Apache Commons Lang

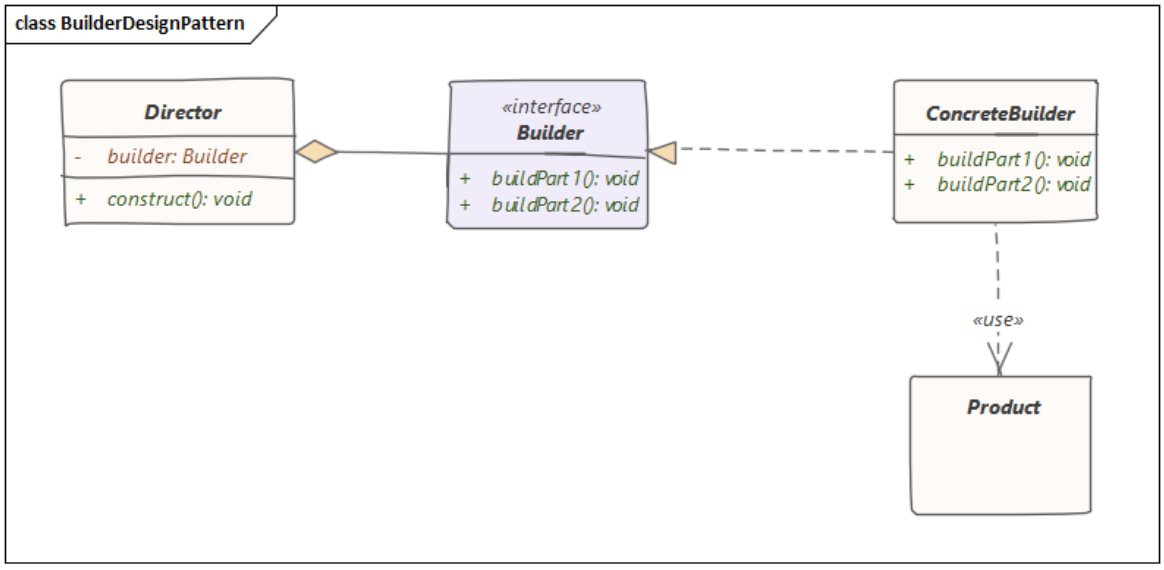
Las bibliotecas estándar de Java no proporcionan suficientes métodos para la manipulación de sus clases principales. Por esto, Apache Commons Lang ofrece estos métodos adicionales. Apache Commons Lang ofrece una variedad de utilidades auxiliares para la API java.lang, especialmente métodos de manipulación de cadenas, numéricos básicos, reflexión de objetos, concurrencia, creación y serialización, así como propiedades del sistema. Además, contiene mejoras básicas para java.util.Date y una serie de utilidades dedicadas a ayudar en la construcción de métodos, como hashCode, toString y equals.

Este proyecto cuenta con un diseño una responsabilidad dispersa. Esto se puede justificar en el hecho de que es una librería para extender los métodos de manipulación de las clases principales de Java. Dentro del proyecto se utilizan varios módulos para desarrollar diferentes métodos para diferentes clases. Por ejemplo, **“org.apache.commons.lang3.StringUtils”** y la clase **“org.apache.commons.lang3.ArrayUtils”,** los cuales son clases que separan las funcionalidades para que las responsabilidades sean adecuadas de cada clase. De igual manera, tiene buenas practicas dentro del proyecto por ejemplo los nombres son adecuados y eficientes para expresar las funcionalidades.

Gracias a la naturaleza del proyecto, al ser una librería, uno de sus mayores retos es hacer que la librería sea sencilla de entender pero que al mismo tiempo sea completa en sus funcionalidades, es decir, a pesar de que debe tener una complejidad alta, pues su objetivo del proyecto el cual es

