Instituto de Tecnología ORT
Carrera de Analista de Sistemas
Materia: Programación 2

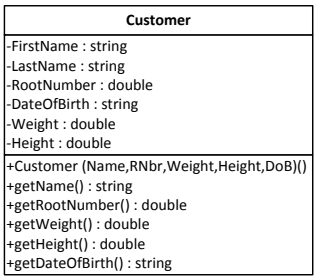
Unidad 4 – Práctica
Ejercicios Cortos

Para todos los escenarios:

a. Indicá el patrón o patrones aplican al escenario
b. Realizá un Diagrama de Clases y/o secuencias donde corresponda.
c. Indicá las notas o anotaciones que sean necesarias para explicar el diseño.
d. Verifiquá su diseño con sus pares.
e. Se sugiere que te apoyes también en las prácticas SOLID+ como pautas de ayuda.

1. Customer – Cliente

Un equipo de desarrollo de tu empresa de otro país, desarrolló una clase y te la pasan ahora (solo la clase pero no su fuente) para que la reutilices en tus proyectos locales. Tu líder la estuvo mirando te advierte que se van a necesitar cambios dado que como está no serviría completamente. Te pide que pienses una solución que permita usarla como está sin cambiarla, pero haciendo los cambios que se necesita, y te explica debajo:

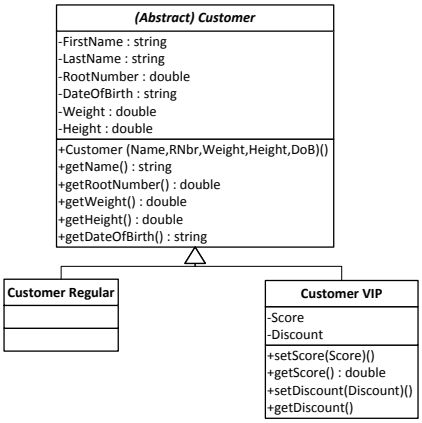


Problemas y cambios por hacer:

- Las variables y métodos están en inglés, pero los clientes locales pidieron que el código y los mensajes esté en español.
- El “rootNumer” en nuestro país es el DNI, y nos piden que manejemos el atributo con ese nombre.
- El peso (“weight”) y la altura (“height”) están en libras y pies, y en los sistemas locales se deben usar en kilogramos y metros. La solución deberá indicar alguna lógica de conversión de medidas.

2. Regular vs. VIP

La solución del escenario anterior funcionó por un tiempo, pero nos avisan que aquel equipo de desarrollo no solo introdujo cambios en su clase Customer, sino que también los proyectos de ellos van a interactuar con los proyectos nuestros. Tu líder te pide que revises tu solución dado que ahora nuestros sistemas van a recibir “customers” desde sistemas de ellos y que además esos clientes pueden ser de dos subtipos regulares o vip (con diferencias entre sí). Te lo explican debajo:



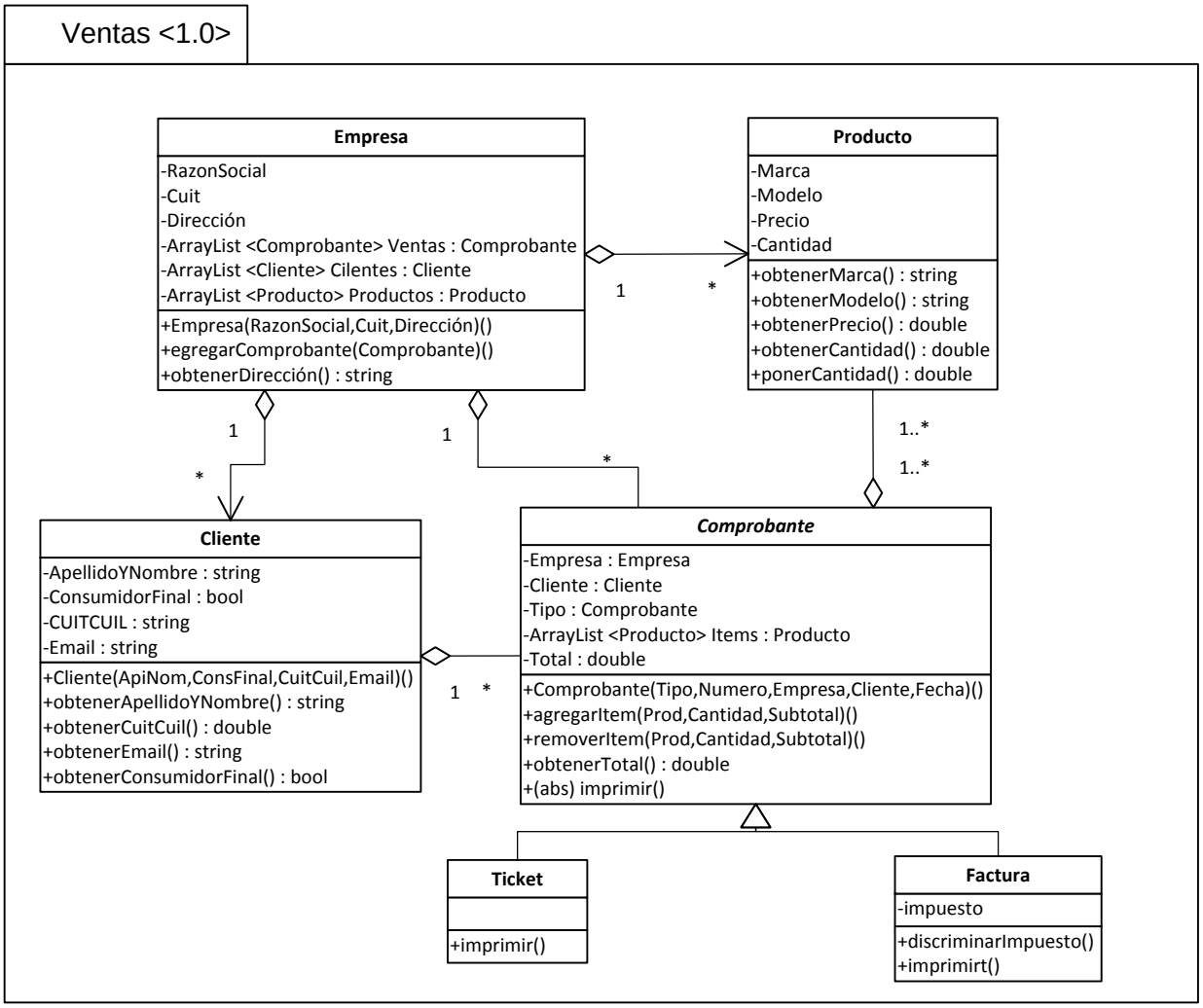
Problemas y cambios por hacer:

- Ahora tenemos subtipos de customers, pero a tu Líder, no le interesa el subtipo VIP por ahora y tampoco quisiera hacer un refactory para manejar el subtipo Regular.

- Tus proyectos pueden recibir un Customer (de cualquier tipo) en cualquier momento. Nos piden que revisemos la solución anterior para ver si la podemos soportar, o que la mejoremos en el caso que sea necesario para que nuestros sistemas no sufran impacto.

3. Ventas 2.0

Hace un tiempo, parte del equipo en el que estás hizo un sistema simple de ventas (representado en parte por el diagrama de clases de abajo) para una empresa cliente que vendía sus productos en sus locales. El sistema dio resultado, ahora la empresa quisiera reutilizar este mismo sistema (o módulo) para vender desde otros canales y conectarlo con sistemas contables. Tu equipo ahora revisa aquel primer diseño para reutilizarlo y poder conectarlo con otros componentes o sistemas externos.

