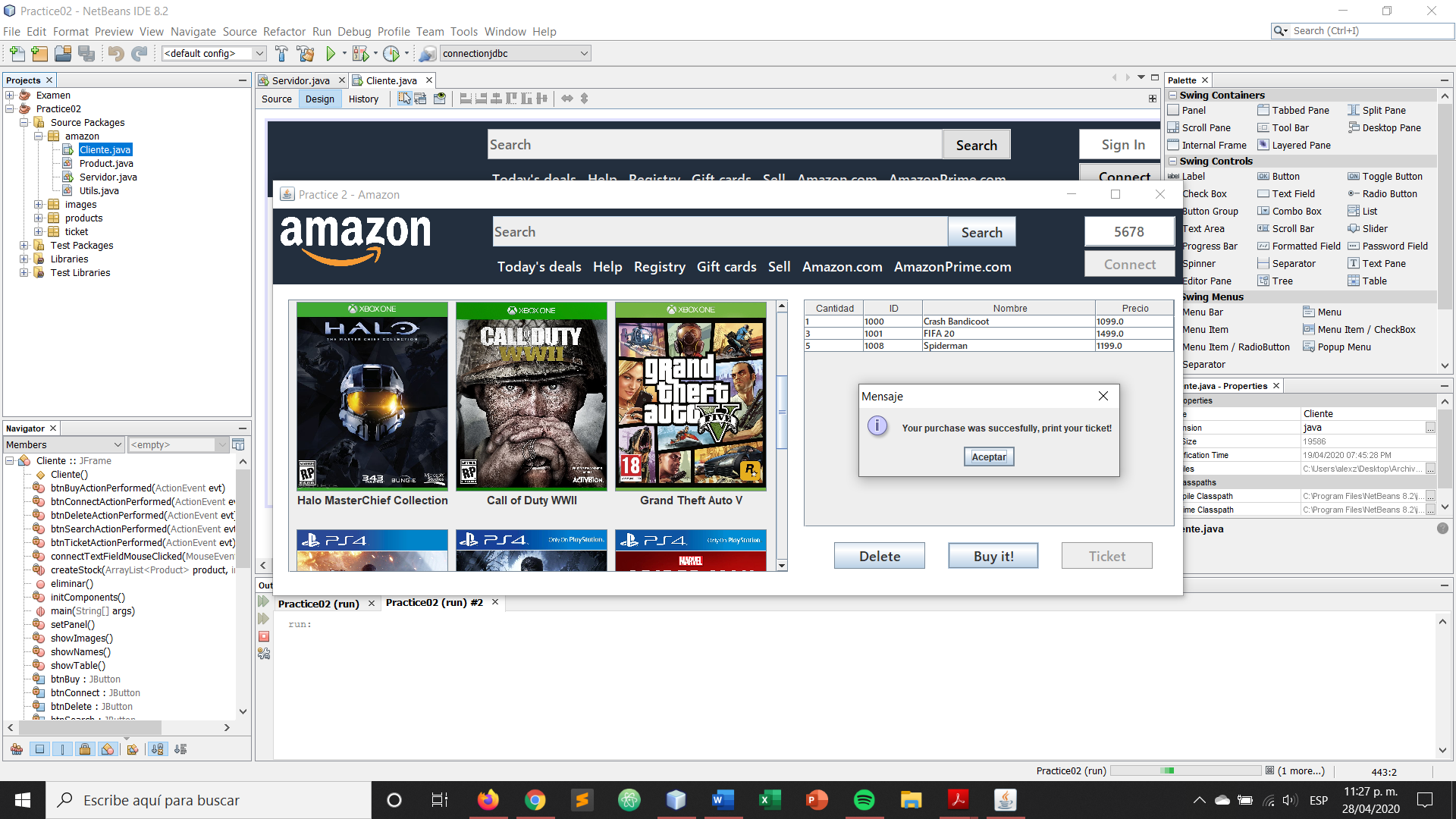
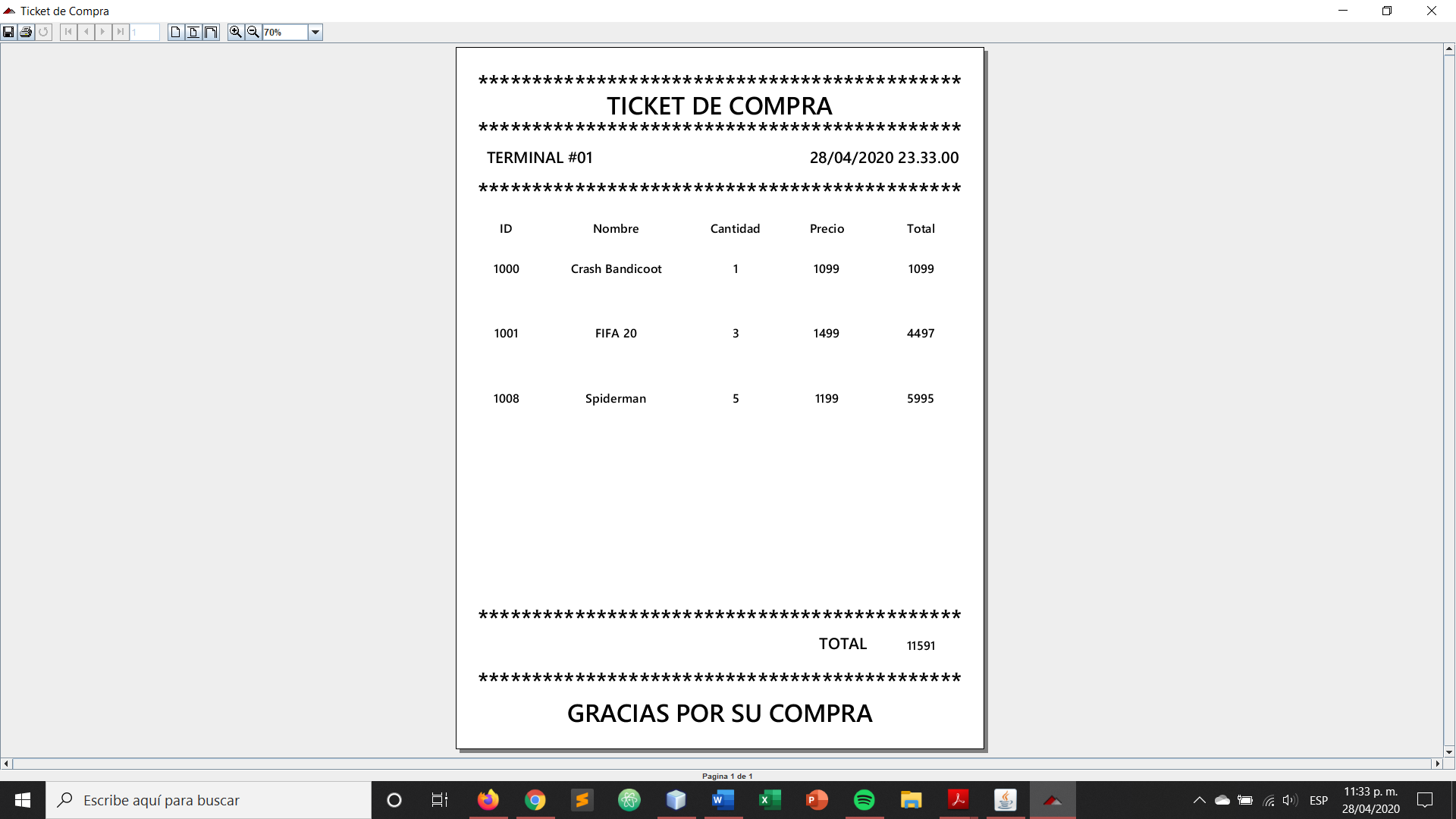


Imagen 4. Compra realizada.

Por último, se realiza la generación del ticket como PDF, se muestra en formato genérico.



COMENTARIOS GENERALES

Para la poder utilizar Jasper, fue necesaria la implementación de una base de datos. Todo lo necesario para poder correr el programa se encuentra dentro de la carpeta de este. Desde el archivo de texto donde se especificaron los productos hasta la base de datos, los conectores.jar y el archivo de Jasper.

CONCLUSIÓN

Aunque de momento puede sonar difícil la implementación del servicio de compras en línea, cuando realmente analizamos las posibilidades que tenemos para desarrollarla nos damos cuenta de que en gran parte tenemos los conocimientos necesarios para lograr el objetivo, aunque siempre es necesario revisar temas y partes puntuales del lenguaje de programación.

De primer momento pareciera que no existe relación con la practica 01, sin embargo, existe tal relación, ya que continuamos utilizando sockets, tanto del lado del cliente como del servidor para enviar y recibir flujos de datos y en cada extremo interactuar con ellos.

Por último, aunque considero que la práctica funciona de forma correcta aún existen aspectos que se podrían mejorar y sin duda alguna pienso retomarla en un futuro y trabajar sobre ella a fin de obtener un mejor resultado.