PATRONES DE DISEÑO

Ingeniería del software 2

Universidad Pablo de Olavide

25 DE MARZO DE 2019

Fernando Moreno Vidal

Índice

1. Introducción a patrones.

1.1. ¿Qué es un patrón de diseño?

Los patrones de diseño son estructuras que proporcionan soluciones ya existentes a los diferentes problemas que se plantean en la implementación de un programa. Para que un patrón sea considerado como tal, han de cumplir las siguientes características:

Ser adaptables en función al problema el cual se quiere dar solución.

Ser reutilizables, es decir, poder solucionar problemas similares en distintas circunstancias.

1.2. Tipos de patrones.

Dependiendo de la función desempeñada por cada patrón, estos se pueden clasificar en los siguientes tipos:

Creación: son los encargados de la creación de los objetos.

Estructura: son los encargados de la composición de las clases y los objetos.

Comportamiento: son los encargados de como interactúan las distintas clases u objetos.

1.3. Propósito de la aplicación.

En el siguiente diseño, se pretende implementar una aplicación mediante el uso de patrones, la cual apoye la gestión de una inmobiliaria, tanto de viviendas de nueva construcción así como de inmuebles ya edificados para su posterior venta.

Se adjuntará la implementación del sistema de gestión de inmuebles, contratos y clientes, cuyo objetivo principal consiste en centralizar todos los datos en un único sistema.

A petición del cliente se desarrollará la aplicación desde cero en lenguaje Java bajo entorno de programación NetBeans 8.2 para la implementación personalizada exigida por el cliente.

Además de la solución implementada, se adjuntará junto a la documentación, tanto el diagrama de clases correspondiente a dicha aplicación, así como los diagramas de clases desde los puntos más importantes de dicha aplicación.

2. Patrones utilizados.

En esta sección se describirán los diferentes patrones en los que nos hemos apoyado para la implementación de la solución para llevar a cabo las distintas acciones requeridas a la aplicación por parte del cliente.

2.1. Factoría.

Clasificación: Patrón de creación.

Intención: Separa la clase que crea los objetos, de la jerarquía de los objetos a instanciar.

Utilidad en nuestro proyecto: • Constructora: Clase encargada de la construcción de viviendas y pisos a partir de una dirección y rangos en cuestión de número de la calle y en caso de ser pisos, número de plantas. • FactoriaDAO: Clase encargada de la instanciación de los diferentes objetos tipo DAO de nuestro sistema.

2.2. Singleton.

Clasificación: Patrón de creación.

Intención: Asegurar que una clase tiene una sola instancia y proporcionar un punto de acceso global a ella.

Utilidad en nuestro proyecto: Nos conviene que solo haya una factoría de creación de entradas por lo que usamos el patrón Singleton para asegurarnos que solo se creará una factoría.

2.3. Observador.

Clasificación: Patrón de comportamiento.

Intención: Definir una dependencia 1 : n de forma que cuando el objeto 1 cambie su estado, los n objetos sean notificados.

Utilidad en nuestro proyecto: Utilizamos el patrón observador para tener actualizados el estado del inmueble, de modo que si un inmueble se pone a la venta, se informarán a la cartera de clientes suscritos a dicha opción.

2.4. Data Access Objeto (DAO).

Clasificación: Patrón de Creación

Intención: Define una clase con operaciones CRUD (crear, leer, actualizar y borrar)para objetos de un tipo concreto.

Utilidad en nuestro proyecto: Mantener la persistencia de los datos de nuestra aplicación mediante un sistema de archivos almacenados de manera local.

2.5. Memento.

Clasificación: Patrón de comportamiento.

Intención: Reestablecer los valores actuales a partir de un punto de restauración previo.

Utilidad en nuestro proyecto: Mantener una copia del sistema en memoria mientras se ejecuta la aplicación.