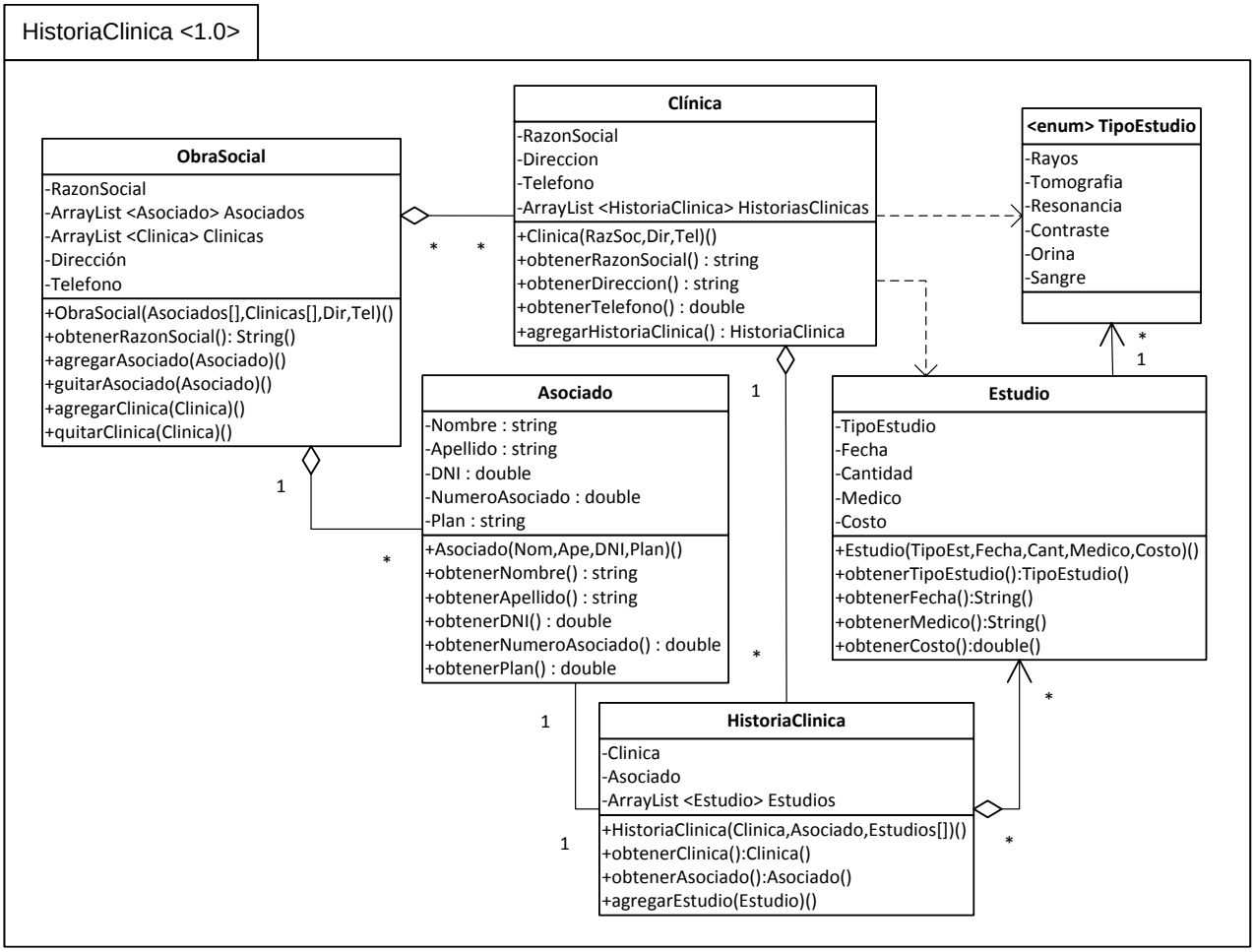


Requerimientos para los cambios:

- El negocio quisiera que se modifique lo menos posible lo que existe, permitiendo le lleguen las ventas desde fuera sin que se rompa nada.
- La gerencia de Finanzas pide que se modifique de alguna manera de modo que se puedan extraer las ventas hechas (el negocio dice que si a eso pero tampoco se rompa nada).

