Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПРОЕКТНАЯ ДОКУМЕНТАЦИЯ

Лабораторная работа № 6
по дисциплине «Новые технологии в программировании»
Вариант № 1

Студент гр. 584-2
_____ Огородов А. С.
_____

Руководитель
к.т.н., доц. каф. КСУП
_____ _____ Горяинов А. Е.
_____

Томск 2017

Оглавление

1 Введение

Ведение проектной документации является неотъемлемой частью разработки любого хоть сколько-нибудь сложного и востребованного программного продукта.

Основным проектным документом является техническое задание (ТЗ) — документ, описывающий требования заказчика к разрабатываемому проекту. Главное назначение ТЗ — обеспечение единого понимания всех аспектов проекта у разработчика и заказчика, что позволяет снизить число конфликтов и взаимных недопониманий в процессе разработки. В конечном счёте, грамотно составленное ТЗ сокращает сроки и расходы на реализацию проекта.

Помимо ТЗ, в состав проектной документации входят и другие документы, уточняющие или дополняющие ТЗ, например, описание архитектуры разрабатываемой программы в виде UML-диаграмм.

Данный отчёт содержит в себе ТЗ и описание программы, разработанной по нему. На рисунке 1.1 изображён графический интерфейс полученной программы.

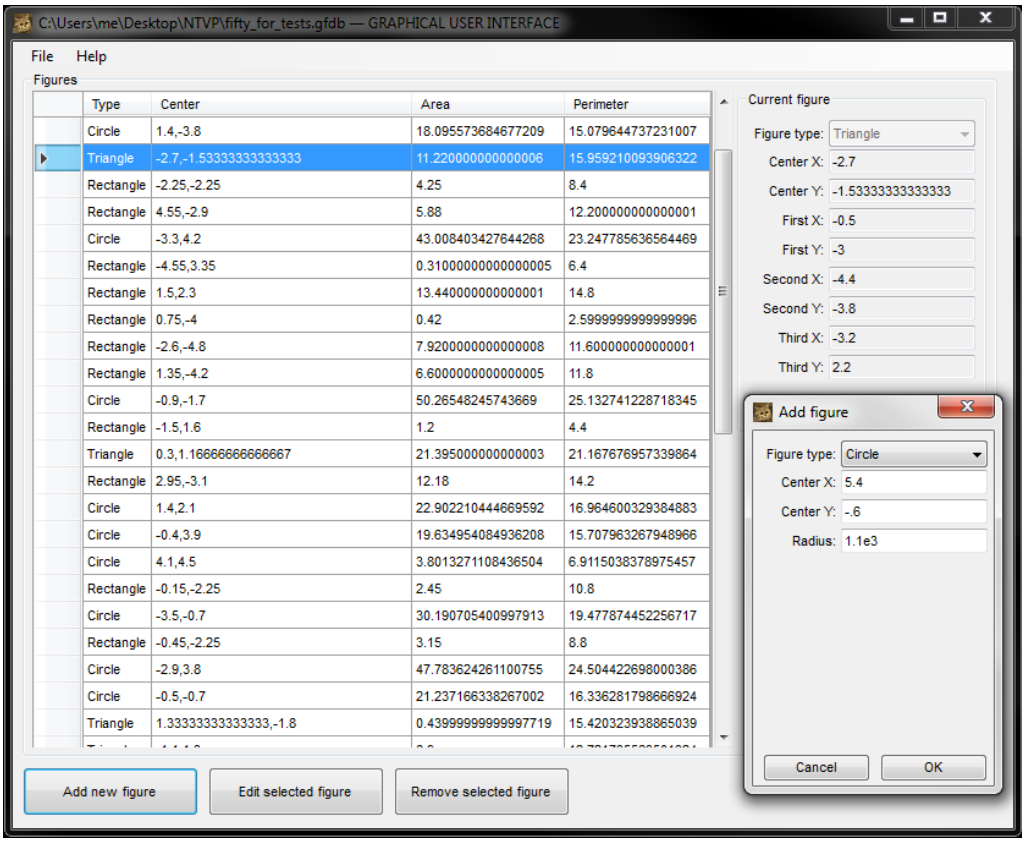

Рисунок 1.1 – Законченная программа

2 Техническое задание

В данном разделе приведено ТЗ на программу «GUI DB for Model», разработанную в ходе лабораторных работ курса «Новые технологии в программировании». Данная программа представляет собой примитивный редактор баз данных (БД) двумерных геометрических фигур.

2.1 Исходная проблема

Не существует простых бесплатных программ, позволяющих вести учёт двумерных геометрических фигур для упрощения решения геометрических задач.

2.2 Цель

Цель данного проекта — разработать программу, реализующую возможность работы с БД плоских геометрических фигур с помощью графического интерфейса пользователя.

