**SIMULADOR SUPERMERCADO**

## INTRODUCCIÓN:

Para empezar el proyecto decidí refrescar el funcionamiento de la autopista y ver qué similitudes hay entre estos dos productos software. El funcionamiento es muy parecido pero con algunos matices.

Para empezar es obvio que en este proyecto no trabajamos con vehículos sino que las estadísticas que queremos contabilizar son las de los clientes, esto lo conseguimos simplificando el concepto de cliente en carritos. Un carrito lo podemos ver como una estructura con la que albergar productos y estos a su vez los podemos caracterizar de distintas formas, en este caso nos interesa guardar el precio de cada uno. Y bueno ya estaría, no?

Pues no del todo, la autopista pedía al usuario como dato el tiempo medio de llegada y funcionaba siempre durante tres horas virtuales. Esta vez nos piden que podamos elegir el tiempo de simulación, el tiempo medio de llegada de carritos, la media de productos por carrito y la media de precio que tienen los productos. Para ello he decidido tunear la autopista un poco y lo he hecho por pasos, más concretamente en tres entregas, cada una evoluciona un poco con la anterior hasta llegar al producto final y terminado.

## PRIMERA ENTREGA:

El objetivo de la primera iteración es conseguir tener software funcionando, por eso hemos decidido que las metas sean fáciles de conseguir rápidamente. Implementamos un supermercado con una sola caja a la cual le llegan carritos cada minuto con un único producto de un euro. Al usuario le damos la capacidad de elegir cuánto tiempo quiere que dure la simulación.

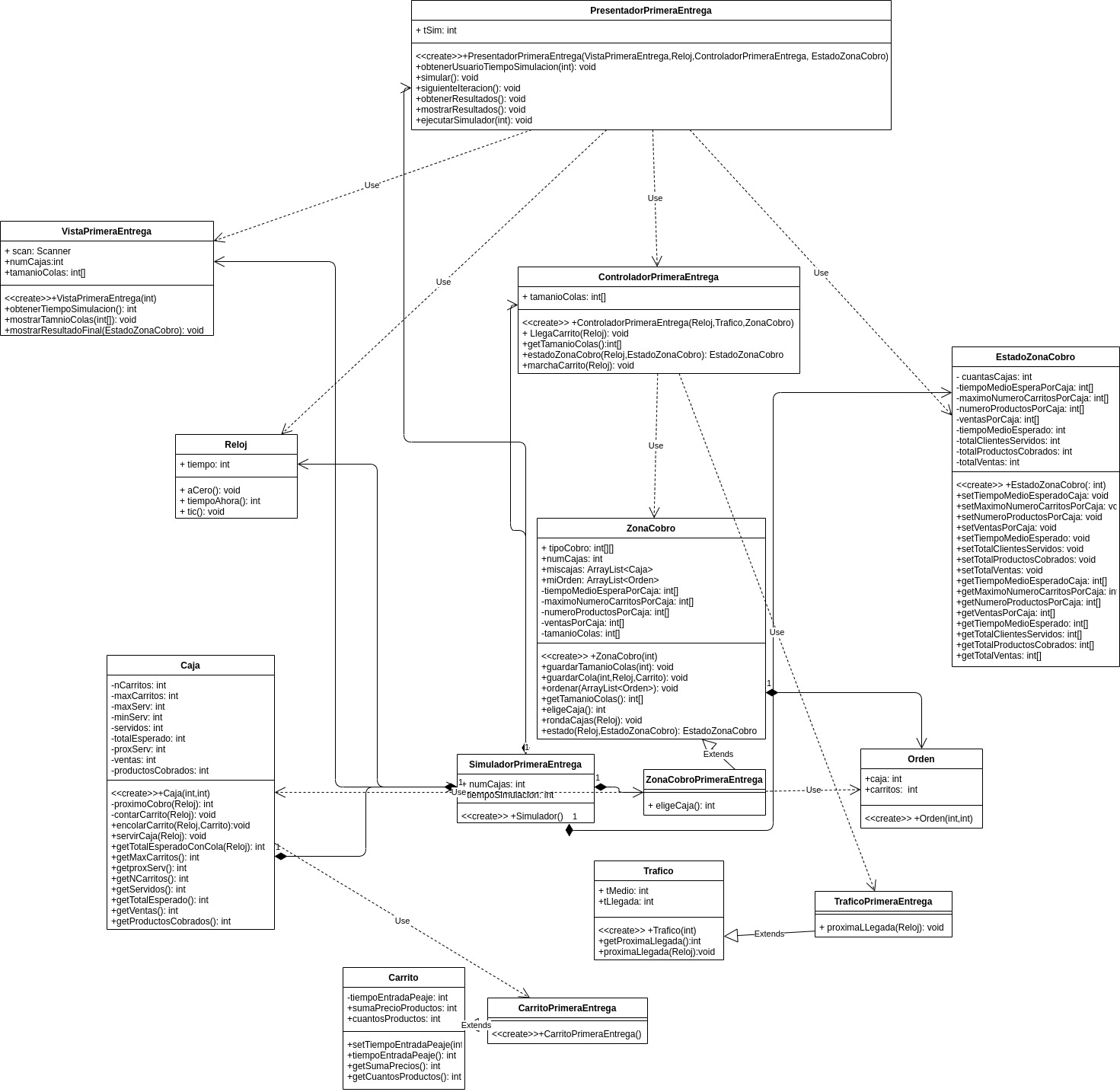
Para ello he decidido decantarme por el modelo vista-presentador, el cual me permite tener una mayor flexibilidad a la hora de mejorar el software en futuras ocasiones. Esta implementación ayuda mucho también para conseguir invertir el control, como verás más adelante.

