

Лабораторная работа № 4

«Диаграмма классов»

Оглавление

Общее понятие	3
Класс	3
<i>Атрибуты</i>	<i>4</i>
<i>Операция</i>	<i>6</i>
Отношения между классами	7
<i>Отношение зависимости</i>	<i>7</i>
<i>Отношение ассоциации</i>	<i>8</i>
<i>Отношение агрегации</i>	<i>9</i>
<i>Отношение композиции</i>	<i>9</i>
<i>Отношение обобщения</i>	<i>9</i>
Интерфейсы	12
Объекты	12
Примеры диаграмм классов	13
Построение диаграммы классов в MS Visual Studio	15
Задание к лабораторной работе	21
Контрольные вопросы	22
Приложение	23

Общее понятие

От умения правильно выбрать классы и установить между ними взаимосвязи часто зависит не только успех процесса проектирования, но и производительность выполнения программы.

Нотация классов в языке UML проста и интуитивно понятна всем, кто когда-либо имел опыт работы с CASE-инструментариями. Схожая нотация применяется и для объектов – экземпляров класса, с тем различием, что к имени класса добавляется имя объекта и вся надпись подчеркивается.

Нотация UML предоставляет широкие возможности для отображения дополнительной информации (абстрактные операции и классы, стереотипы, общие и частные методы, детализированные интерфейсы, параметризованные классы). При этом возможно использование графических изображений для ассоциаций и их специфических свойств, таких как отношение агрегации, когда составными частями класса могут выступать другие классы.

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят или инвариантны от времени.

Класс

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис. 5.1). В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

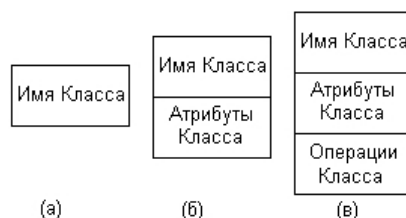


Рис. 1. Графическое изображение класса на диаграмме классов

Примеры

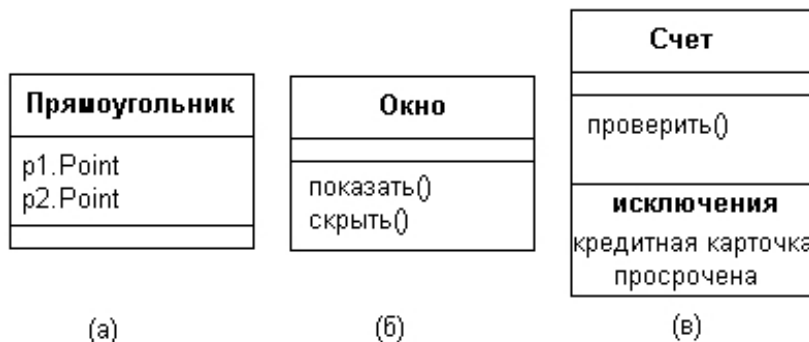


Рис. 2. Примеры классов

Имя класса должно быть уникальным в пределах пакета, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой). Оно указывается в первой верхней секции прямоугольника. В дополнение к общему правилу наименования элементов языка UML, имя класса записывается по центру секции имени полужирным шрифтом и должно начинаться с заглавной буквы.

Класс может не иметь экземпляров или объектов. В этом случае он называется абстрактным классом, а для обозначения его имени используется наклонный шрифт (курсив). В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом. Данное обстоятельство является семантическим аспектом описания соответствующих элементов языка UML.

Атрибуты

Во второй сверху секции прямоугольника класса записываются его атрибуты (attributes) или свойства. В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения.

Квантор видимости может принимать одно из трех возможных значений и, соответственно, отображается при помощи специальных символов:

- Символ "+" обозначает атрибут с областью видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.

- Символ "#" обозначает атрибут с областью видимости типа защищенный (protected). Атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса.

- И, наконец, знак "-" обозначает атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

Примеры атрибутов

- цвет: Color – здесь цвет является именем атрибута, Color – именем типа данного атрибута.

- имя_сотрудника [1..2] : String – здесь имя_сотрудника является именем атрибута, который служит для представления информации об имени, а возможно, и отчестве конкретного сотрудника. Тип атрибута String (Строка)

- форма:Многоугольник – здесь имя атрибута форма может характеризовать такой класс, который является геометрической фигурой на плоскости. В этом случае тип атрибута Многоугольник указывает на тот факт, что отдельная геометрическая фигура может иметь форму треугольника, прямоугольника, ромба, пятиугольника и любого другого многоугольника, но не окружности или эллипса.

Примеры значения атрибутов

- цвет:Color = (255, 0, 0) – в RGB-модели цвета это соответствует чистому красному цвету в качестве исходного значения для данного атрибута.

- имя_сотрудника[1..2]:String = Иван Иванович – возможно, это нетипичный случай, который, скорее, соответствует ситуации имя_руководителя[2]:81пп§ = Иван Иванович.

- форма: Многоугольник = прямоугольник – вряд ли требует комментариев, поскольку здесь речь идет о геометрической форме создаваемого объекта.

Операция

Операция (operation) представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию. Совокупность операций характеризует функциональный аспект поведения класса.

При этом каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, имени операции, выражения типа возвращаемого операцией значения и, возможно, строка-свойство данной операции.

Для повышения производительности системы одни операции могут выполняться параллельно или одновременно, а другие – только последовательно. В этом случае для указания параллельности выполнения операции используется строка-свойство вида «{concurrency = имя}», где имя может принимать одно из следующих значений: последовательная (sequential), параллельная (concurrent), охраняемая (guarded). При этом придерживаются следующей семантики для данных значений:

- последовательная (sequential) – для данной операции необходимо обеспечить ее единственное выполнение в системе, одновременное выполнение других операций может привести к ошибкам или нарушениям целостности объектов класса.

- параллельная (concurrent) – данная операция в силу своих особенностей может выполняться параллельно с другими операциями в системе, при этом параллельность должна поддерживаться на уровне реализации модели.

- охраняемая (guarded) – все обращения к данной операции должны быть строго упорядочены во времени с целью сохранения целостности объектов данного класса, при этом могут быть приняты дополнительные меры по контролю исключительных ситуаций на этапе ее выполнения.