2.3 Задачи

Для выполнения поставленной цели необходимо решить следующие задачи:

– программа должна предоставлять возможность работы БД с поддержкой трёх типов геометрических фигур: кругов, прямоугольников и треугольников; предельно допустимое количество записей о фигурах в одной БД — 50;

– программа должна предоставлять возможность хранения БД геометрических фигур в файлах;

– программа должна иметь графический интерфейс.

2.4 Конечный пользователь и контекст использования

Конечными пользователями программы являются школьники, которые могут использовать разрабатываемую программу в качестве средства компьютерного моделирования двумерных геометрических задач и средства расчёта периметров и площадей моделируемых фигур. Так как для решения геометрических задач необходимо обладать знаниями геометрии, данная программа является узкоспециализированной.

Критерии качества, описанные далее в п. 2.5 — небольшое потребление памяти и возможность работы на самой распространённой операционной системе (ОС) [1] — отражают то, что целевая аудитория располагает только базовыми вычислительными средствами.

2.5 Критерии качества

Разработанная программа должна отвечать следующим критериям качества:

– должна присутствовать проверка вводимых пользователем данных на корректность;

– программа должна работать на ОС Microsoft Windows 7;

– количество потребляемой программой оперативной памяти при работе с БД, содержащей максимальное количество записей, не должно превышать 50 мегабайт;

– время выполнения операций над БД, содержащей максимальное количество записей, не должно превышать 3 секунд.

2.6 Дополнительные требования

К разрабатываемой программе также предъявляются следующие требования:

– разработка должна вестись на языке C#, используемая версия .NET Framework — 4.5.2;

– исходные тексты программы должны храниться с использованием системы контроля версий git;

– логика программы должна быть подвергнута модульному тестированию;

– программа должна распространяться в виде установочного файла.

2.7 Функциональные возможности

Функциональные возможности программы можно разделить на два высокоуровневых блока — собственно работа с БД и операции над файлами, хранящими в себе БД. Данные блоки можно рассматривать отдельно друг от друга: при реализации работы с файлами БД знания о структуре БД используются только при сериализации или десериализации отдельно взятой записи. Аналогично, реализация работы с записями не затрагивает работу с файлами.

2.7.1 Работа с БД

Записи в БД — данные о геометрических фигурах трёх различных типов: круг, прямоугольник, треугольник. Для всех типов фигур общими являются свойства:

– строка — тип фигуры на английском языке;
– координаты центра фигуры на плоскости xOy;
– площадь фигуры, вычисляется автоматически по параметрам фигуры;
– периметр фигуры, вычисляется автоматически по параметрам фигуры.

У каждого типа фигур также имеются свои уникальные свойства:

– у круга: неотрицательный радиус;
– у прямоугольника: неотрицательные высота и ширина;
– у треугольника: координаты трёх его точек на плоскости xOy.

Все значения свойств, не вычисляющихся автоматически, а задаваемых пользователем, не превышают 10^9 по модулю.

На главном окне программы присутствует таблица, отображающая фигуры в БД со свойствами, общими для всех типов. Отдельную фигуру в таблице можно выделить с помощью клика мышью, выделение можно перемещать клавишами со стрелками вверх и вниз.

Если какая-либо фигура выделена, то её параметры, специфичные для типа фигуры, отображаются в отдельном элементе главного окна, а кнопки «Remove selected figure» и «Edit selected figure» становятся активными. По нажатию кнопки «Remove selected figure» выбранная фигура удаляется из таблицы. После нажатия кнопки «Edit selected figure» появляется дополнительное окно редактирования, содержащее полную информацию о фигуре и кнопки «Cancel» и «OK». Данные о фигуре (в том числе её тип, при смене которого поля ввода заменяются на требуемые для выбранного типа) можно редактировать, при вводе некорректных данных в поля ввода текст в них становится полужирным, а кнопка «OK» блокируется до исправления данных. Если окно было закрыто или была нажата кнопка «Cancel», изменения фигуры в таблице на главном окне не происходит. Если же была нажата кнопка «OK», то данные выделенной фигуры в таблицы заменяются данными, введёнными в окне редактирования.

На главном окне также присутствует кнопка «Add new figure». По нажатию на неё появляется окно с данными о фигуре, аналогичное окну редактирования фигуры. Если данные были введены корректно и была нажата кнопка «OK», в конец таблицы добавляется новая фигура с данными из формы, в любом другом случае добавления не происходит.

2.7.2 Работа с файлами БД

Любое изменение в таблице, представляющей БД, приводит к изменению её статуса на «есть несохранённые изменения», что обозначается символом «*» после имени файла, отображаемого в заголовке главного окна программы. При закрытии окна с БД, содержащей несохранённые изменения, предлагается сохранить их аналогично описываемому далее механизму сохранения файлов БД.

БД, отображаемую в таблице, можно сохранить в файл с заданным пользователем именем. В случае, если имя еще не было задано, при сохранении появляется диалоговое окно, запрашивающее у пользователя имя файла, в который будет произведено сохранение БД. После сохранения статус «есть несохранённые изменения» сбрасывается. Пункт «Save as» главного меню программы задаёт имя для файла независимо от того, было ли оно задано раньше.

