**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

по лабораторной работе № 2

по дисциплине «Алгоритмы и структуры данных»

Тема: **«** ПОДДЕРЖКА ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ**»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1308 |  | Макаров М.В. |
| Студент гр. 1308 |  | Томилов Д. |  | Макаров М.В. |
| Преподаватель |  | Манирагена В. |  | Морозов С.М. |

Санкт-Петербург

2022

**Цель работы.**

Ознакомление с механизмами обработки исключительных ситуаций, представленными в языке С++. Получение практических навыков обработки исключительных ситуаций.

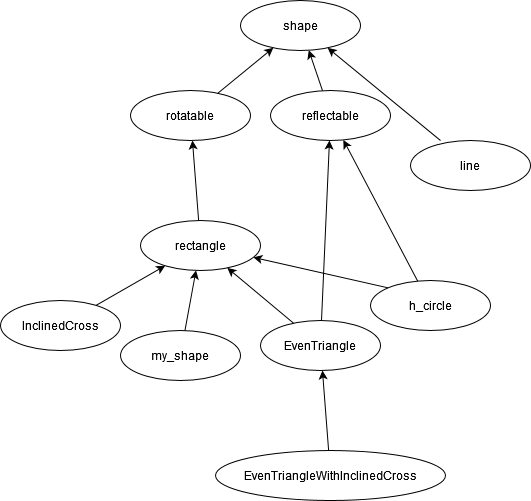
**Задание**

Переработать программу работы с библиотекой фигур, дополнив ее механизмом контроля исключительных ситуаций.

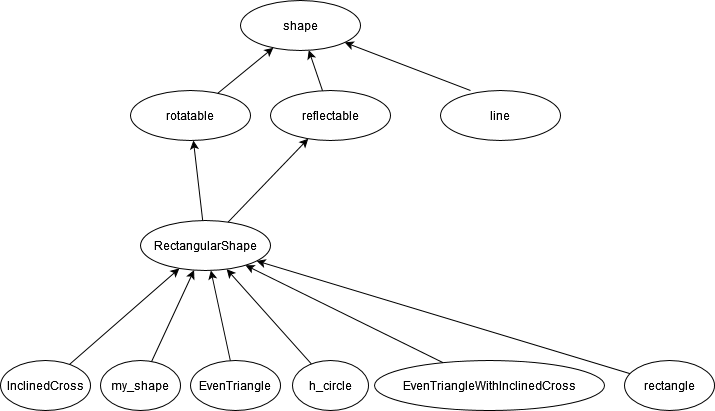
**Описание решения задачи**

Для начала было решено перестроить систему классов, представленную в работе, поскольку, как реализовывать генерацию и перехват ошибок в предложенной нам структуре, было решительно непонятно.

Ранее структура классов выглядела так:



Теперь она выглядит вот так:



Идея данного изменения состоит в том, чтобы, во-первых, создать абстрактный класс (RectangularShape), в котором будут происходить все манипуляции с геометрией объектов и, во-вторых, проверять, нужно ли сгенерировать исключение, именно в этом классе.

Будем обрабатывать 2 типа ошибок: ошибки при инициализации объекта и ошибки при передвижении объекта.

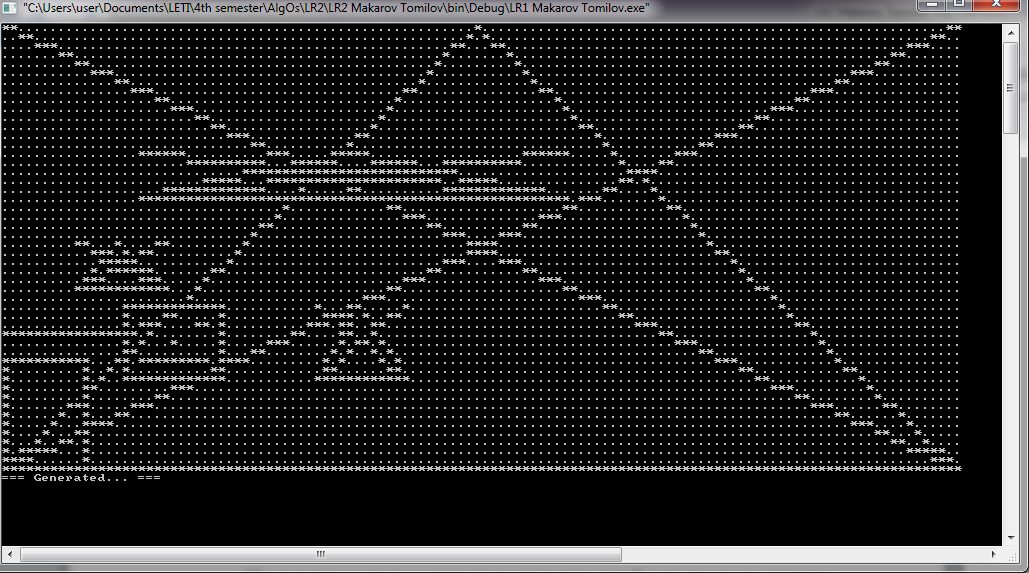
При ошибке инициализации объекта просто изменяем размеры объекта таким образом, чтобы он влезал на экран. Это делается в классе RectangularShape. Из-за этого все классы, производные от него, также изменяют свои размеры.