Примеры операций

- +создать() – Эта операция не возвращает никакого значения после своего выполнения.

- +нарисовать(форма: Многоугольник = прямоугольник, цвет_заливки: Color = (O, O, 255)) – может обозначать операцию по изображению на экране монитора прямоугольной области синего цвета, если не указываются другие значения в качестве аргументов данной операции.

- запросить_счет_клиента(номер_счета:целое):целое – обозначает операцию по установлению наличия средств на текущем счете клиента банка. При этом аргументом данной операции является номер счета клиента, который записывается в виде целого числа (например, «123456»).

- выдать_сообщение(): {"Ошибка деления на ноль"} – смысл данной операции не требует пояснения, поскольку содержится в строке-свойстве операции. Данное сообщение может появиться на экране монитора в случае попытки деления некоторого числа на ноль, что недопустимо.

Отношения между классами

Базовыми отношениями или связями в языке UML являются:

- Отношение зависимости (dependency relationship)
- Отношение ассоциации (association relationship)
- Отношение обобщения (generalization relationship)
- Отношение реализации (realization relationship)

Отношение зависимости

Отношение зависимости в общем случае указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов.

Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели.

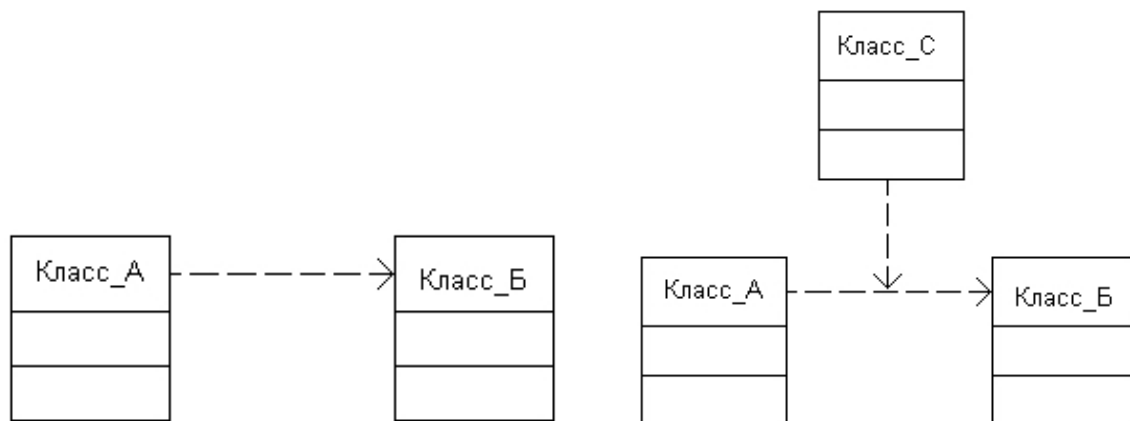


Рис. 3. Графическое изображение отношения зависимости на диаграмме классов

Отношение ассоциации

Отношение ассоциации соответствует наличию некоторого отношения между классами. Данное отношение обозначается сплошной линией с дополнительными специальными символами, которые характеризуют отдельные свойства конкретной ассоциации.

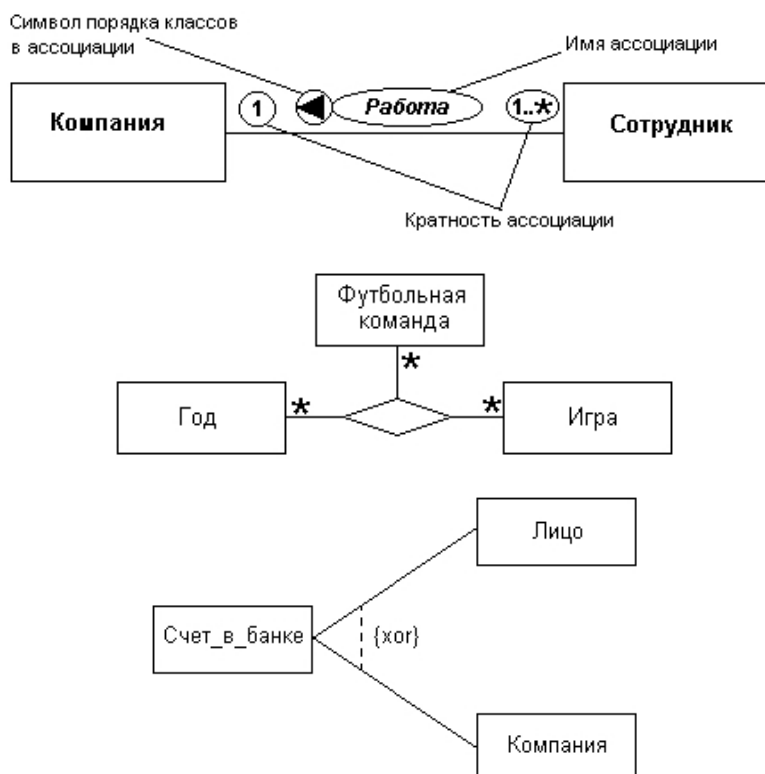


Рис. 4. Графическое изображение отношения ассоциации между классами

Отношение агрегации

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности («часть-целое»). Части системы никак не обязаны наследовать ее свойства и поведение, поскольку являются вполне самостоятельными сущностями. Более того, части целого обладают своими собственными атрибутами и операциями, которые существенно отличаются от атрибутов и операций целого.

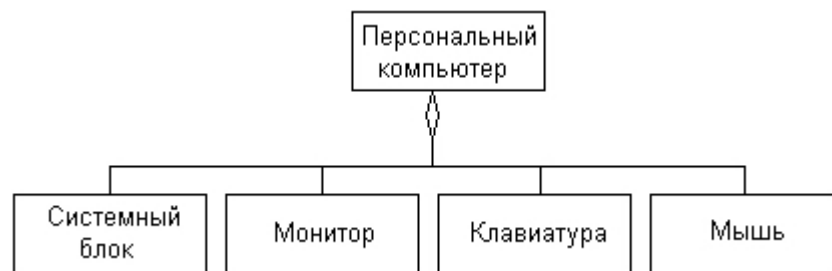


Рис. 5 Пример отношения агрегации

Отношение композиции

Отношение композиции является частным случаем отношения агрегации. Это отношение служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.