2.6 Decorator.

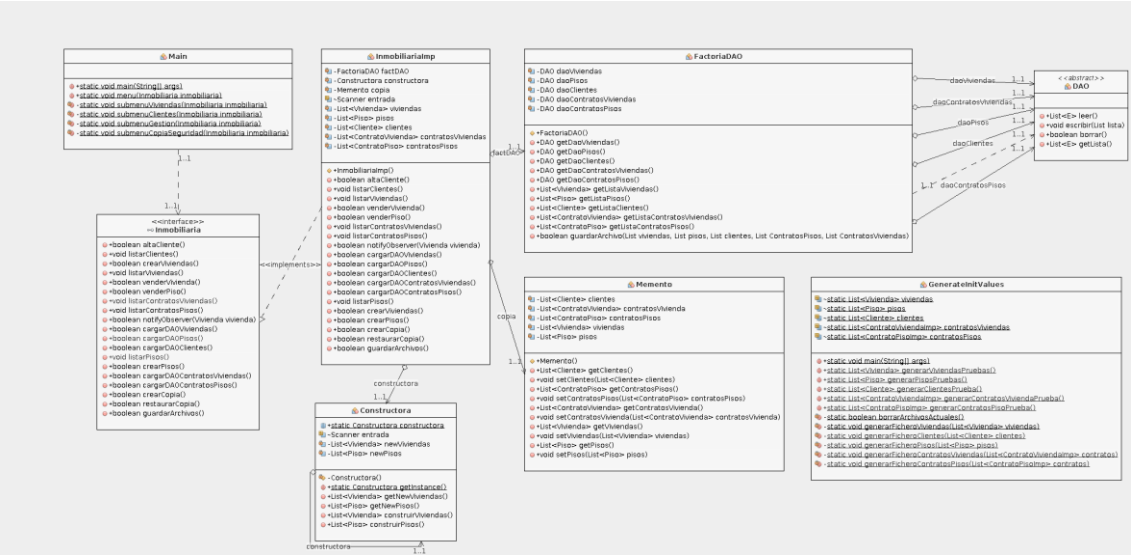
Clasificación: patrón de estructura.

Intención: dotar de funcionalidades dinámicamente a objetos mediante composición. Y así evitar jerarquías de clases complejas.

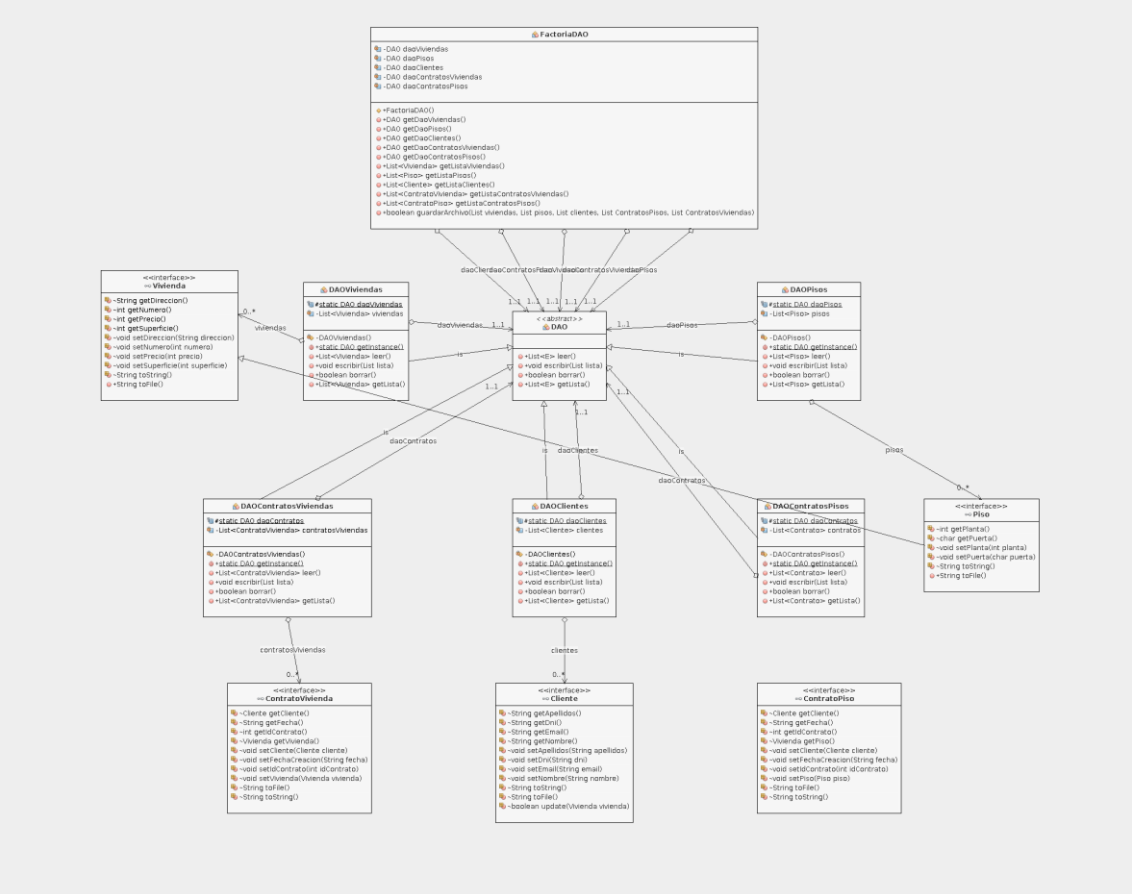
Utilidad en nuestro Proyecto: añadir si el inmueble es primera línea de playa o céntrico.