Файл с БД, сохранённый ранее, можно загрузить в программу через диалоговое окно выбора файла для открытия. Если в текущем файле есть несохранённые изменения, то перед открытием нового файла пользователю предлагается сохранить текущий файл. Если новый файл был выбран, то таблица в главном окне программы начинает отображать данные из нового файла.

Должна иметься возможность создания новой пустой БД, которая замещает открытую в программе в данный момент. Если в текущей БД есть несохранённые изменения, то перед заменой её предлагается сохранить в файл.

2.8 План разработки

Разработку программы можно разделить на этапы, соответствующие лабораторным работам курса:

– разработка бизнес-логики: библиотеки для работы с геометрическими фигурами;

– разработка графического интерфейса и средств работы с БД;

– создание модульных тестов для бизнес-логики;

– наращивание функциональности графического интерфейса, рефакторинг исходного кода и сборка установочного файла;

– оформление документации к программе.

Время, отведённое на выполнение каждого из этапов — две недели. Всего на разработку программы отведено два с половиной месяца.

3 Описание программной системы

В данном разделе приведено краткое описание проекта программы, отвечающей требованиям поставленного ТЗ, и процесса её создания.

3.1 Описание вариантов использования

Диаграмма вариантов использования (use case diagram), являющаяся частью стандарта UML [2], служит для описания взаимодействия системы с действующими лицами [3]. Варианты использования специфицируют ожидаемое поведение системы (её части), описывая последовательности действий, включая их варианты, которые осуществляются системой для достижения действующим лицом определенного результата. При этом конкретная реализация функций вариантов использования не специфицируется. Диаграмма вариантов использования программы приведена на рисунке 3.1.

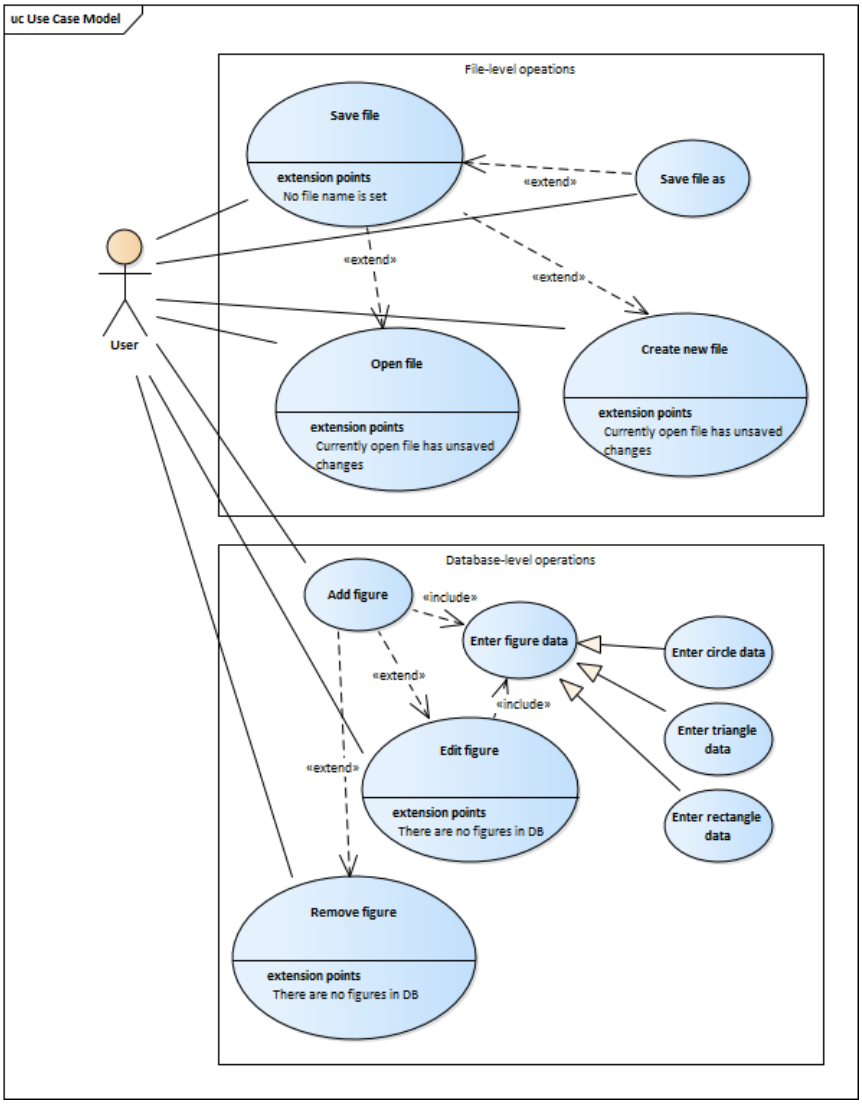

Рисунок 3.1 – Диаграмма вариантов использования программы

3.2 Описание классов программы

Разработка программы велась с использованием объектно-ориентированного программирования (ООП) — подхода к программированию, основной целью которого является повторное использование существующего программного обеспечения [4] (в частности, весь графический интерфейс программы состоит из стандартных переиспользуемых компонентов). Средством реализации основной цели ООП являются абстрактные типы данных с поддержкой наследования, называемые классами.