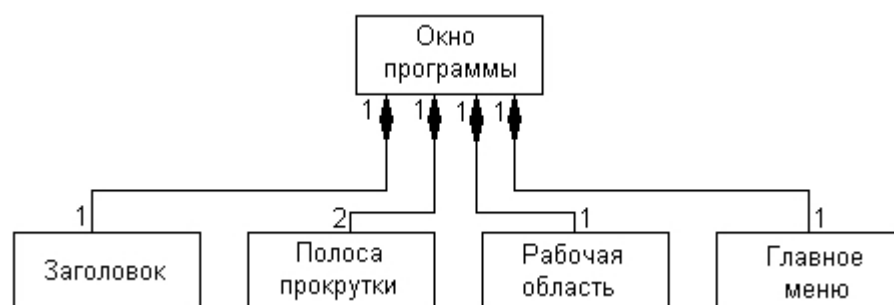


Рис. 6 Пример отношения композиции

Отношение обобщения

Отношение обобщения является обычным таксономическим отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком).

Данное отношение может использоваться для представления взаимосвязей между пакетами, классами, вариантами использования и другими элементами языка UML.

Применительно к диаграмме классов данное отношение описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка.

Рядом со стрелкой обобщения может размещаться строка текста, указывающая на некоторые дополнительные свойства этого отношения. Данный текст будет относиться ко всем линиям обобщения, которые идут к классам-потомкам. Другими словами, отмеченное свойство касается всех подклассов данного отношения. При этом текст следует рассматривать как ограничение, и тогда он записывается в фигурных скобках.

В качестве ограничений могут быть использованы следующие ключевые слова языка UML:

- {complete} – означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может. Пример – класс Клиент_банка является предком для двух классов: Физическое_лицо и Компания, и других классов-потомков он не имеет. На соответствующей диаграмме классов это можно указать явно, записав рядом с линией обобщения данную строку-ограничение;

- {disjoint} – означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов. В приведенном выше примере это условие также выполняется, поскольку предполагается, что никакое конкретное физическое лицо не может являться одновременно и конкретной компанией. В этом случае рядом с линией обобщения можно записать данную строку-ограничение;

- {incomplete} – означает случай, противоположный первому. А именно, предполагается, что на диаграмме указаны не все классы-потомки. В последующем возможно восполнить их перечень не изменяя уже построенную диаграмму. Пример – диаграмма класса «Автомобиль», для которой указание всех без исключения моделей автомобилей соизмеримо с созданием соответствующего каталога. С другой стороны, для отдельной задачи, такой как разработка системы продажи автомобилей конкретных моделей, в этом нет необходимости. Но указать неполноту структуры классов-потомков все же следует;

- {overlapping} – означает, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам. Пример – класс «Многоугольник» является классом-предком для класса «Прямоугольник» и класса «Ромб». Однако существует отдельный класс «Квадрат», экземпляры которого одновременно являются объектами первых двух классов. Вполне естественно такую ситуацию указать явно с помощью данной строки-ограничения.

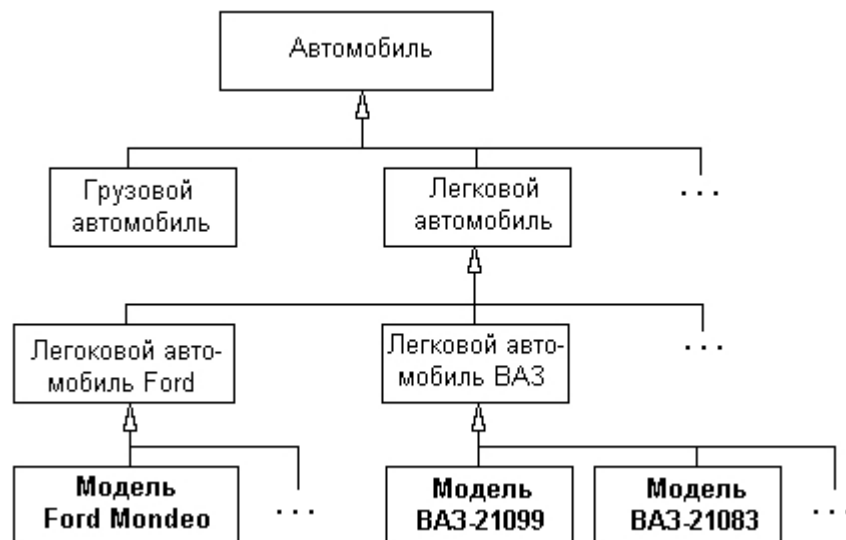
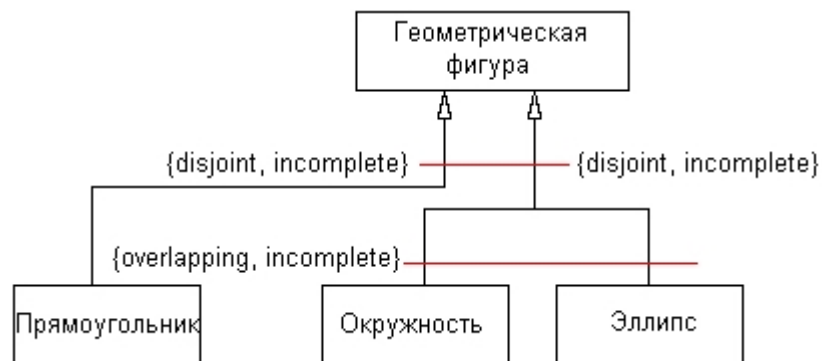
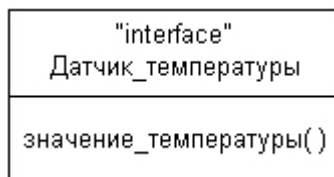


Рис. 7. Пример отношения обобщения

Интерфейсы



При этом секция атрибутов у прямоугольника отсутствует, а указывается только секция операций.