3. Diagrama de clases.

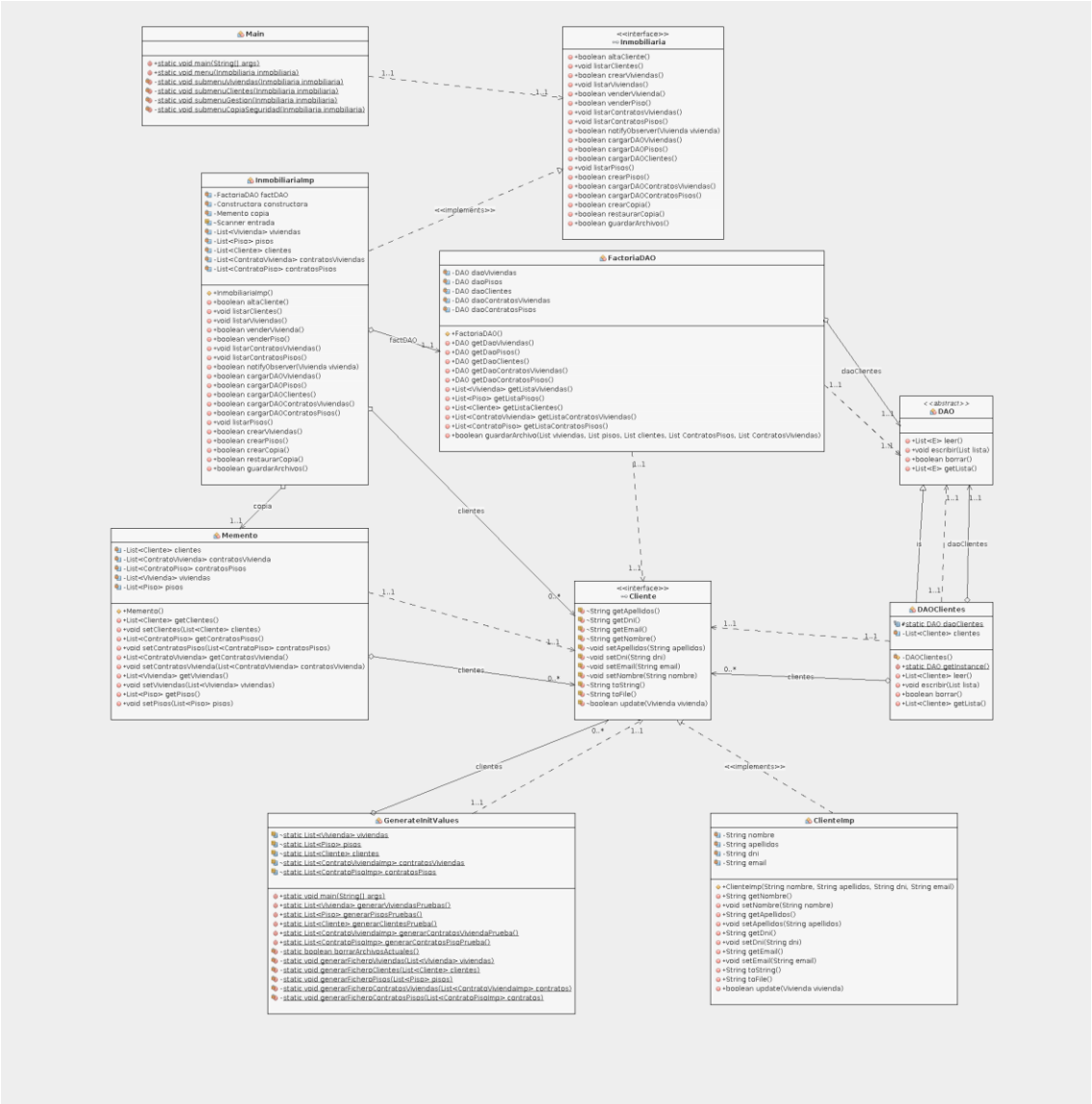
3.1 Vista General.



