Для своей программы я решил использовать паттерн «Одиночка».

Этот паттерн я использовал для класса Validator, который служит для принятия и проверки ввода пользователя. Наличие нескольких объектов данного класса ни к чему, это лишь приведет к трате памяти компьютера. Особенно учитывая, что в этом объекте также создается и используется объект класса Scanner, который также необходим лишь в одном количестве.

Данный паттерн реализован следующим образом. Класс объявлен как final, чтобы нельзя было создать подклассы, сам он хранит статическую приватную переменную с собственным объектом. Конструктор также объявлен как private, чтобы не допустить создание новых объектов. Точкой входа в класс является статическая функция getValidator(), которая проверяет на null переменную validator, и только если переменная пуста, то он создает новый объект класса, иначе возвращает уже существующий объект.

UML диаграмма класса Validator представлена на рисунке 1.

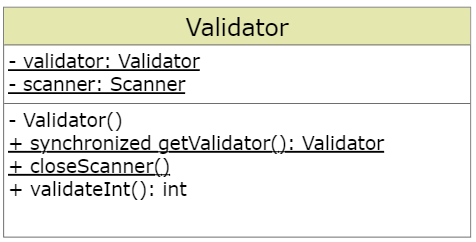


Рисунок 1 – UML диаграмма класса Validator

Паттерн, который точно не стоит применять в моей программе, это паттерн «Состояние». Это поведенческий паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояние. Во-первых, никаких состояний в моей программе не предусмотрено, во-вторых, объекты в моей программе всегда должны вести себя одним и тем же образом. Никакие изменения в поведении не допускаются.