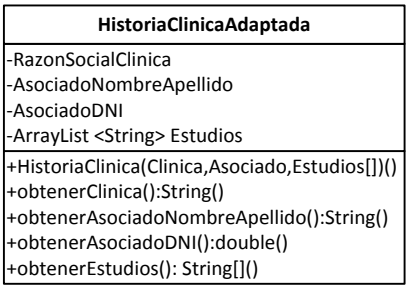
4. Historias Clínicas

Tenemos un sistema de historias clínicas que tiene que conectarse con otros para compartir las historias clínicas. El intercambio implica tener algún punto de acceso al sistema que permita consultar las historias clínicas por DNI exportándolas, y recibir también historias de otros sistemas externos. Se llamó a una reunión para recordar lo que existe hoy y ver que se va a recibir y que hay que exportar. Te piden que pienses en algo para la reunión.



Requerimientos para los cambios:

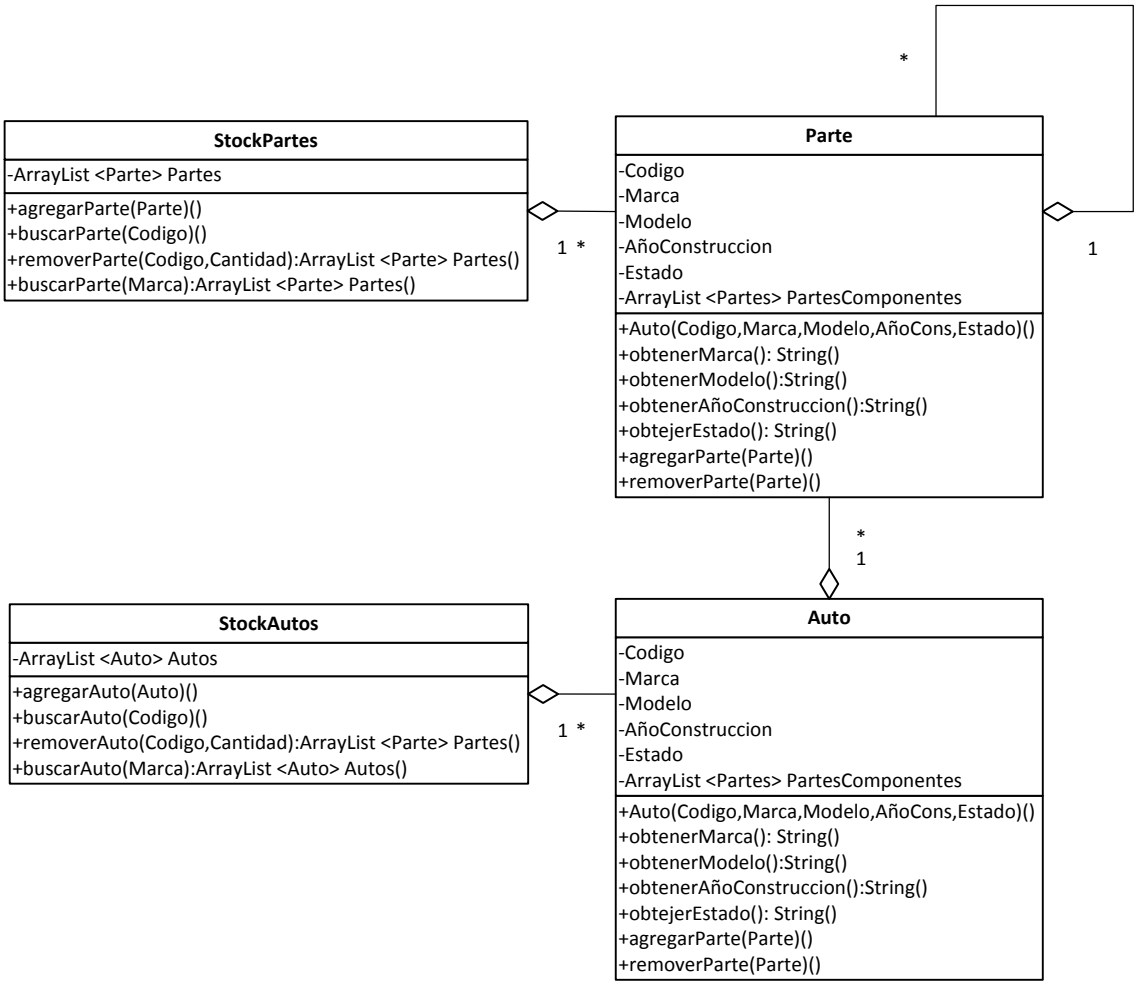
- El negocio anticipó que no puede probar todo el sistema por este nuevo cambio.
- También pidió que para los datos que se exporten no se muestren los costos de los estudios.
- El arquitecto comentó al pasar que seguro habrá que usar dos o tres patrones para completar la solución (inclusive alguno creacional).
- Se comenta que se negocia un solo objeto HistoriaClinica adaptada para importar y exportar. Te adjuntan un formato -> sugerido para que lo adaptes al tuyo.