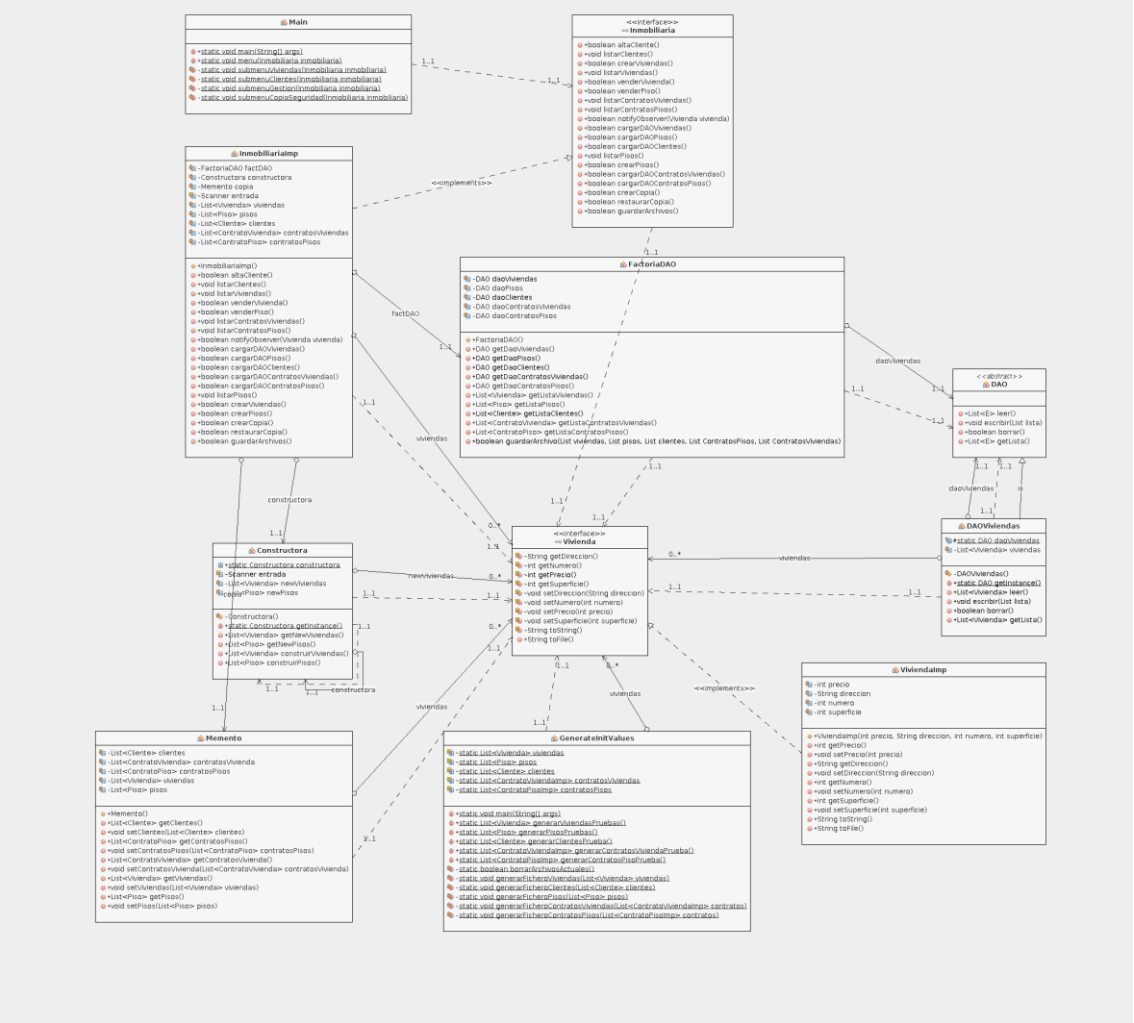
3.2 Factoría-DAO-Singleton.



