МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Лабораторная работа № 5**

по курсу «Объектно-ориентированное программирование»

**«Классы и объекты в C++»**

**8 ВАРИАНТ**

Выполнили:

студенты гр. КТбо2-1

Яшенков А.В.

Проверил:

Тарасов С. А.

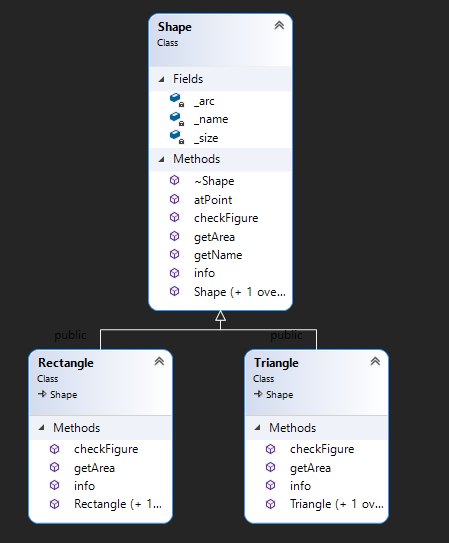
**Таганрог 2020**

1. **Вариант задания**

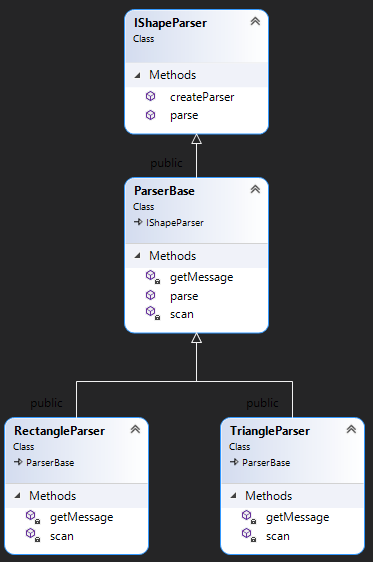
Класс Shape с потомка Triangle, Rectangle. Фабричное создание объектов типа Shape осуществляется с помощью класса IShapeParser.

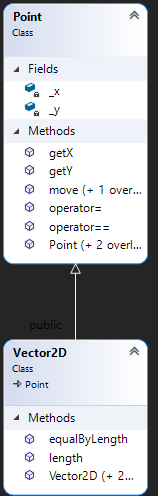
Класс ShapeStorage хранит

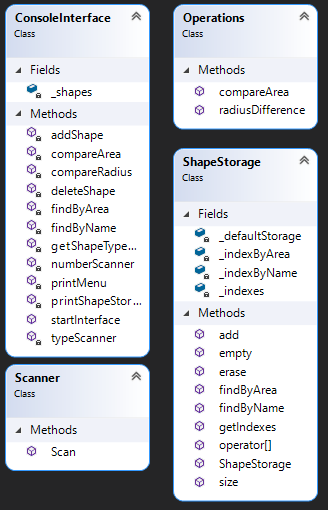
1. **UML-диаграмма наследования классов Triangle и Rectangle от класса Shape.**