Объекты

Объект (object) является отдельным экземпляром класса, который создается на этапе выполнения программы. Он имеет свое собственное имя и конкретные значения атрибутов. В силу самых различных причин может возникнуть необходимость показать взаимосвязи не только между классами модели, но и между отдельными объектами, реализующими эти классы.

Для графического изображения объектов используется такой же символ прямоугольника, что и для классов. Отличия проявляются при указании имен объектов, которые в случае объектов обязательно подчеркиваются

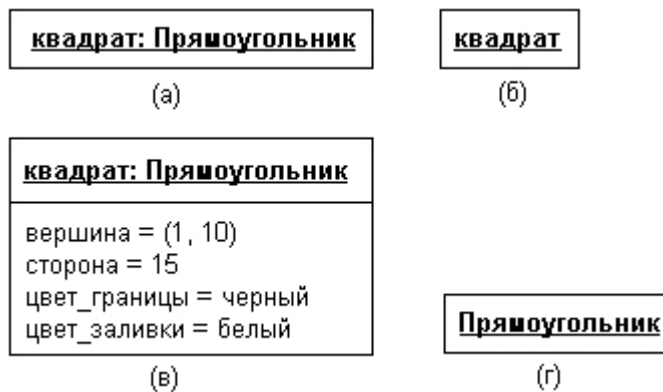


Рис. 8. Пример объектов класса

Примеры диаграмм классов

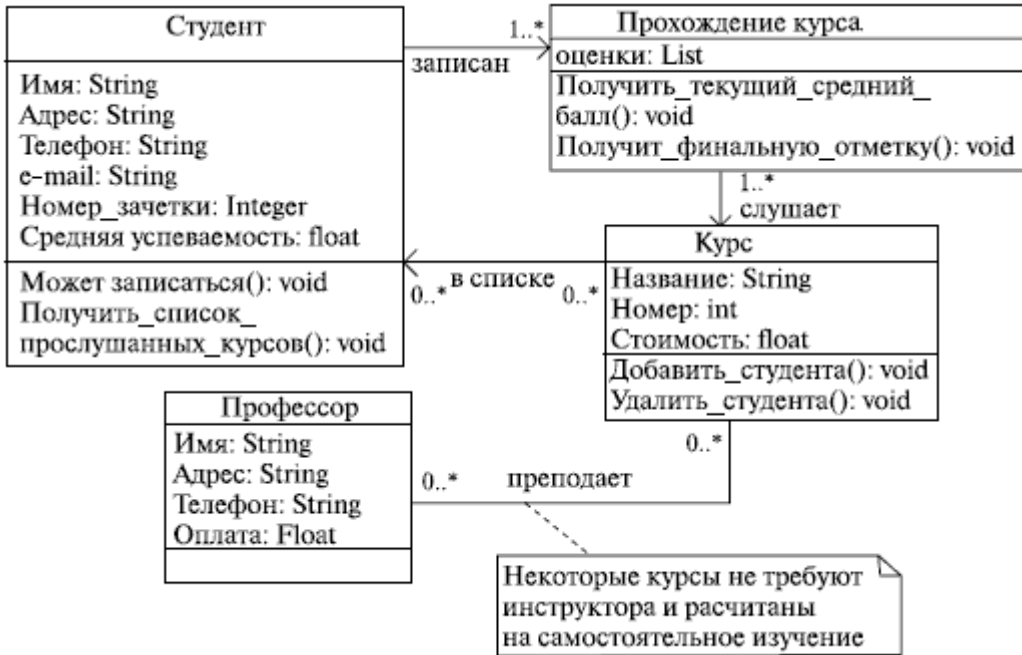
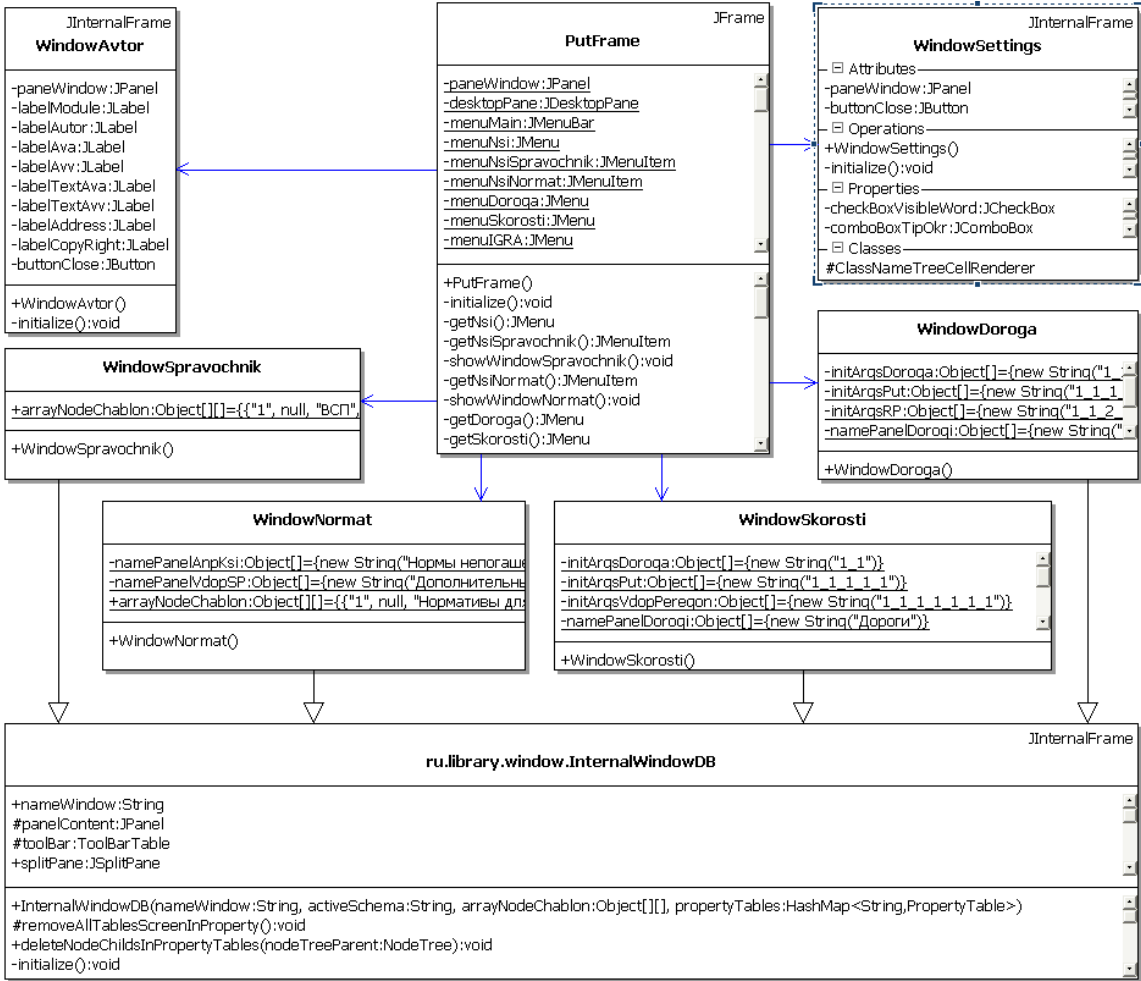
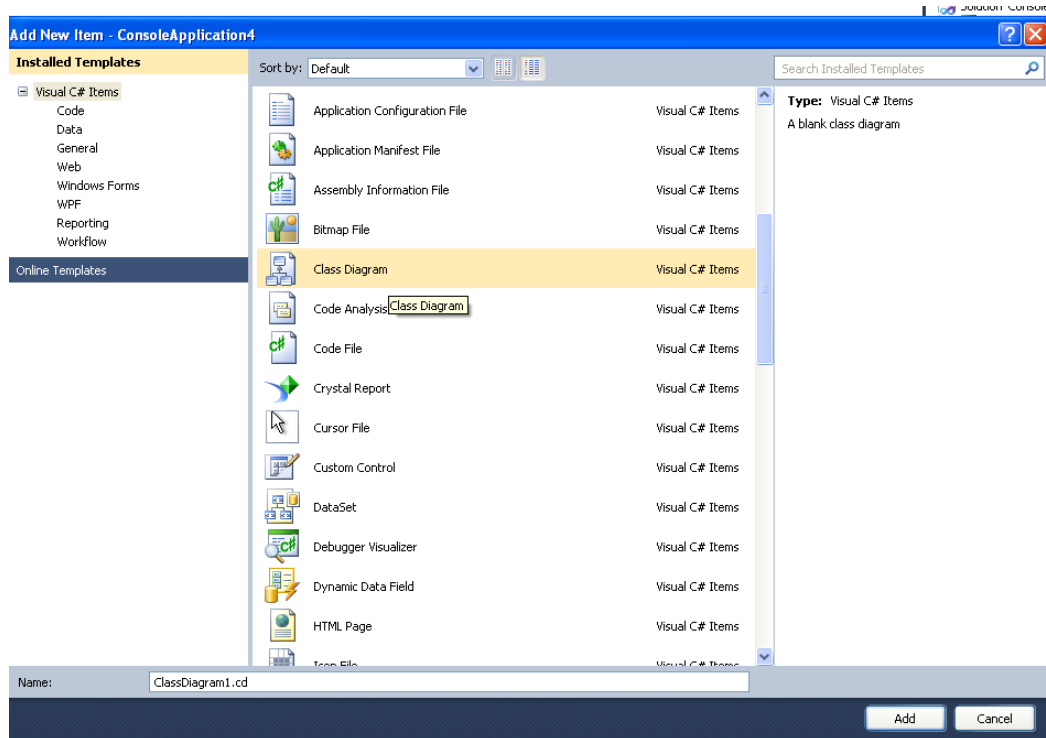
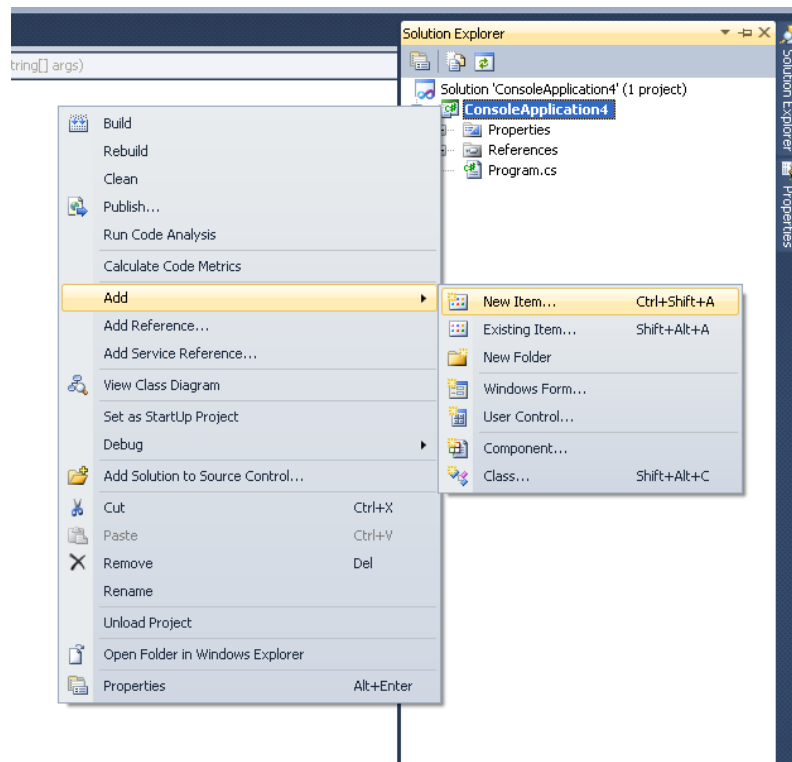