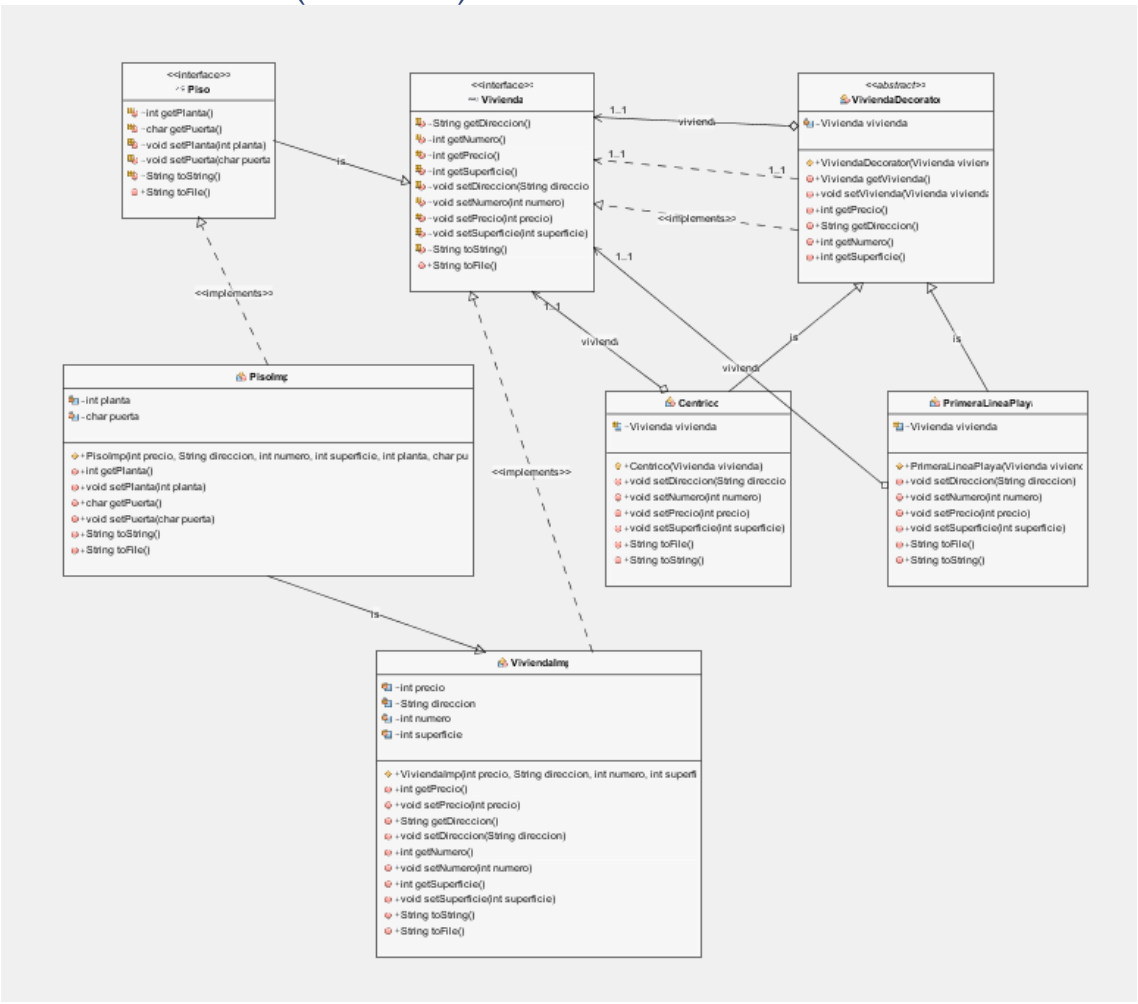
3.3 Vista Clientes.