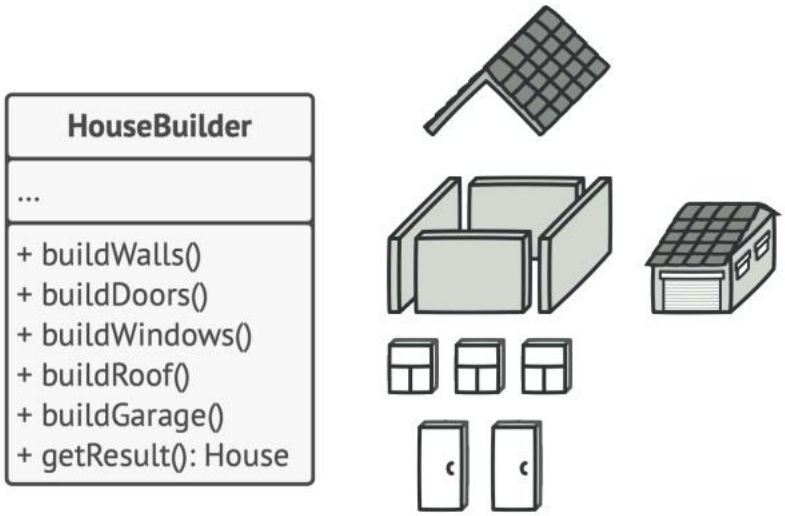
expandir la manipulación de las clases de Java es necesario que sea fácil de usar y comprender para que sea de utilidad para los usuarios.

Uno de los patrones que usa este proyecto es el patrón de creación Builder. El patrón de diseño Builder se define como una solución versátil para los retos de creación de objetos en la programación orientada a objetos. Hace posible la separación de la construcción de objetos complejos de su representación. Al emplear un enfoque paso a paso y utilizar objetos simples, el patrón Builder construye objetos intrincados de manera eficiente. Se trata de uno de los enfoques más eficaces para crear objetos complejos y está reconocido como uno de los patrones de diseño del Gang of Four, que ofrece soluciones para problemas de diseño comunes en el software orientado a objetos (Saxena.B, 2020). Este patrón destaca especialmente cuando se construyen varios objetos inmutables a través de un proceso de construcción unificado. En el siguiente diagrama se muestra como la interfaz Builder define



(Saxena.B, 2020)

pasos necesarios para que el objeto producto se ejecute correctamente, de esta manera también se podría diversificar el tipo de producto que se podría crear. Por ejemplo,



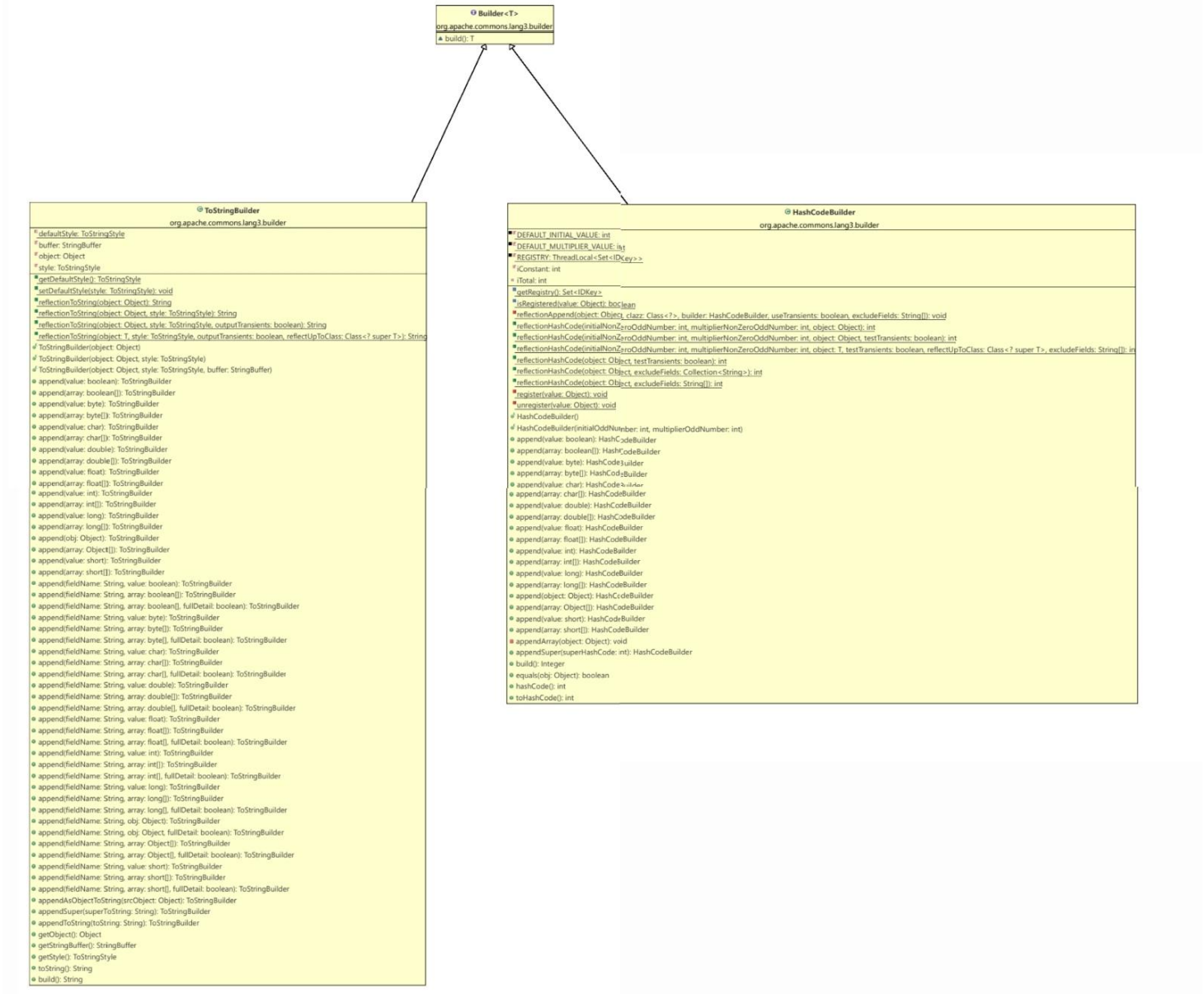
El patrón Builder te permite construir objetos complejos paso a paso. El patrón Builder no permite a otros objetos acceder al producto mientras se construye.