Entrando más en el código, las clases que he cambiado son:

* **Simulador** ha sufrido un remodelaje total y ha pasado a tener la única función de instanciar las demás clases únicamente, este ha sido el cambio más fuerte y el que nos permite flexibilizar el código cerrándolo a cambios y abriéndolo a expansiones.
* **Controlador** está prácticamente igual, el cambio que se puede ver a simple vista es que esta clase devuelve los datos a imprimir encapsulados en una clase llamada EstadoZonaCobro en vez de imprimirlos directamente.
* Peaje pasa a denominarse **ZonaCobro**: he cambiado los valores de los intervalos de tiempo que tardan las cajas en cobrar a un carrito ya que antes tenían los valores de la autopista. Esta entrega al elegir una caja obviamente siempre elige la única caja que hay y además el resultado de la simulación se devuelve encapsulado como una clase en vez de ser impreso directamente.
* **Trafico** en esta entrega al ser constante devuelve como llegada del siguiente carrito el momento actual más el tiempo medio de llegada de carritos.
* En **Caja,** que es el equivalente de Cabina en la autopista,al contar los carritos cambian los parámetros que se actualizan.
* **Carrito** la he conseguido cambiando la clase Vehiculo de la autopista. He añadido dos variables sumaPrecioProductos y cuantosProductos las cuales nos dicen cuántos productos tiene cada carrito y cuánto cuestan en total. El método que las instancia les da valor 1 siempre a las dos.

También he creado clases desde cero:

* **Vista** la cual se encarga de recoger los datos del usuario vía terminal y de sacar los datos por pantalla.
* **Presentador** que es el nuevo simulador de la antigua autopista, se encarga de utilizar los recursos que le brinda el simulador y controla la vista también.