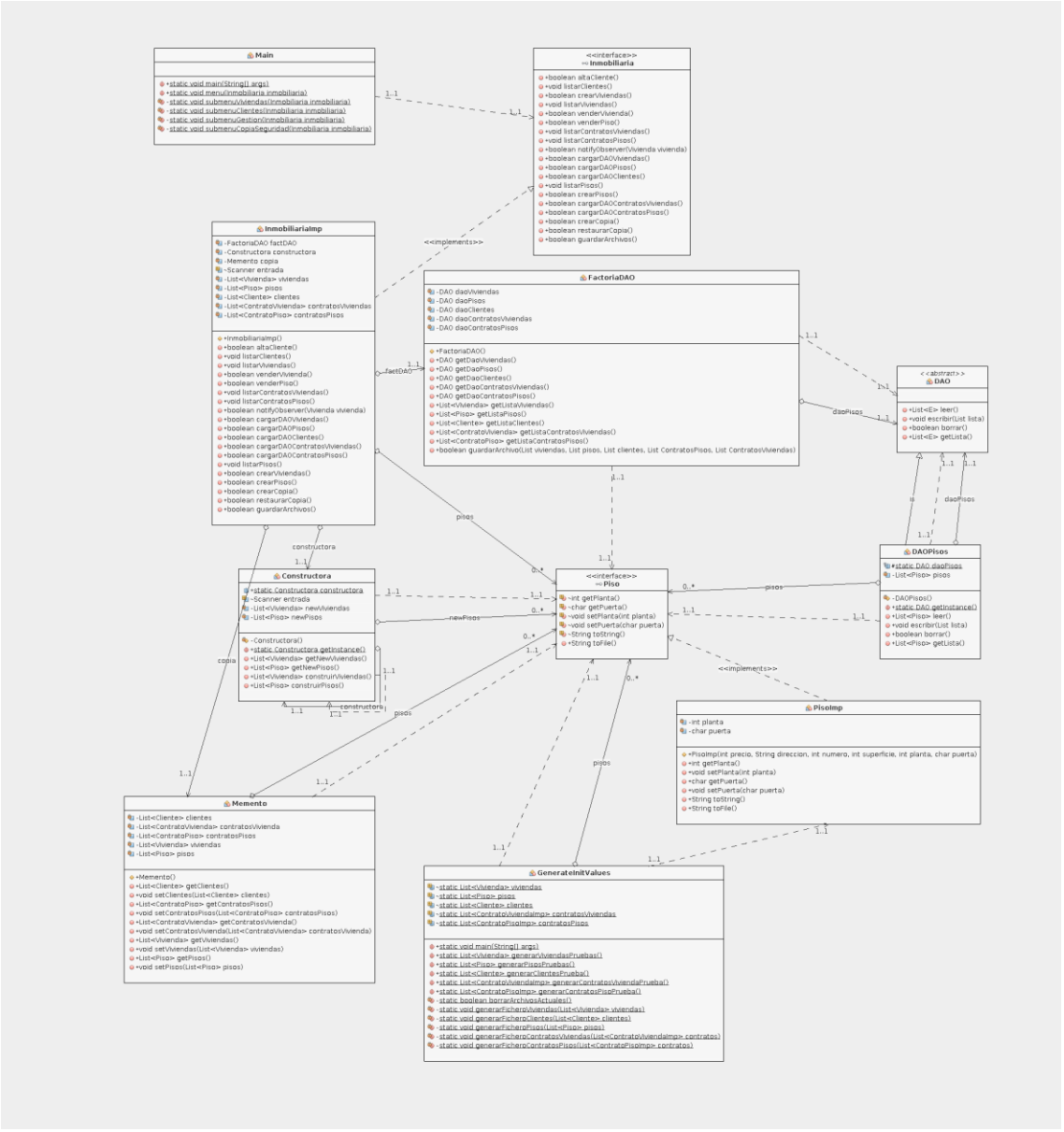
3.4 Vista Viviendas (general).