Диаграмма объектов классов

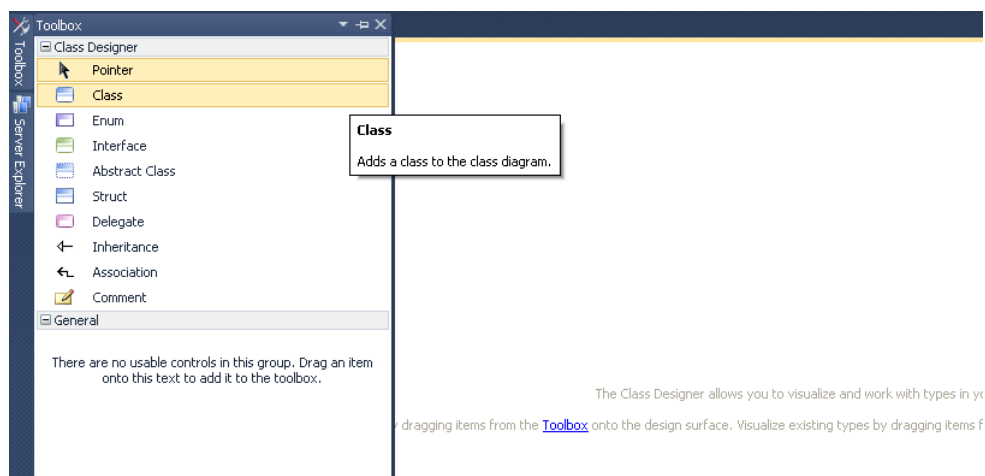


Построение диаграммы классов в MS Visual Studio

1. Создать консольное приложение
2. Добавить в проект элемент «Диаграмма классов»

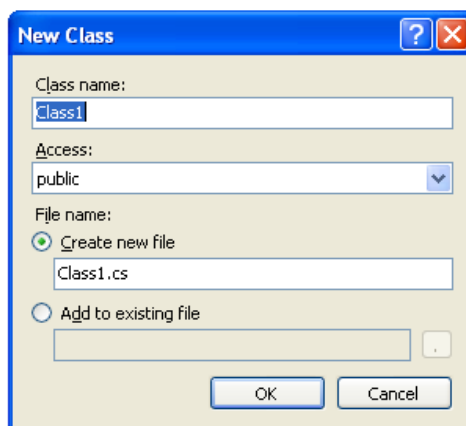


3. Построение диаграммы выполняется с помощью инструментов «Toolbox»



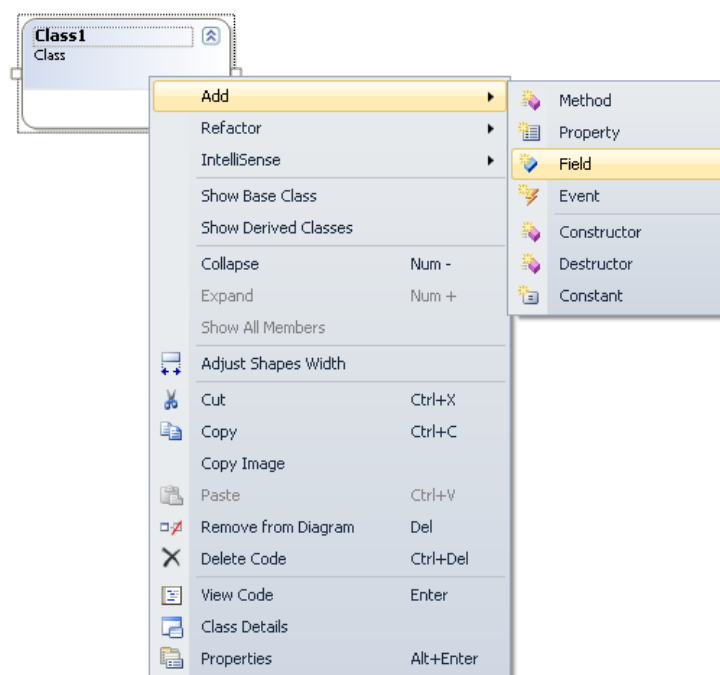
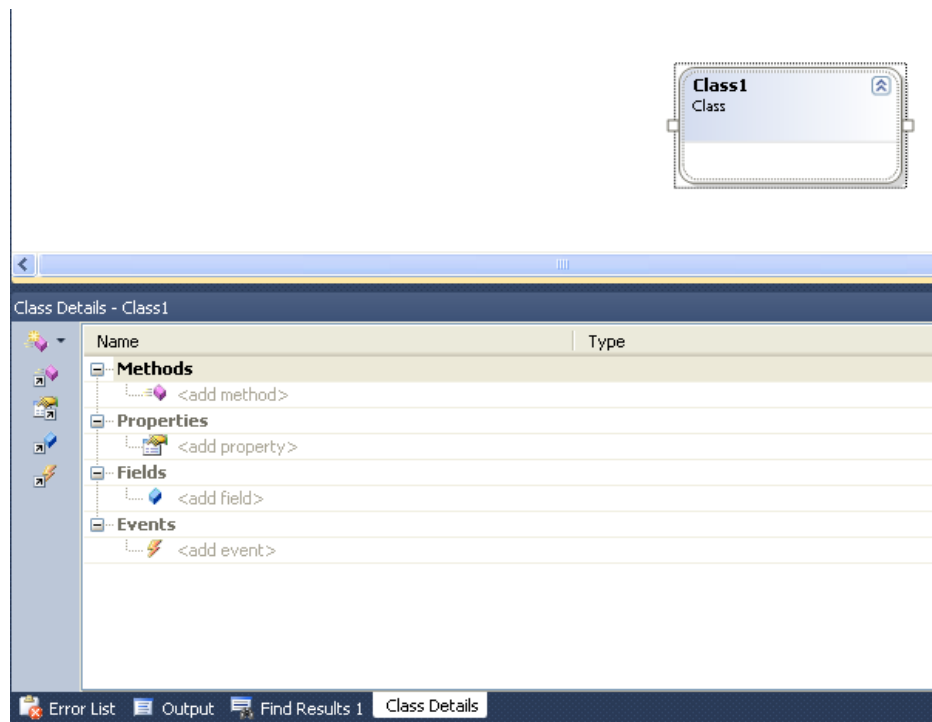
4. Для создания класса перетащите инструмент «Class» на поле диаграммы классов.

Выводится диалоговое окно для указания имени класса, области видимости и названия файла для создаваемого класса.