Диаграмма классов программы служит для описания классов, входящих в систему, их статической структуры и типы взаимосвязей между классами. Множество элементов диаграммы в совокупности отражает декларативные знания о структуре системы. Высокоуровневая UML-диаграмма, отображающая связи между всеми классами программы и принадлежность их к пакетам, без указания полных списков полей и методов классов, приведена на рисунке 3.2.

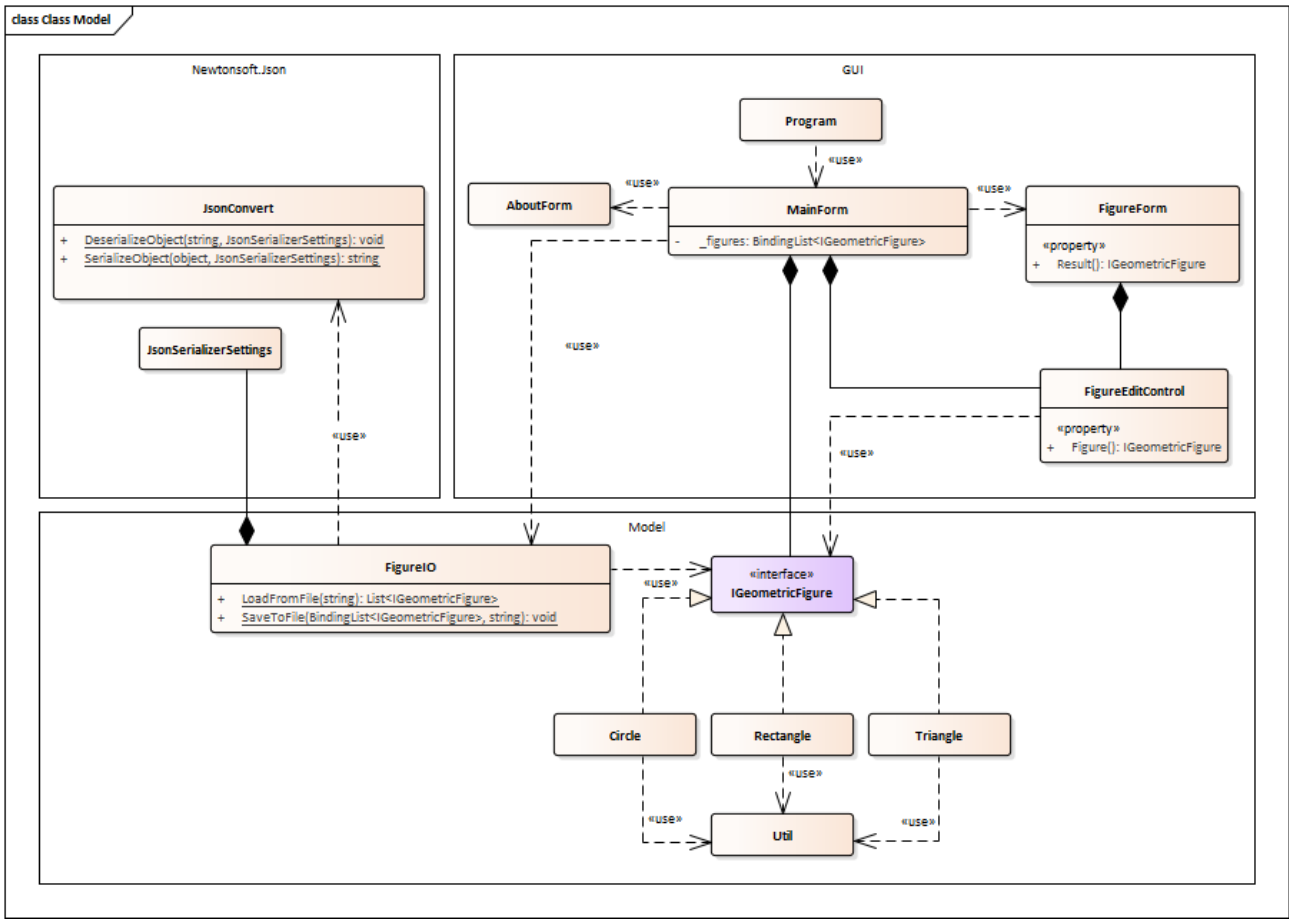

Рисунок 3.2 – Классы программы

Из рисунка 3.2 видно, что программа состоит из трёх пакетов. Пакет GUI содержит в себе классы, отвечающие за графический интерфейс программы, такие как формы программы, а также класс Program, являющийся точкой входа программы. Пакет

Newtonsoft.Json состоит из классов одноимённой библиотеки, используемых в программе для сохранения и загрузки файлов БД. Пакет Model содержит в себе логику работы с геометрическими фигурами, поэтому рассматривается подробнее. На рисунке 3.3 приведена UML-диаграмма уровня реализации данного пакета, то есть с указанием всех имеющихся полей и методов классов в дополнение к их взаимоотношениям.

