## **Практическое занятие 1**

## **Объектно-ориентированный анализ и проектирование приложения Конвертор чисел.**

**Цель:** Объектно-ориентированный анализ, проектирование и реализация приложения «Конвертор чисел» под Windows для преобразования действительных чисел представленных в системе счисления с основанием p1 в действительные числа, представленные в системе счисления с основанием p2.

В процессе выполнения работы студенты **изучают**: отношения между классами: ассоциация, агрегация, зависимость, их реализацию средствами языка программирования высокого уровня; этапы разработки приложений в технологии ООП; элементы технологии визуального программирования; диаграммы языка UML для документирования разработки.

**Функциональные требования к проектируемому приложению**

Интерфейс приложения может выглядеть так (рисунок 1):

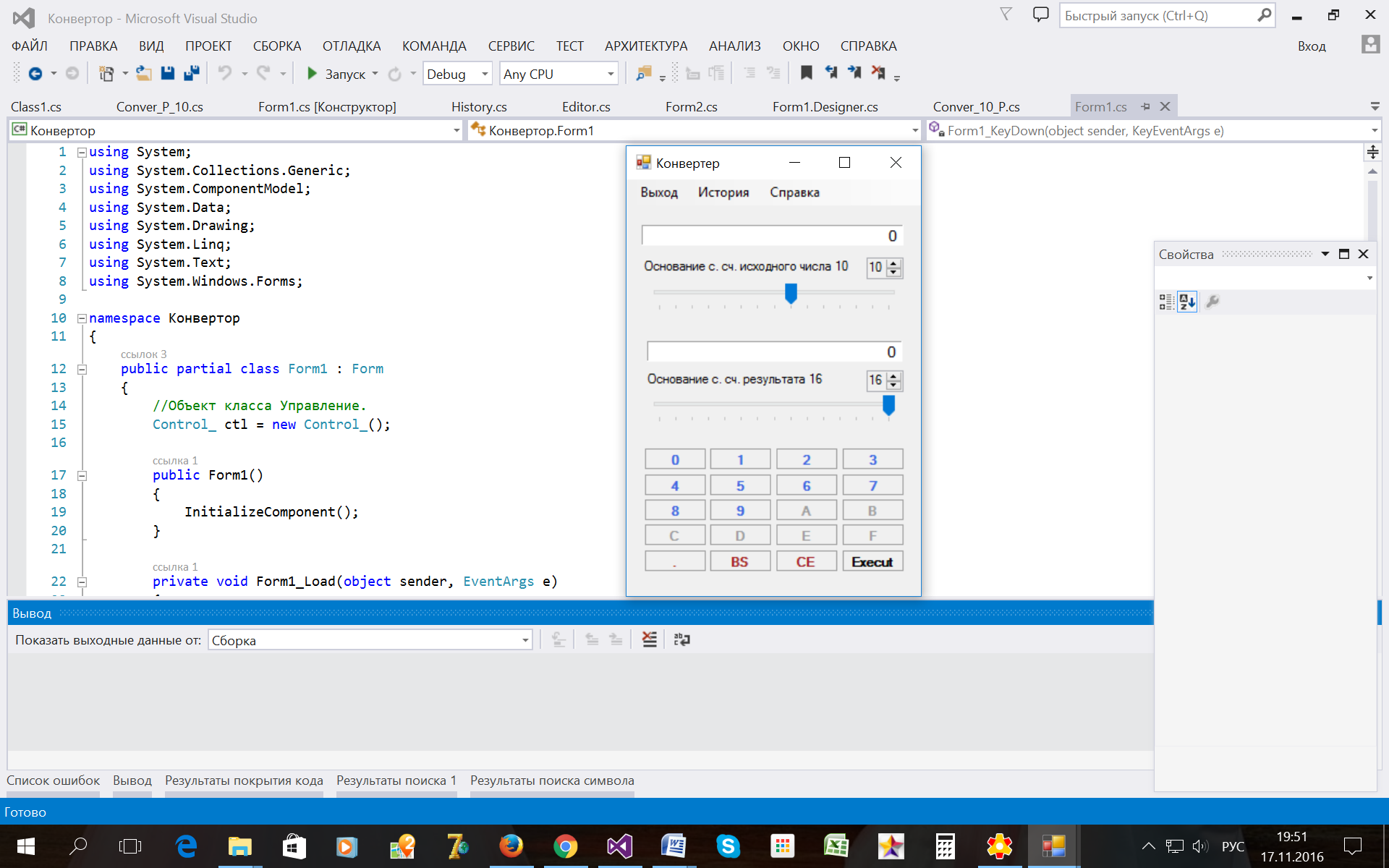


Рисунок 1 - Главная форма приложения

Приложение должно обеспечивать пользователю:

1. Преобразование действительного числа представленного в системе счисления с основанием p1 в число, представленное в системе счисления с основанием p2. Основания систем счисления p1, p2 для исходного числа и результата преобразования выбираются пользователем из диапазона от 2..16.
2. Возможность ввода и редактирования действительного числа представленного в системе счисления с основанием p1 с помощью командных кнопок и мыши, а также с помощью клавиатуры.
3. Контекстную помощь по элементам интерфейса.
4. Справку о назначении приложения.
5. Просмотр истории сеанса (журнала) работы пользователя с приложением – исходные данные, результат преобразования и основания систем счисления, в которых они представлены; дополнительные повышенные требования: автоматический расчёт необходимой точности представления результата.

Функциональные требования представлены диаграммой прецедентов (use-case диаграммой) расположенной на рисунке 2.

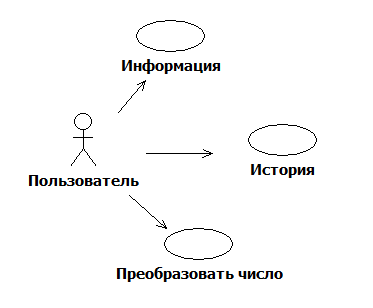


Рисунок 2 - Диаграммой прецедентов

**Сценарий для прецедента «Преобразовать число»**

*Основной поток событий*

1. Пользователь выбирает основание системы счисления p1 исходного числа.
2. Пользователь выбирает основание системы счисления p2 результата.
3. Пользователь вводит действительное число, представленное в системе счисления с основанием p1.
4. Пользователь вводит команду «Преобразовать».
5. Система выводит введённое пользователем число, представленное в системе счисления с основанием p2.
6. Система сохраняет исходные данные и результат преобразования в Историю.

*Альтернативный поток событий 1. Введённое пользователем число выходит за границы допустимого диапазона.*

3.1. Пользователь получает окно с сообщением.

3.2. Приложение переходит в режим Ввод и редактирование.

*Альтернативный поток событий 2. Количество разрядов в результате превышает размер поля вывода визуального компонента.*

4.1. Пользователь получает окно с сообщением.

4.2. Приложение переходит в режим Ввод и редактирование.

Можно осуществить декомпозицию прецедента Преобразовать, в результате мы получим для него следующую диаграмму (рисунок 3):

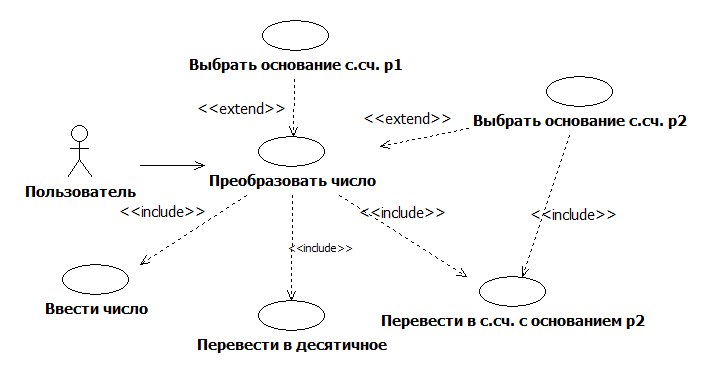


Рисунок 3 - Декомпозиция прецедента Преобразовать

**Сценарий для прецедента «Преобразовать»**

*Предусловие*

Завершён ввод и редактирования исходного числа.

*Основной поток событий*

1. Пользователь вводит команду «Преобразовать».
2. Система выводит введённое пользователем число, представленное в системе счисления с основанием p2.
3. Система сохраняет исходные данные и результат преобразования в Историю.

*Альтернативный поток событий 1. Введённое пользователем число выходит за границы допустимого диапазона.*

1.1. Пользователь получает окно с сообщением.

1.2. Приложение переходит в режим Ввод и редактирование.

*Альтернативный поток событий 2. Количество разрядов в результате превышает размер поля вывода визуального компонента.*

1.1. Пользователь получает окно с сообщением.

1.2. Приложение переходит в режим Ввод и редактирование.

**Сценарий для прецедента «Выбрать основание с. сч. p2»**

*Предусловие*

Прецедент «Преобразовать» завершён.