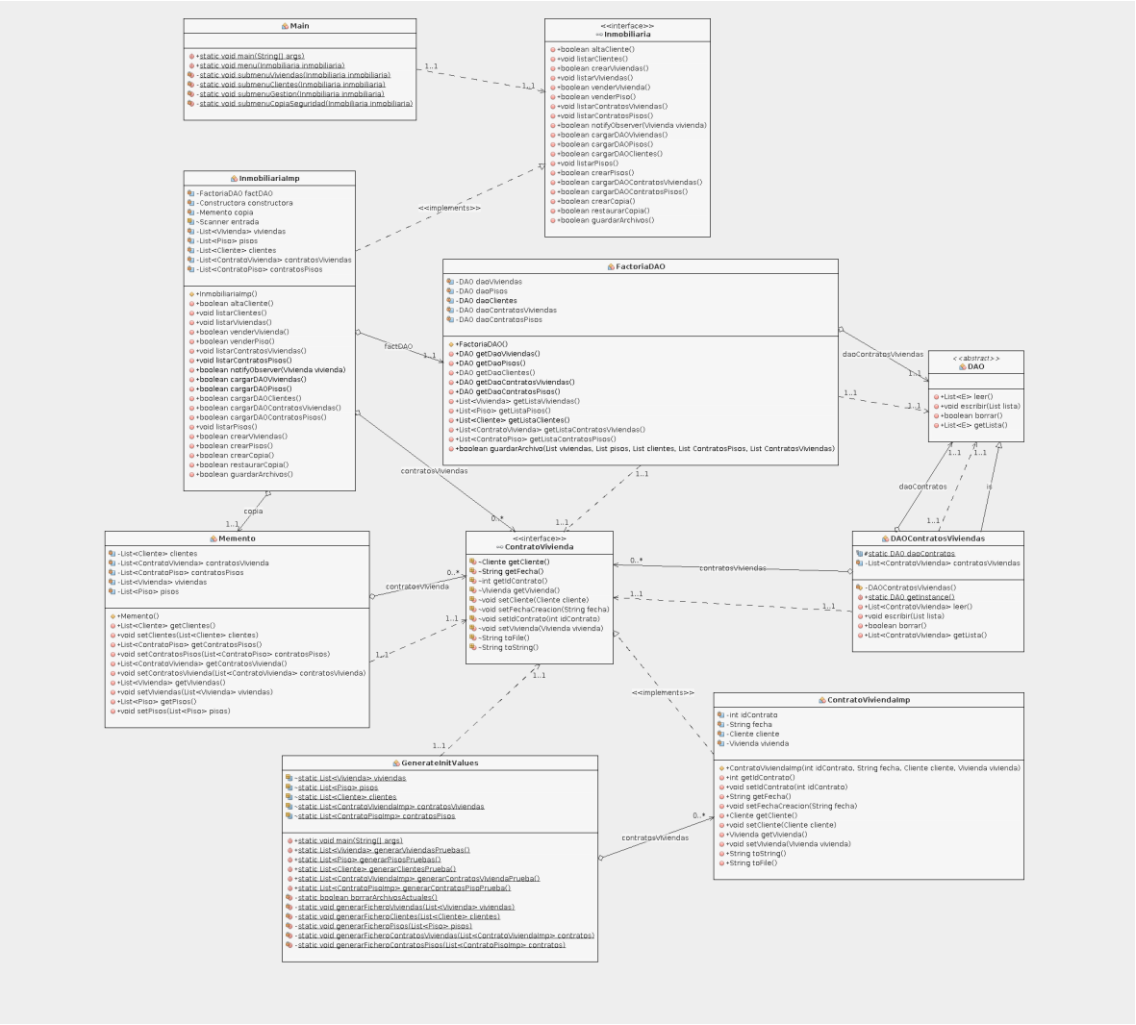
3.5 Vista Viviendas (decorator).