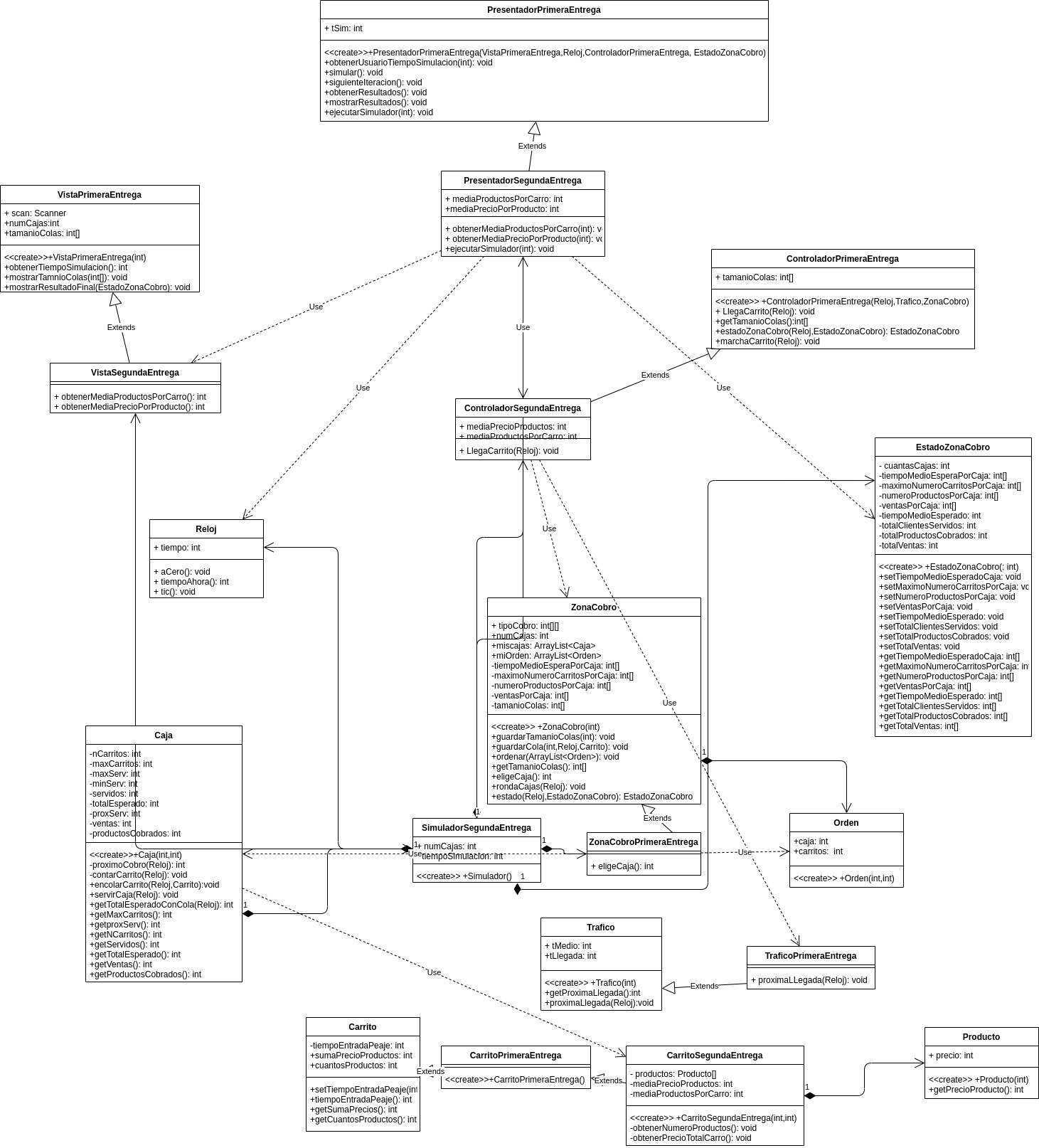


## SEGUNDA ENTREGA:

Para la siguiente versión del producto decidimos mejorar el carrito, le damos la capacidad de producir carritos con un número variable de productos y a su vez estos con precios variables.

Esto lo he conseguido cambiando la clase Carrito, la cual calcula el número de productos que contiene con una fórmula de distribución exponencial con una media dada por el usuario, esta media la obtenemos en la vista.

Para esta entrega he creado otra clase nueva Producto que tiene como variable su precio y que es calculado igual que el número de productos que tiene el carrito pero con otra media independiente.