*Основной поток событий*

1. Пользователь изменяет основания систем счисления p2.
2. Введённое пользователем число отображается в системе счисления с выбранным основанием.

*Альтернативный поток событий 1. Количество разрядов в результате превышает размер поля вывода визуального компонента.*

3.1. Пользователь получает окно с сообщением.

3.2. Приложение переходит в режим Ввод и редактирование.

**Диаграмма классов модели объектно-ориентированного анализа.**

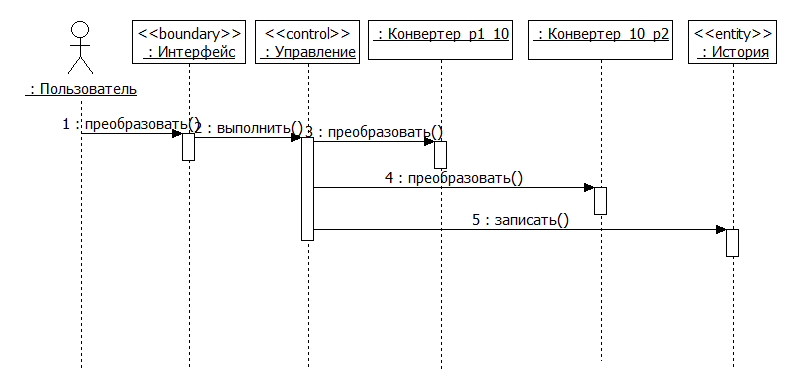
Проанализировав прецеденты можно выделить следующие классы анализа приложения. Они представлены на диаграмме классов анализа на рисунке 4.



**Рисунок 4 -** Диаграмма классов анализа

**Обмен сообщениями между объектами. Диаграмма последовательностей.**

Давайте спроектируем обмен сообщениями между объектами в процессе выполнения прецедента «Преобразовать». Добавим объект класса Управление для организации обмена сообщениями между объектами в ходе выполнения прецедента. На диаграмме последовательностей приведённой на рисуноке 5 приведёна последовательность сообщений между объектами в основном потоке событий прецедента «Преобразовать».

Рисунок 5 - Диаграмма последовательностей

На диаграмме последовательностей приведённой на рисунке 6 приведёна последовательность сообщений между объектами в процессе реализации прецедента «Ввести число».

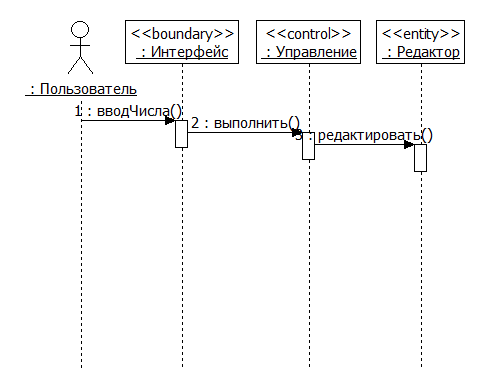
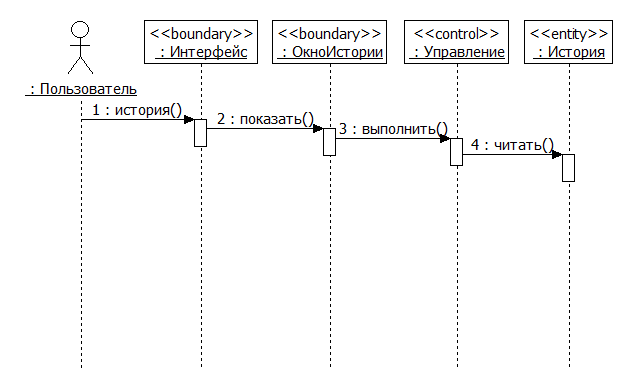


Рисунок 6 - Последовательность сообщений в прецеденте «Ввести число».

На диаграмме последовательностей приведённой на рисунке 7 приведёна последовательность сообщений между объектами в процессе реализации прецедента «История».