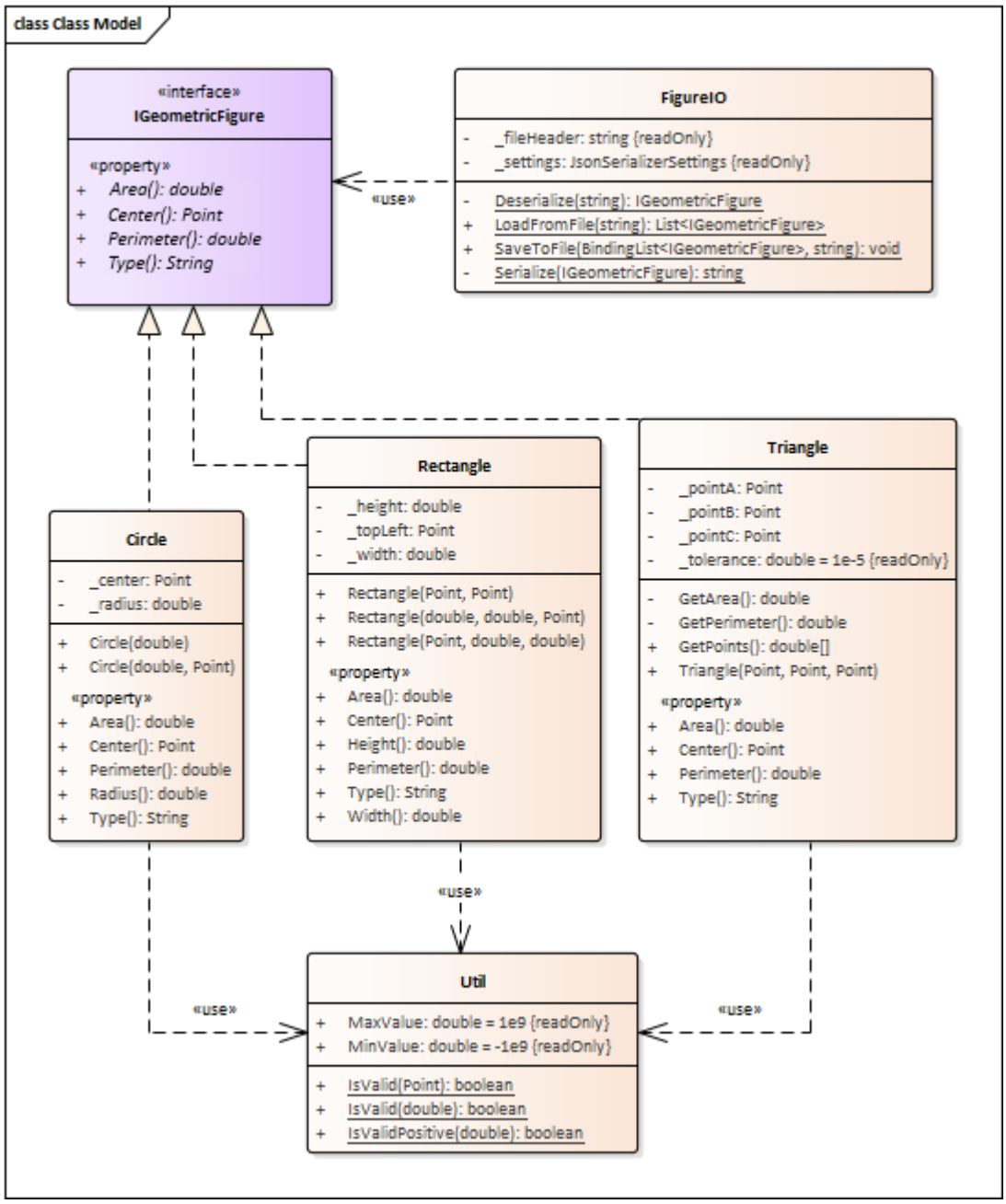

Рисунок 3.3 – Классы пакета Model

UML-диаграммы не являются единственным способом описания структуры программы. Более подробным в плане представления внутренней структуры классов является их табличное описание, где помимо имён и типов свойств, полей и методов представлены их краткие характеристики. В таблице 3.1 приведено описание интерфейса IGeometricFigure.