**3. UML-диаграммы остальных классов.**







**4. Листинг**

**ConsoleInterface.h**

**#pragma once**

**#include "IShapeParser.h"**

**#include "ShapeStorage.h"**

**#include <vector>**

**class ConsoleInterface**

**{**

**public:**

**void startInterface();**

**private:**

**const char\* getShapeTypeMessage() const;**

**void printMenu() const;**

**char typeScanner() const;**

**int numberScanner() const;**

**void addShape();**

**void deleteShape();**

**void printShapeStorage();**

**void compareArea();**

**void compareRadius();**

**void findByName();**

**void findByArea();**

**ShapeStorage \_shapes;**

**};**

**IShapeParser.h**

**#pragma once**

**#include "Shape.h"**

**class IShapeParser**

**{**

**public:**

**virtual std::unique\_ptr<Shape> parse() const = 0;**

**static std::unique\_ptr<IShapeParser> createParser(char str);**

**};**

**Operations.h**

**#pragma once**

**#include "Shape.h"**

**class Operations**

**{**

**public:**

**double compareArea(const Shape& obj\_1, const Shape& obj\_2);**

**double radiusDifference(const Shape& obj\_1, const Shape& obj\_2);**

**};**

**ParserBase.h**

**#pragma once**

**#include "IShapeParser.h"**

**class ParserBase : public IShapeParser**

**{**

**public:**

**std::unique\_ptr<Shape> parse() const override;**

**private:**

**virtual const char\* getMessage() const = 0;**

**virtual std::unique\_ptr<Shape> scan() const = 0;**

**};**

**Point.h**

**#pragma once**

**#include <iostream>**

**class Point**

**{**

**public:**

**Point() = default;**

**Point(double x, double y);**

**Point(const Point& other);**

**Point& operator=(const Point& other);**

**bool operator==(const Point& other) const;**

**double getX() const;**

**double getY() const;**

**void move(const Point& other);**

**void move(double x, double y);**

**friend std::ostream& operator<<(std::ostream& out, const Point& point);**

**private:**

**double \_x = 0;**

**double \_y = 0;**

**};**

**Rectangle.h**

**#pragma once**

**#include "Shape.h"**

**class Rectangle : public Shape**

**{**

**public:**

**Rectangle() = default;**

**Rectangle(const std::string& name, std::vector<Point>& arrayPoints);**

**const char\* info() const override;**

**double getArea() const override;**

**void checkFigure() const override;**

**};**

**RectangleParser.h**

**#pragma once**

**#include "ParserBase.h"**

**class RectangleParser : public ParserBase**

**{**

**private:**

**const char\* getMessage() const override;**

**std::unique\_ptr<Shape> scan() const override;**

**};**

**Scanner.h**

**#pragma once**

**#include <iostream>**

**class Scanner**

**{**

**public:**

**int Scan(std::istream& is, const char\*, ...);**

**};**

**Shape.h**

**#pragma once**

**#include "Point.h"**

**#include <vector>**

**class Shape**

**{**

**public:**

**Shape() = default;**

**Shape(const std::string& name, std::vector<Point>& arrayPoints);**

**virtual ~Shape() = default;**

**virtual const char\* info() const = 0;**

**virtual double getArea() const = 0;**

**virtual void checkFigure() const = 0;**

**const Point& atPoint(int index) const;**

**const std::string& getName() const;**

**private:**

**std::string \_name = nullptr;**

**int \_size = 0;**

**std::vector<Point> \_arc;**

**};**

**ShapeStorage.h**

**#pragma once**

**#include "Shape.h"**

**#include "IShapeParser.h"**

**#include <memory>**

**#include <map>**

**#include <vector>**

**#include <set>**

**class ShapeStorage**

**{**

**public:**

**ShapeStorage() = default;**

**void add(std::unique\_ptr<Shape>&& source, int index);**

**void erase(int index);**

**std::set<int> findByArea(double area);**

**std::set<int> findByName(const std::string& name);**

**const Shape& operator[](int index);**

**bool empty() const;**

**int size() const;**

**const std::set<int>& getIndexes() const;**

**private:**

**std::map<int, std::unique\_ptr<Shape>> \_defaultStorage;**

**std::map<double, std::set<int>> \_indexByArea;**

**std::map<std::string, std::set<int>> \_indexByName;**

**std::set<int> \_indexes;**

**};**

**Triangle.h**

**#pragma once**

**#include "Shape.h"**

**class Triangle : public Shape**

**{**

**public:**

**Triangle() = default;**

**Triangle(const std::string& name, std::vector<Point>& arrayPoints);**

**const char\* info() const override;**

**double getArea() const override;**

**void checkFigure() const override;**

**};**

**TriangleParser.h**

**#pragma once**

**#include "ParserBase.h"**

**class TriangleParser : public ParserBase**

**{**

**private:**

**const char\* getMessage() const override;**

**std::unique\_ptr<Shape> scan() const override;**

**};**

**Vector2D.h**

**#pragma once**

**#include "Point.h"**

**class Vector2D : public Point**

**{**

**public:**

**Vector2D() = default;**

**Vector2D(double x, double y);**

**Vector2D(const Vector2D& other);**

**bool equalByLength(const Vector2D& other) const;**

**double length() const;**

**};**