Shvets, A. 2022

Usualmente, este tipo de patrón se usa para crear diferentes objetos complejos desde una misma clase, de igual manera el patrón Builder ayuda a simplificar la construcción de objetos complejos.

Dentro del proyecto de Apache Commons Langs se puede evidenciar el patrón Builder en el módulo "**org.apache.commons.lang3.builder**" principalmente. En este módulo existen las clases como "**ToStringBuilder**" y "**HashCodeBuilder**" las cuales se encargan de facilitar la creación de objetos dentro de su propósito especifico, por ejemplo, representación de strings para

la clase “**ToStringBuilder**” y generación de Hash Code para “**HashCodeBuilder**”.



En el diagrama anterior se muestra las dos clases mencionadas y como se desarrollan las distintas funcionalidades para construir objetos tanto de String y Hash code. Por ejemplo, en la clase de String se implementa otra clase llama "**ToStringStyle**" que en el fondo solo cambia el tipo de fuente y apariencia de un string, pero de igual manera ayuda a crear un string más complejo entre diferentes tipos de configuraciones.

En este proyecto se puede observar el patrón Builder en su estructura completa. Esto se debe a que es una librería y este tipo de patrón ayuda a cumplir el reto de hacer la librería tanto entendible como compleja y funcional. Pues como se mencionó antes este patrón ayuda con la creación de objetos complejos desde un paso a paso para simplificar la complejidad del código. Estas implementaciones se ven en interfaces como la interfaz Builder y el módulo completo de Builder del proyecto mencionado.

Puntualmente, el patrón Builder se utiliza para crear objetos complejos. De igual manera, crear una cadena de métodos que a pesar de ser extensa es simple de comprender y logra el objetivo de simplificar el código para que pueda ser usable para los usuarios. Por otro lado, también permite la personalización de objetos dentro de las diferentes clases constructoras dentro del proyecto.

¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene? Debido a que las librerías deben ser simples pero funcionales y extensas en sus implementaciones, este patrón permite solucionar la complejidad del código para que se mas entendible separando en varias capas los pasos para construir los objetos de las clases de Java. La ventaja de este patrón es la simplificación del código y como puede ser globalizado para las diferentes clases existentes.

¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto? A pesar de ser un patrón poderoso para el desarrollo de librerías, también pude generar un sobre uso innecesario de memoria, ya que, al descomponer tanto la creación de un objeto puede producir que se implementen métodos de más, o que se sobre cargue el código afectando el rendimiento de este.

¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón? Otro patrón que se

podría implementar sería el Factory Pattern que encapsula la creación de objetos.

Este proyecto se encuentra publicado en el siguiente enlace: GitHub - apache/commons-lang: Apache Commons Lang. De igual manera, se puede acceder a la siguiente página web donde se obtiene la mayor información del proyecto: Lang – Home (apache.org)

Referencias:

- (Shvets, A. 2022) Recuperado del: “Sumérgete en los PATRONES DE DISEÑO”
- (Saxena.B, 2020) Recuperado del: Builder Design Pattern In Java - DZone