Таблица 3.1 – Описание интерфейса IGeometricFigure

Название	Тип	Описание
Описание интерфейса		
Интерфейс IGeometricFigure — сущность, описывающая общие для всех типов геометрических фигур в программе свойства		
Свойства		
+ Area	double	Площадь фигуры
+ Center	Point	Координаты центра фигуры
+ Perimeter	double	Периметр фигуры
+ Type	String	Тип фигуры

Каждый из классов, реализующих данный интерфейс, для своей работы требует использования дополнительных свойств, полей и методов, не специфицируемых интерфейсом. Далее приводятся описания каждого из классов геометрических фигур.

Описание класса Circle приведено в таблице 3.2.

Таблица 3.2 – Описание класса Circle

Название	Тип	Описание
Описание класса		
Класс Circle — сущность для описания круга в программе		
Свойства		
+ Radius	double	Радиус круга, неотрицателен
Поля		
- _center	Point	Координаты центра фигуры, реализация свойства интерфейса
- _radius	double	Внутреннее представление радиуса для реализации свойства Radius
Методы		
+ Circle(double)		Конструктор по радиусу, центр устанавливается в точку (0; 0). double — радиус круга
+ Circle(double, Point)		Конструктор по радиусу и центральной точке. double — радиус круга; Point — центр круга

Описание класса Rectangle приведено в таблице 3.3

Таблица 3.3 – Описание класса Rectangle

Название	Тип	Описание
Описание класса		
Класс Rectangle — сущность для описания прямоугольника в программе		
Свойства		
+ Height	double	Высота прямоугольника, неотрицательна
+ Width	double	Ширина прямоугольника, неотрицательна
Поля		
- _height	double	Внутреннее представление высоты для реализации свойства Height
- _topleft	Point	Координата верхнего левого угла прямоугольника
- _width	double	Внутреннее представление ширины для реализации свойства Width
Методы		
+ Rectangle(Point, Point)		Конструктор по верхней левой и нижней правой точкам. Point — верхняя левая точка прямоугольника; Point — нижняя правая точка прямоугольника
+ Rectangle(double, double, Point)		Конструктор по ширине, высоте и центральной точке. double — ширина; double — высота; Point — центр
+ Rectangle(Point, double, double)		Конструктор по верхней левой точке, ширине и высоте. Point — верхняя левая точка; double — ширина; double — высота

Описание класса Triangle приведено в таблице 3.4

Таблица 3.4 – Описание класса Triangle

Название	Тип	Описание
Описание класса		
Класс Triangle — сущность для описания треугольника в программе		
Поля		
- _pointA	Point	Первая точка треугольника
- _pointB	Point	Вторая точка треугольника
- _pointC	Point	Третья точка треугольника

Окончание таблицы 3.4

Название	Тип	Описание
- _tolerance	const double	Константа для сравнения числового типа double. Если два числа различаются меньше, чем на это значение, они считаются равными
Методы		
- GetArea()	double	Расчёт площади треугольника для реализации свойства интерфейса
- GetPerimeter()	double	Расчёт периметра треугольника для реализации свойства интерфейса
+ GetPoints()	double[]	Получение координат всех точек треугольника
+ Triangle(Point, Point, Point)		Конструктор по трём точкам. Point — первая точка треугольника; Point — вторая точка треугольника; Point — третья точка треугольника

3.3 Разработка программы

Разработка программы велась с использованием следующего стека технологий:

– язык C# 5.0, .NET Framework 4.5.2;
– IDE SharpDevelop 5.1;
– система контроля версий git 2.13.3;
– библиотеки Windows Forms и Newtonsoft.Json;
– библиотека модульного тестирования NUnit 2.6.4.

Для каждого свойства, поля и метода каждого класса в коде имеется XML-комментарий, кратко характеризующий его назначение, что облегчает дальнейшее переиспользование кода программы.

3.4 Система контроля версий

Для хранения исходного кода программы на сервисе GitLab был создан git-репозиторий ntvp-labs, расположенный по адресу https://gitlab.com/kalitka/ntvp-labs.

На рисунке 3.4 приведены последние двадцать три коммита (из восьмидесяти) в ветку master репозитория. Данное представление получено с помощью программы gitk, входящей в стандартную поставку git для ОС Windows.