Далее рассмотрим то, как обрабатываются ошибки при перемещении, повороте и изменении размера объекта. Это делается по схожей схеме: сначала в классе RectangularShape вычисляется, будут ли границы объекта находиться за границами экрана. Если нет, то внешняя геометрия объекта соответствующим образом преобразуется и исполнение программы передается дальше классам, производным от RectangularShape. Они, в свою очередь, меняют какие-то свои параметры. Если же объект частично скрыт границами экрана, то внешняя геометрия объекта не изменяется и генерируется соответствующее исключение (CantMove/ CantRotate/ CantResize). Это исключение считывается производным классом в соответствующих методах, и производный класс понимает, что никаких действий с геометрией совершать не нужно.

**Примеры выполнения программы**

**Пример 1**

В этом примере создадим треугольник с крестом, превышающий по размеру экран.

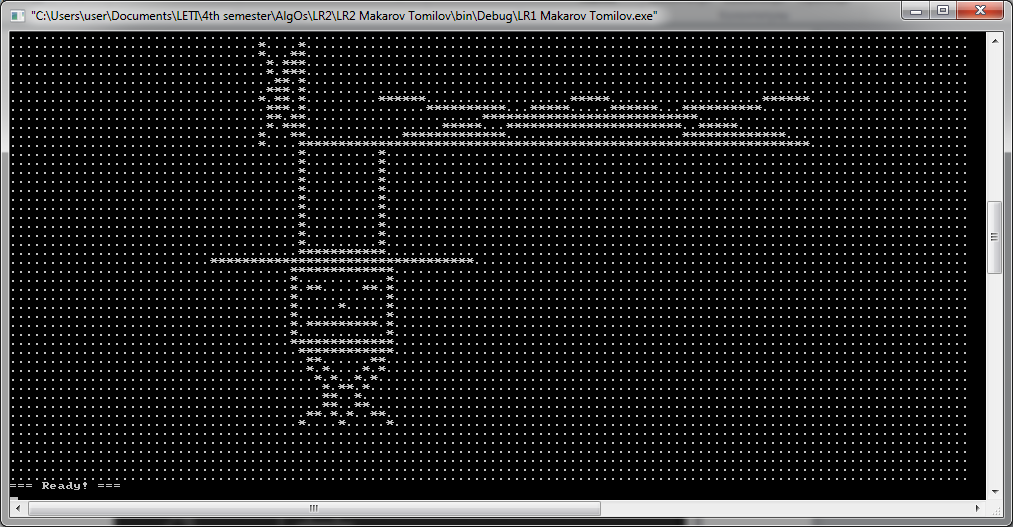


Как видно, размер был автоматически подогнан при генерации объекта, чтобы помещаться на экран.

**Пример 2**

В этом примере сделаем так, чтобы правый рог на шляпе был слишком большим, чтобы при его повороте он бы вылетал за границы экрана. То же самое в этом примере происходит и со шляпой (прямоугольник слева внизу).



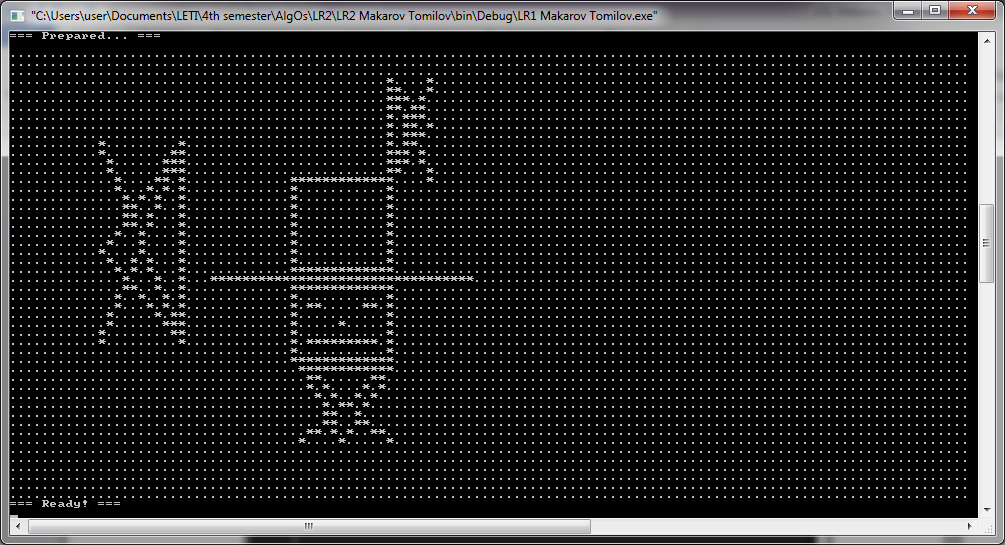


Как мы видим, на стадии подготовки к сборке ни рог, ни шляпа не были повёрнуты, хоть в коде это прописано.

**Пример 3**

В этом примере после поворотов рогов попытаемся увеличить правый рог в 3 раза, а левый в 2 раза.





Как мы видим, левый рог увеличился в 2 раза, поскольку ему хватало на это места, а правый не увеличился, поскольку тогда произошёл бы выход за границы экрана.

Однако, при сборке фигуры только правый рог встал на своё место, поскольку только для него хватило места при сдвиге. Левому же рогу места не хватило.

**Выводы.**

Был получен опыт в использовании механизма обработки исключительных ситуаций. Была построена система обработки исключительных ситуаций, которую довольно легко можно было бы масштабировать на большее количество фигур.