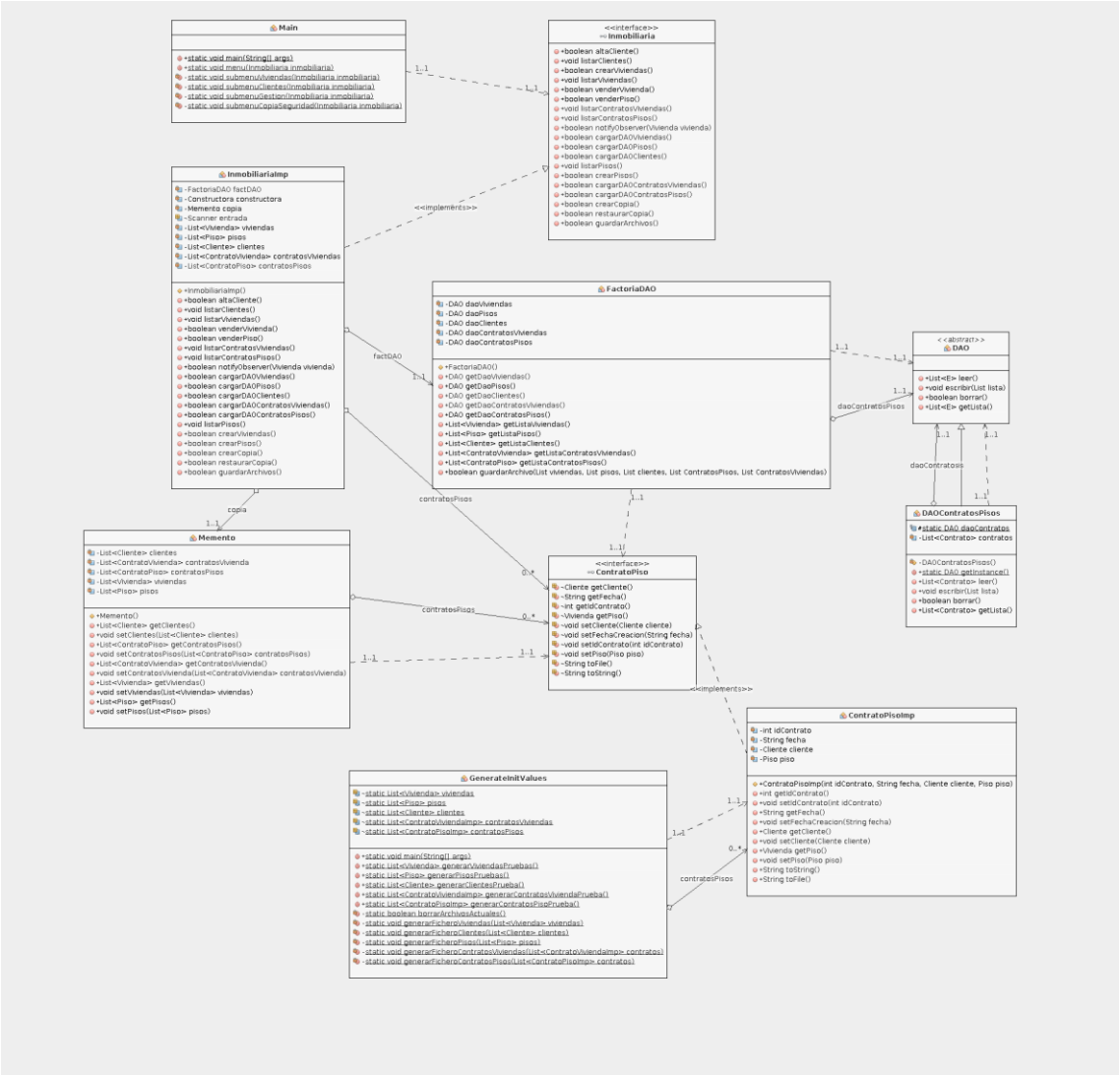
3.6 Vista Pisos.



3.7 Vista Contratos-Viviendas.



3.8 Vista Contratos-Pisos.



4. Bibliografía.

(1) Head First Design Patterns. Eric Freeman, Elisabeth Freeman. Editorial O'Reilly.

(2) Apuntes de clase.

(3) https://es.wikipedia.org

(4) https://es.stackoverflow.com

(5) http://minisconlatex.blogspot.com.es

(6) http://github.com

(7) http://jarroba.com

(8) http://codejavu.blogspot.com.es

(9) https://rootear.com/desarrollo/patron-decorator

5. Anotaciones.