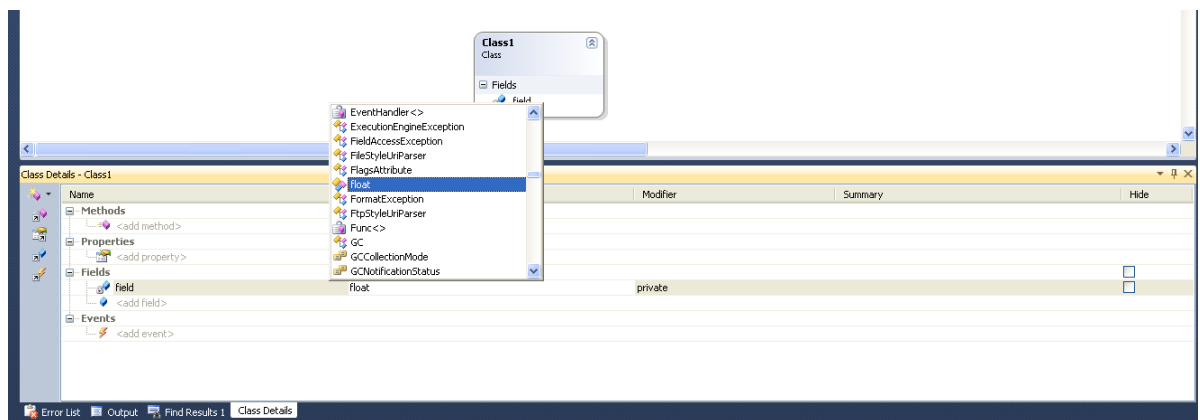
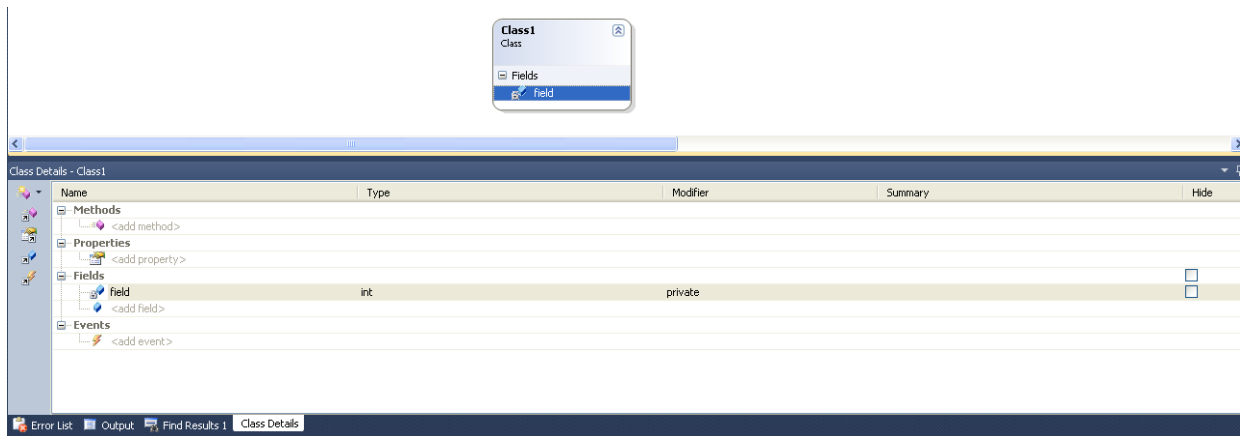


Созданный класс на диаграмме классов отображается в виде прямоугольника, при этом параметры класса указываются в окне «Class Details».

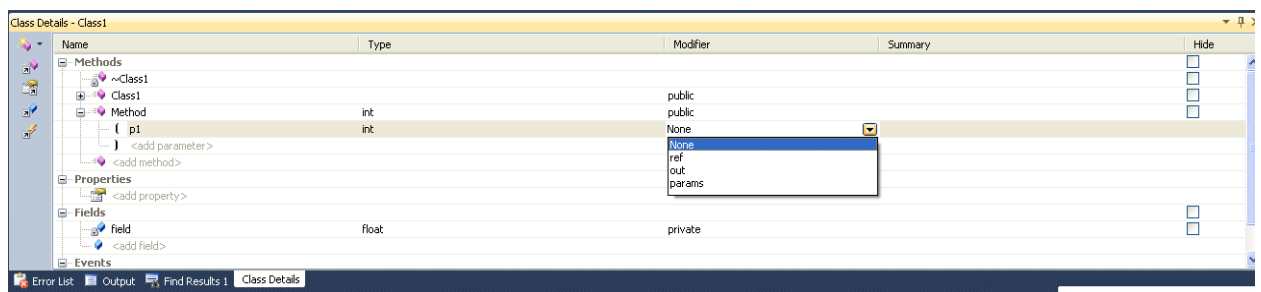
Для добавления полей и методов (в том числе конструктора и деструктора) в класс используется или окно «Class Details» или контекстное меню, после нажатия правой кнопки мыши на прямоугольнике, обозначающего созданный класс.