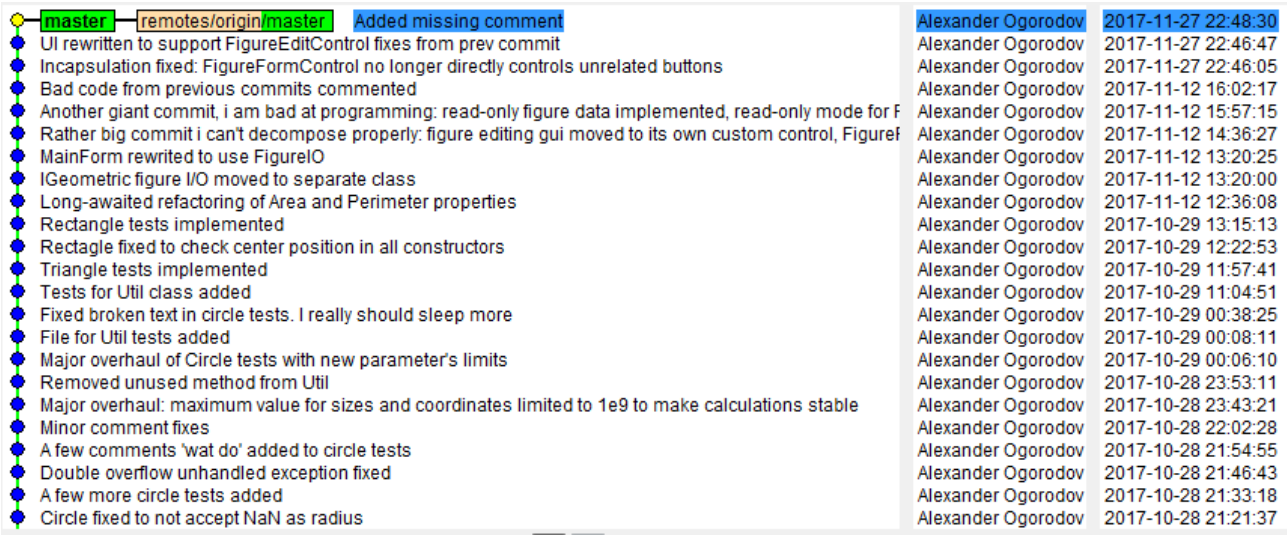

Рисунок 3.4 – Последние коммиты в репозиторий

3.5 Тестирование программы

Тестирование программы позволяет убедиться, что реальное поведение программы соответствует ожидаемому, описанному в документации. Своевременное тестирование даёт возможность выявлять ошибки на ранних стадиях разработки, что значительно снижает затраты на их устранение и в целом упрощает разработку и дальнейшую поддержку программы.

3.5.1 Модульное тестирование

Модульное тестирование — процесс изолированного тестирования отдельных модулей программ для подтверждения корректности их работы.

В данной программе для классов пакета Model — Circle, Rectangle, Triangle и Util — с использованием библиотеки модульного тестирования NUnit были написаны тесты для всех публичных свойств и методов. Были использованы как позитивные тесты, проверяющие правильность работы классов на корректных данных, так и негативные, проверяющие правильность типа выдаваемых классом исключений при подаче на вход некорректных данных. Общее количество написанных тестовых случаев — 147. На тестовом компьютере (процессор AMD Ryzen 3 1300X, видеокарта NVIDIA GeForce GTX 750 TI, 8 ГБ оперативной памяти, 64-битная версия ОС Microsoft Windows 7) среднее время выполнения всех тестов составляет 0,13 с. На рисунке 3.5 приведена иерархия тестов, отображаемая в IDE SharpDevelop.

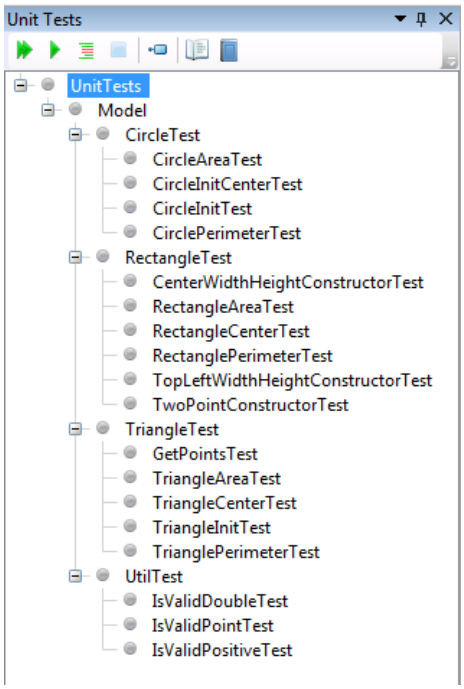

Рисунок 3.5 – Иерархия модульных тестов

3.5.2 Функциональное тестирование

Функциональное тестирование предназначено для проверки реализуемости в программе функциональных требований к ней, заданных ТЗ.

Проверка вводимых пользователем данных о геометрических фигурах не даёт создавать экземпляры фигур с некорректными данными — кнопка «OK» для подтверждения добавления или изменения фигуры блокируется до исправления некорректных данных, выделяемых полужирным шрифтом. Примеры ввода корректных и некорректных данных приведены на рисунке 3.6.