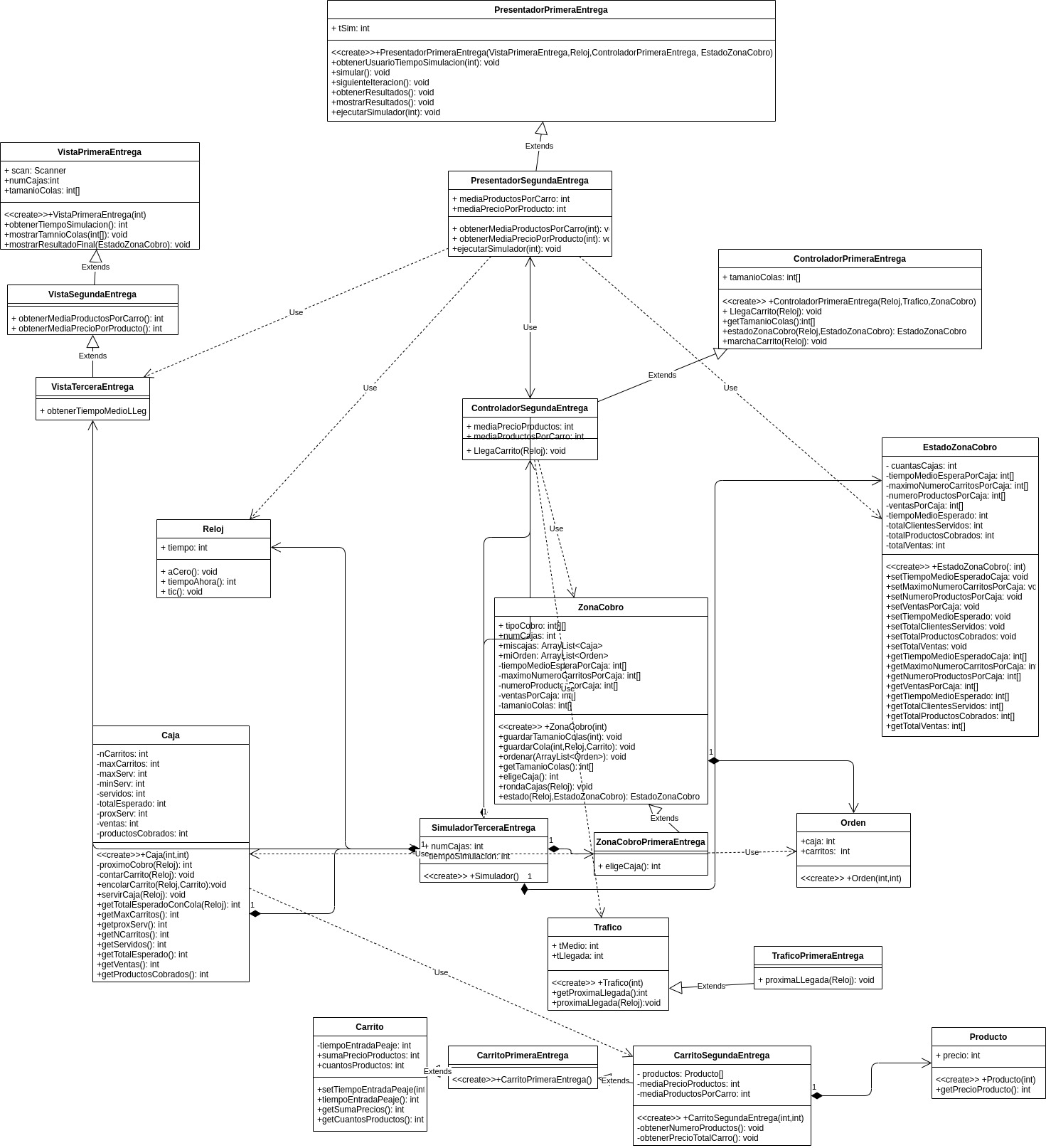


## TERCERA ENTREGA:

Para la tercera entrega decidimos aumentar el número de cabinas que tiene el supermercado y hacer que los carritos lleguen en intervalos no fijos de tiempo.

Para conseguir que los carritos no lleguen en tiempos fijos de tiempo he utilizado la implementación que ya teníamos en la autopista, ya que me venía de perlas.

El número de cabinas como hasta ahora era un dato que le pasaba el simulador al presentador lo único que he hecho ha sido pasarle un 10 en vez de un 1. Aunque sí que es verdad que he utilizado la función que elegía cabinas, en este caso cajas, de la autopista un poco modificada en vez de la utilizada hasta ahora. En esta función he cambiado las probabilidades de elegir una de las tres cajas con menos cola y añadido una probabilidad del 0.1 de ir a una caja aleatoria.

