Семестровая работа №1

По алгоритмам и структурам данных

Выполнил Файзуллин Ренат

Студент группы 11-906

Проверил:

Старший преподаватель кафедры

программной инженерии, к.ф.-м.н.

Хадиев К.Р.

Казань 2020

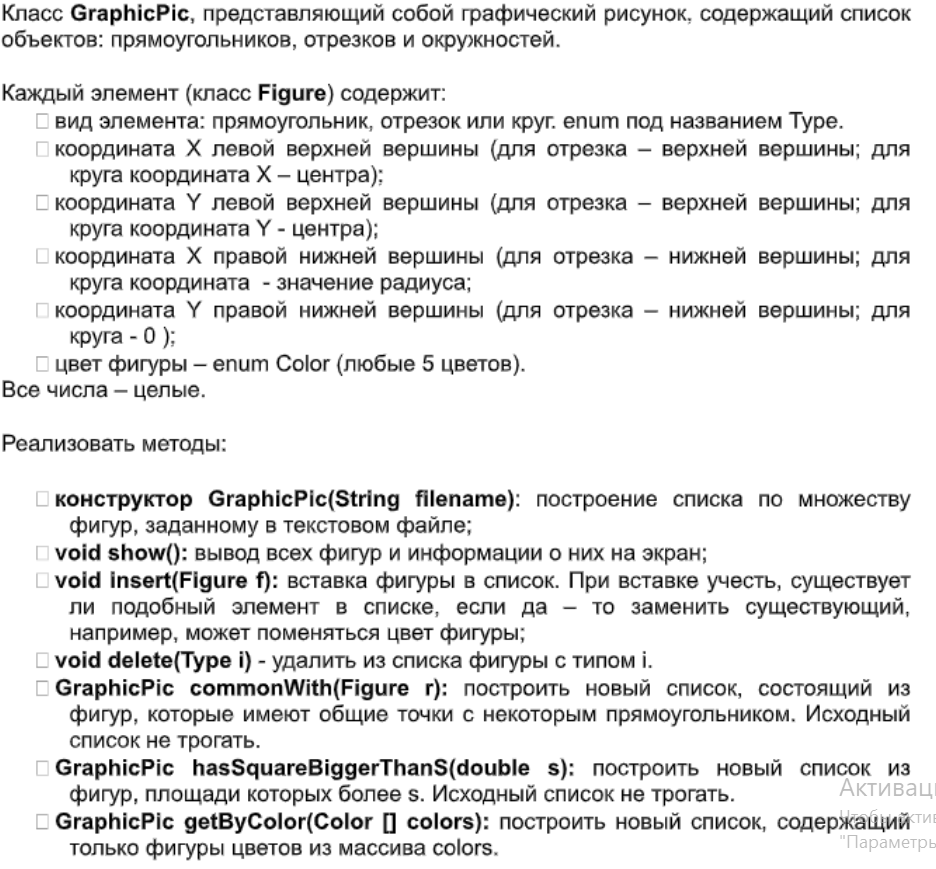
Содержание

Задача

Алгоритмы

Тесты

Задача



Алгоритмы

1. Конструктор GraphicPic(String filename)

При создании списка из множества фигур мы передаем конструктору строку с названием файла. Название строки мы передаем из Main класса. Далее в самом конструкторе мы задаем поля массива из фигур и количество этих элементов. С помощью классов FileReader и BufferedReader мы считываем каждую строку из файла и добавляем ее значения в элемент списка фигур путем метода add(Figure f).

Сложность конструктора O(n)

* Если увеличим количество входных данных на число n, то и количество операция увеличится на n раз. Для каждого элемента мы проходимся по конструктору ровно один раз.

1. Метод void show()

Данный метод просто выводит все элементы нашего списка на экран с помощью цикла for.

Пробегаемся по каждому элементу списка и выводим их

Сложность O(n)

* Если увеличим количество входных данных на число n, то и количество операция увеличится на n раз. Для каждого элемента мы проходимся по методу ровно один раз.

1. Метод insert(Figure f)

В данном методе мы сравниваем фигуру заданную в параметре метода с помощью другого метода equals(). Но для того чтобы поле с цветом нашей фигуры не сравнивалось, мы переопределяем метод equals в классе Figure.Когда находим такой же элемент то меняем значение цвета.

Сложность O(n)

* - Если увеличим количество входных данных на число n, то и количество операция увеличится на n раз. Для каждого элемента мы проходимся по методу ровно один раз.

4.Метод void delete(Type i)

В этом методе с помощью цикла for и прохождение по каждому элементу мы проверяем совпадение типа фигуры заданного в параметре метода и типа фигуры элемента списка. Если совпадение произошло, то этот элемент списка мы заменяем на последний. Стоит учесть, что при совпадении типов фигур нам необходимо уменьшить количество элементов списка на 1, и так же начать проверять список с данного элемента.

Сложность O(n)

* Если увеличим количество входных данных на число n, то и количество операция увеличится на n раз. Для каждого элемента мы проходимся по методу ровно один раз.

5.Метод hasSquareBiggerThanS(double s)

Для начала нам нужно высчитать площадь каждой из фигур.

Для этого в классе Figure мы создаем метод getSquare(), с помощью которого находим тип фигуры и вычисляем для нее площадь.

В самом методе hasSquare мы создаем новый список фигур с пустым конструктором. Пробегая по-нашему уже существующему списку фигур, мы находим те фигуры, площадь которых больше заданного числа s и добавляем их в наш список новый список путем метода add (Figure f).

Сложность O(n)

* Если увеличим количество входных данных на число n, то и количество операция увеличится на n раз. Для каждого элемента мы проходимся по методу ровно один раз.

6. Метод getByColor(Color [] color)

Как в предыдущем методе мы создаем новый список фигур с пустым конструктором. Для каждого элемента из списка фигур мы должны проверить каждый цвет из массива цветов. Задаем двойной цикл и при совпадении цветов добавляем элемент уже существующего списка фигур в новый список.

Сложность O(n^2)

- В цикле у нас есть внутренний цикл. Для каждого элемента их списка нужно проверить каждый цвет, из этого выходит сложность O(n^2)