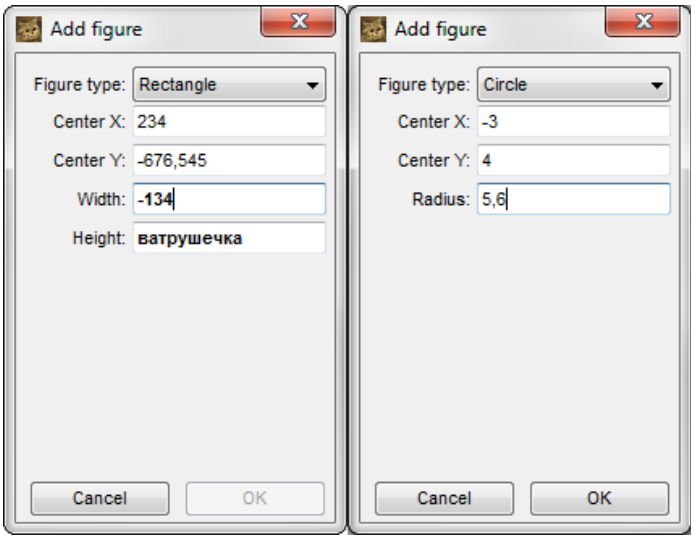

Рисунок 3.6 – Ввод некорректных (слева) и корректных (справа) данных

На тестовом компьютере программа с открытым файлом БД с предельно допустимым количеством записей — 50 — потребляет около 40 МБ оперативной памяти, что ниже установленного в ТЗ порога в 50 МБ. Программа так же укладывается и в ограничения по быстродействию: операции над записями в БД из 50 записей проходят быстрее, чем за 3 секунды.

3.6 Сборка установщика

Для сборки установщика программы была использована свободно распространяемая утилита с открытым исходным кодом InnoSetup [5]. С её помощью был создан установочный файл, размещающей в заданном каталоге (по умолчанию — C:\Program Files\guidb_geometric) исполняемый файл программы, GUI.exe, две библиотеки, необходимые для его работы — Model.dll (бизнес-логика) и Newtonsoft.Json.dll (работа с форматом JSON) и файлы, позволяющие отменить установку программы. По желанию пользователя также возможно создание ярлыка для исполняемого файла программы на рабочем столе.

3.7 Расчёт стоимости программы

Хотя на создание программы отводилось два с половиной месяца, реальное количество времени, затраченное на разработку, меньше. Точных сведений о чистом времени разработки программы не сохранилось, но из анализа времени создания коммитов в git-репозитории проекта можно примерно оценить значение: около пятидесяти часов вместе с написанием документации. Зная время, затраченное на разработку программы, можно приблизительно подсчитать стоимость её разработки Q по формуле:

$$Q = S \cdot k \cdot \frac{t_{actual}}{t_{monthly}} \quad , \tag{3.1}$$

где S — заработная плата разработчика за месяц;

k — коэффициент, обычно равный 1,3;

t_{actual} — затраченное на разработку время в часах;

t_{monthy} — количество рабочих часов за месяц.

Если считать, что средняя заработная плата за месяц работы начинающего программиста по восемь часов пять дней в неделю (8·5·4 = 160 часов в месяц) составляет 20 000 рублей, то по формуле (3.1) стоимость разработки программы составляет:

$$Q = 20\,000 \cdot 1{,}3 \cdot \frac{50}{160}\,руб. = 8\,125\,руб \quad .$$

4 Заключение

В ходе лабораторных работ курса «Новые технологии в проектировании» была создана программа, полностью отвечающая требованиям, описанным в ТЗ на неё: все поставленные для достижения цели задачи выполнены. Программа полностью соответствует заданным критериям качества и прочим требованиям и, соответственно, подходит для решения исходной проблемы из ТЗ. Написанная программа также сопровождается документацией.

Проведённое тестирование — модульное и функциональное — подтверждает правильность поведения программы в том числе в тех случаях, когда в интерфейс программы вводятся заведомо некорректные данные.

Так как в проектной документации присутствует описание архитектуры программы, а весь её исходный текст прокомментирован, её расширение или повторное использование классов в других проектах не вызовет больших сложностей.

Стоимость разработки программы составила 8 125 рублей. В процессе разработки большое количество времени ушло на изучение работы с библиотеками, в частности NUnit и Newtonsoft.Json. Создание программы, аналогичной данной для другой предметной области, займёт в несколько раз меньше времени, и соответственно, будет в несколько раз дешевле.

Список использованных источников

1. Desktop Windows Version Market Share Worldwide | StatCounter Global Stats. [Электронный ресурс]. — Режим доступа: http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide (дата обращения: 11.12.2017).

2. About the Unified Modeling Language Specification Version 2.5. [Электронный ресурс]. — Режим доступа: http://www.omg.org/spec/UML/About-UML/ (дата обращения 15.12.2017).

3. Новые технологии в программировании : учебное пособие / Д. В. Гарайс, А. Е. Горяинов, А. А. Калентьев. — Томск : Эль Контент, 2014. — 176 с.

4. Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения. / С. А. Орлов. — СПб.: Питер, 2013. — 688 с.

5. Inno Setup. [Электронный ресурс]. — Режим доступа: http://www.jrsoftware.org/isinfo.php (дата обращения 15.12.2017).