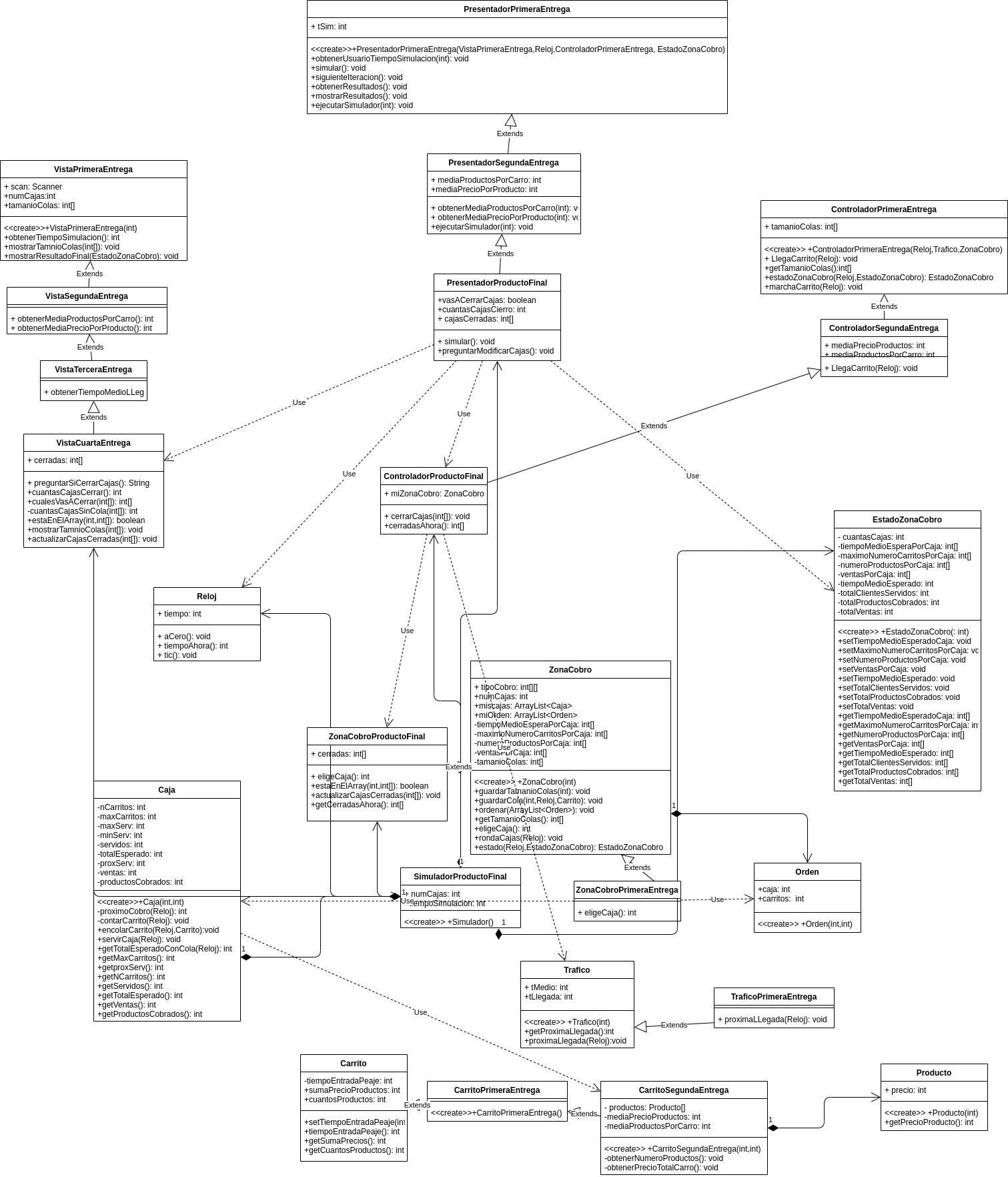


## PRODUCTO FINAL:

Por último falta la posibilidad de abrir/cerrar cajas cada hora contando anteriormente con los datos de las colas que hay en cada caja. Esta parte junto con la primera entrega han sido la más retantes a la hora de diseñar el funcionamiento del algoritmo.

Tras unas horas de pensar detenidamente cómo iba a atacar el problema finalmente decidí preguntarle al usuario si quiere abrir/cerrar alguna caja, en caso positivo cuantas, sin dejarle que modifique la apertura de más cajas que las que hay sin cola, y cuales van a ser las que va a cerrar/abrir.

Al principio de la simulación todas las cajas están abiertas y al pasar una hora las cajas que están cerradas se actualizan si así lo decide el usuario. Si alguna caja está cerrada no es ni siquiera candidata para los clientes a la hora de elegir caja.



## CONCLUSIONES:

Aplicar el modelo vista-presentador, la inversión de control, la programación orientada a objetos... evita muchos fallos a la larga y permite evolucionar el proyecto con una facilidad mucho mayor que sin utilizarlas.

Lo malo es el comienzo, necesita de una compresión del producto a otro nivel, lo cual obliga a documentarse y pensarlo todo mucho más, esto se traduce en una lentitud mayor a la hora de conseguir que el proyecto arranque.

La parte buena es que una vez montada la base del edificio es mucho más fácil construir sobre una estructura fuerte y testada. Además de que como el software está en continua comprobación, si falla algo sabes que el problema está en la última parte añadida. Así te ahorras mucho tiempo de localizar dónde puede estar el error y puedes centrarte más en cómo solucionarlo.

Resumiendo, esta forma de programar es una inversión de esfuerzo en el trabajo que estás realizando al inicio, es una pérdida de agilidad a corto plazo, que se amortiza sola a medio y largo plazo, trayendo grandes beneficios.