5. Automotriz

Este cliente necesita un refactory en sus sistemas de stock y ventas. El sistema actual tiene componentes viejos (pensados cuando solo se vendían autos y no partes como ahora), y la semántica

se fue distorsionando con el tiempo de forma que se ha vuelto complejo de mantener. Un analista hizo un diagrama de clases de lo actual para que tu equipo lo estudie y de una solución.



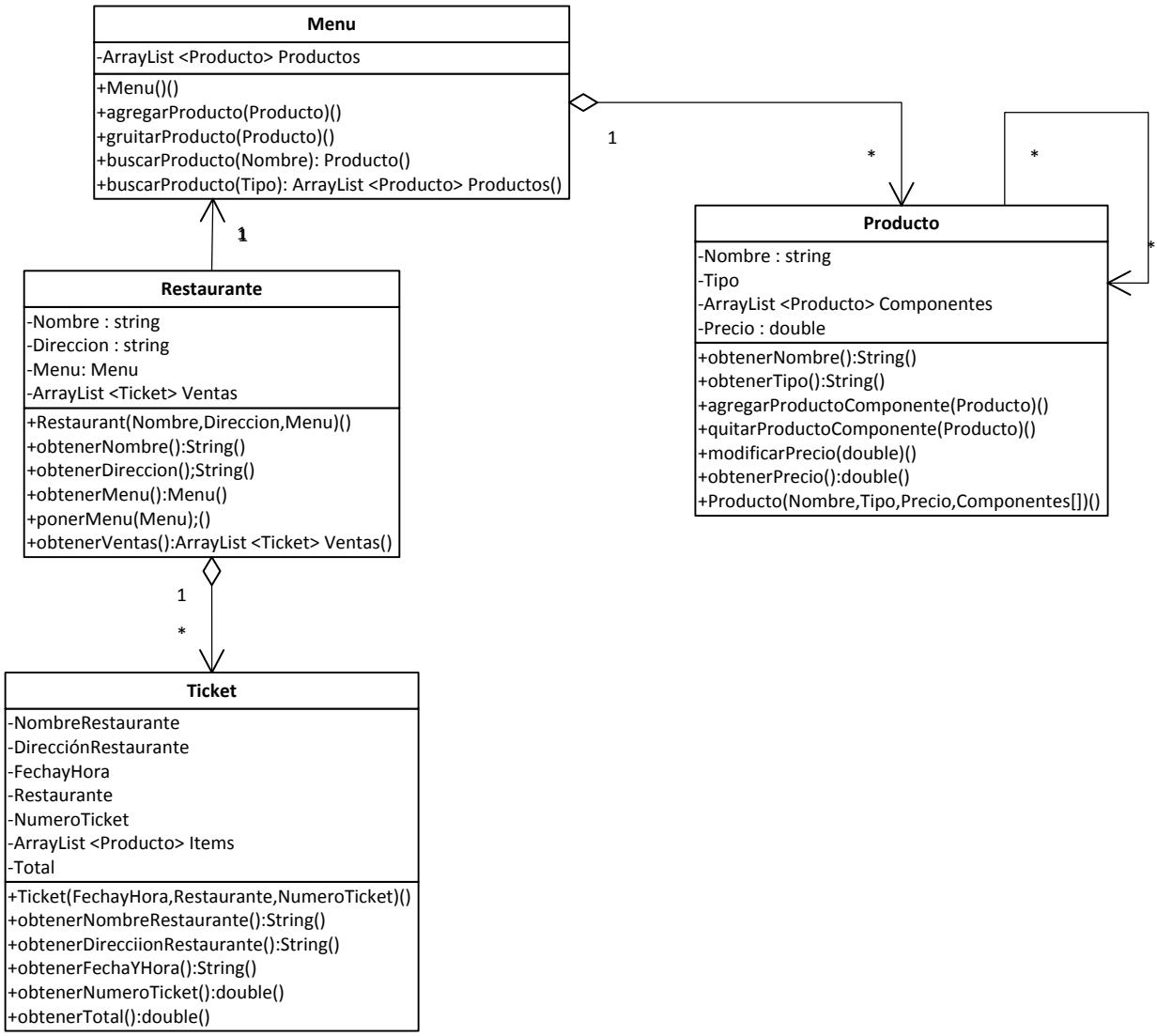
Problemas y cambios por hacer:

- El equipo técnico quisiera que el resto de los módulos o sistemas que trabajan con este módulo no tuvieran saber a cuál clase stock (de autos o de partes) tienen que consultar cuando se necesita algo.
- Tampoco quisieran tener que conocer a las clases stock, ya que cuando estas cambian se descompilan otros módulos, u otros sistemas dejan de funcionar.
- Quisieran también que las clases no estuvieran sueltas.
- Un compañero con más años, es optimista y piensa que se puede resolver fácilmente, quizás usando dos patrones combinados.

6. Comida Rápida

Tu empresa compite en una licitación para el desarrollo de un módulo de ventas para un restaurante de comida rápida conocido. El cliente probó sistemas enlatados pero les resultó complejo y caro. El analista relacionado con el proyecto adelantó un primer diseño para que el equipo lo mejore. El equipo piensa que puede funcionar, pero que necesita mejoras.

Instituto de Tecnología ORT
Carrera de Analista de Sistemas
Materia: Programación 2



Problemas y posibles mejoras:

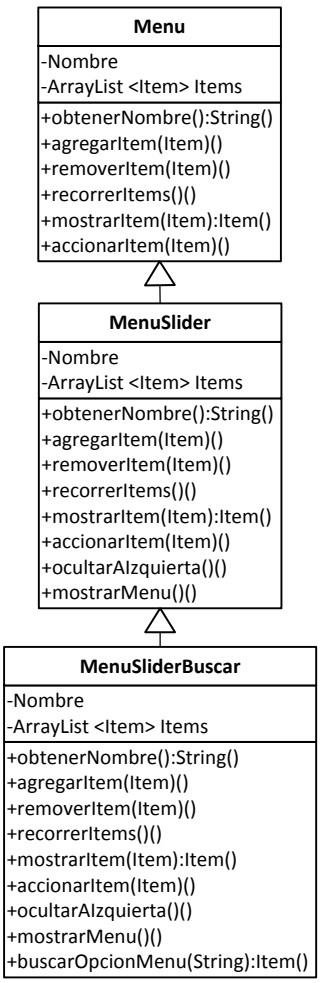
- La idea de un producto con una relación con sí mismo está bien, pero para este caso podría traer algunos problemas. Tus compañeros piensan que con un ligero cambio y la implementación de un patrón se mejora considerablemente.
- El analista cree que en un producto con componentes, se podrían manejar descuentos fácilmente.
- El arquitecto opina que al módulo le falta un punto de entrada para que esta funcionalidad sea reusable por otros módulos.

7. Mejora de Componentes

Un equipo de tu empresa se ocupa de crear y mejorar componentes visuales. Crearlos no es el problema, sino mejorarlos.

De momento no pueden ver una manera ágil de mejorar sus componentes sin tener que hacer refactory en todos los sistemas que lo usan.

Debajo te muestran una familia de componentes típica para que estudies una solución.



Problemas y posibles mejoras:

- Esta familia de componentes está simplemente extendida sobre clases concretas, y el arquitecto piensa que hay mucho acoplamiento de clase.
- Seguir extendiendo de esta manera con más funcionalidad es un problema por el tamaño de la generalización. Debemos buscar otra manera.