**Рисунок 7 -** Последовательность сообщений в прецеденте «История»

**Диаграмма классов проекта**

Проанализировав сообщения, которыми обмениваются классы в процессе выполнения прецедентов можно построить следующую диаграмму классов проекта. Для упрощения взаимодействия между классами в процессе работы приложения добавим класс «Управление». Тогда наша диаграмма примет следующий вид (рисунок 8):

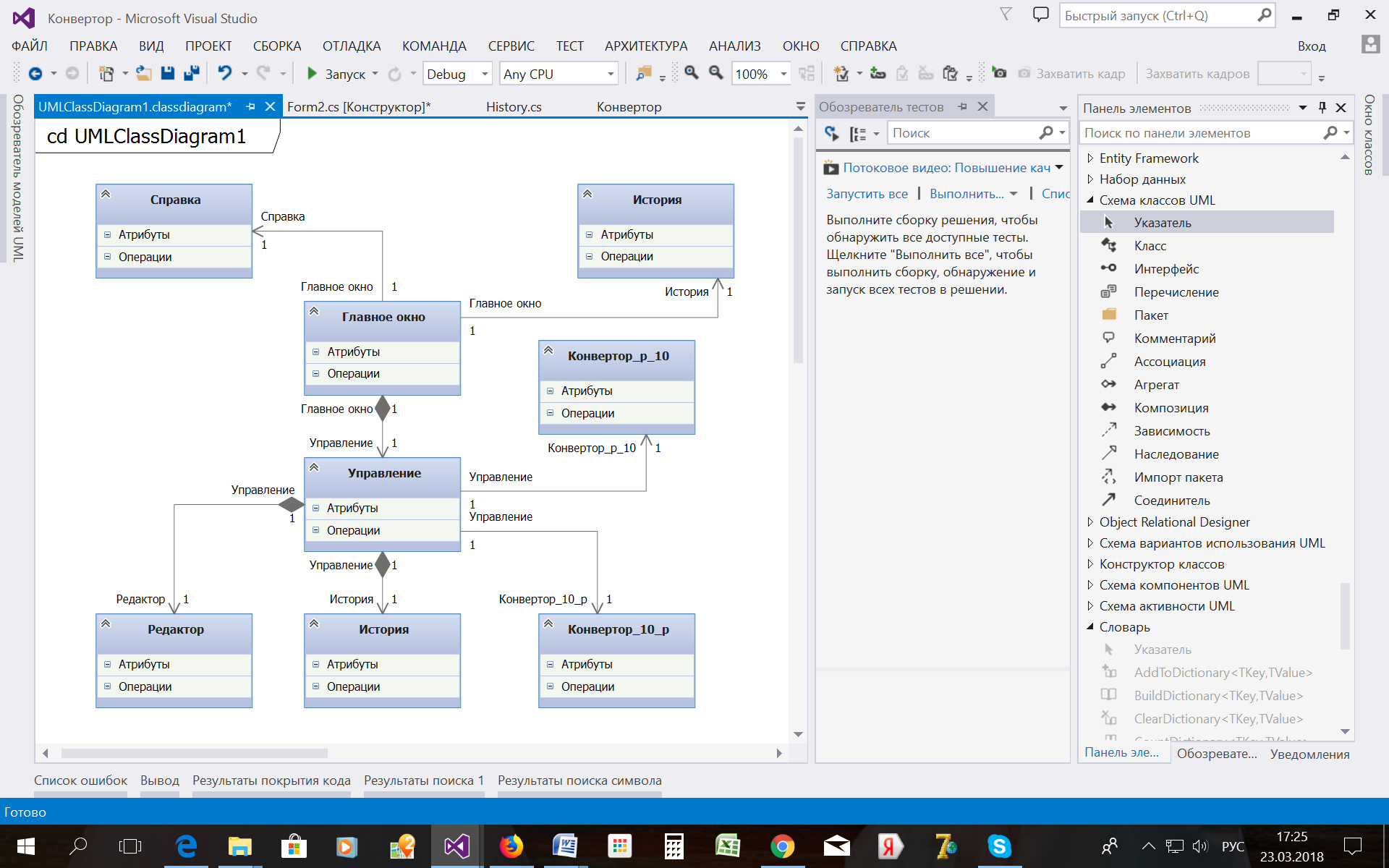


Рисунок 8 - Диаграмма классов проекта

Из диаграммы классов видно, что объект интерфейсного класса «Главное окно» вызывает методы класса «Управление», «История» и «Справка». Объект же класса «Управление» в свою очередь вызывает методы объектов классов «Редактор», «История» и «Конвертор\_p1\_10», «Конвертор\_10\_p2». Диаграмма состояний для объекта класса «Управление» представлена (рисунок 9):



Рисунок 9 - Диаграмма состояний для объекта класса «Управление»

Объект класса «Управление» может находиться в двух состояниях: «Редактирование» и «Редактирование завершено».

Порядок выполнения разработки.

Разработка приложения разбита на лабораторные работы, в каждой из которых вы реализуете один или несколько логически связанных классов приложения.