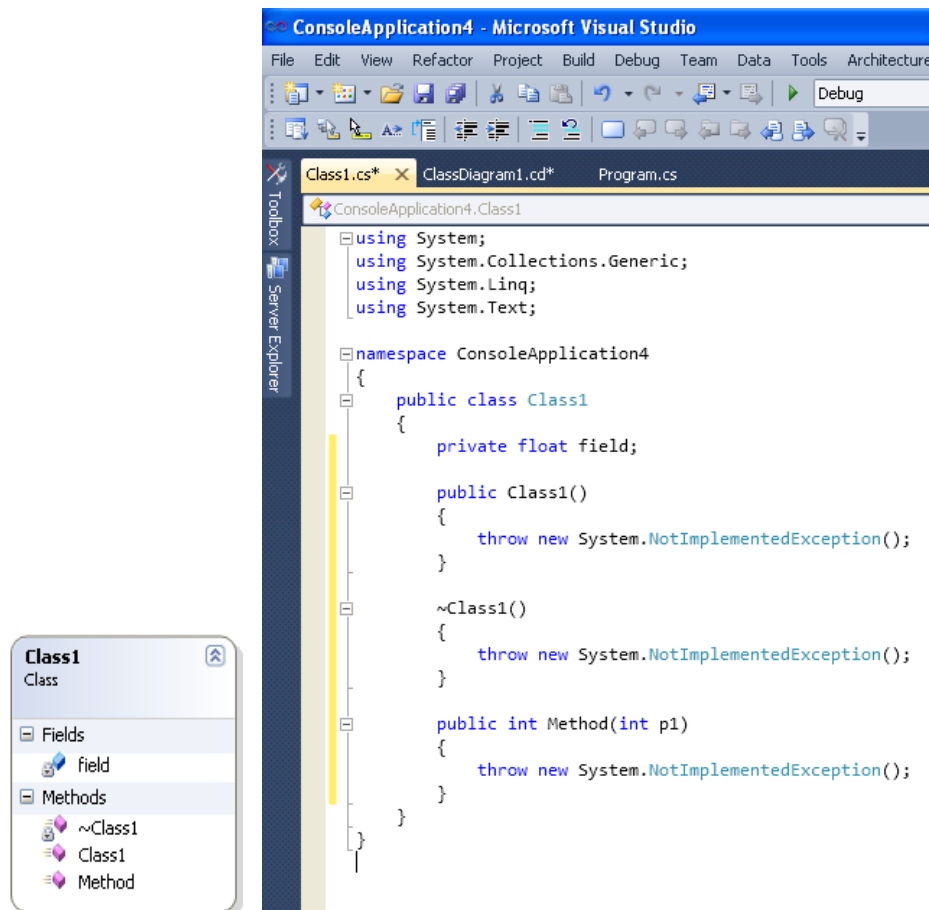
При добавлении поля указывается имя поля, область видимости и тип.



При добавлении метода указывается тип возвращаемого значения и набор передаваемых параметров.



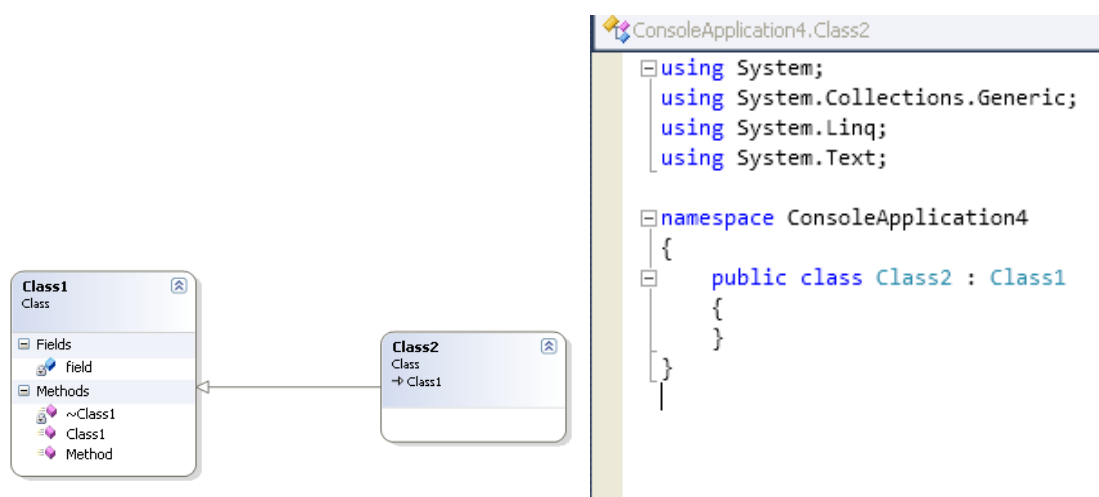
Добавленные в класс поля и методы прописываются в виде кода также и в файле, связанном с данным классом.



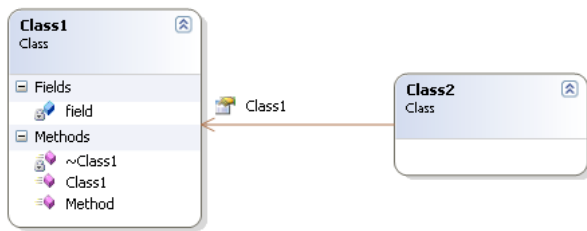
5. Взаимосвязь между классами

Поддерживается два вида взаимосвязей между классами Inheritance (наследование) и Association (ассоциация, объединение).

Связь Inheritance



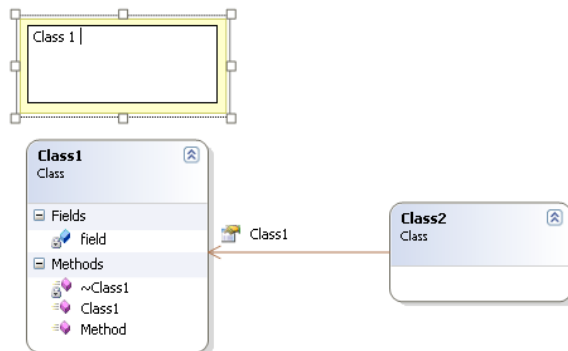
Связь Association



```
using System;

namespace ConsoleApplication4
{
    public class Class2
    {
        public Class1 Class1
        {
            get
            {
                throw new NotImplementedException();
            }
            set
            {
            }
        }
    }
}
```

6. Создание примечания (комментария)



Задание к лабораторной работе

1. Изучить теоретический материал.
2. Разработать диаграмму классов, соответствующую диаграмме вариантов использования, построенной в №1А лабораторной работе.
3. Создать диаграмму классов в оболочке для программирования (например, MS Visual Studio).
4. Построить диаграмму объектов класса.
5. Оформить отчет, который включает вариант задания, диаграмму вариантов использования, диаграмму классов, диаграмму объектов классов и полученный код.

Требования к диаграмме классов: содержит несколько классов с различными связями. Каждый класс должен иметь поля, свойства, различные методы.

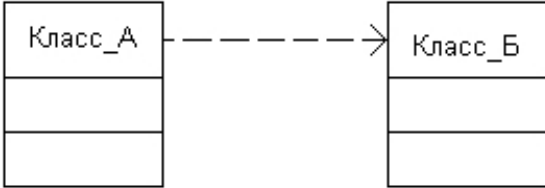


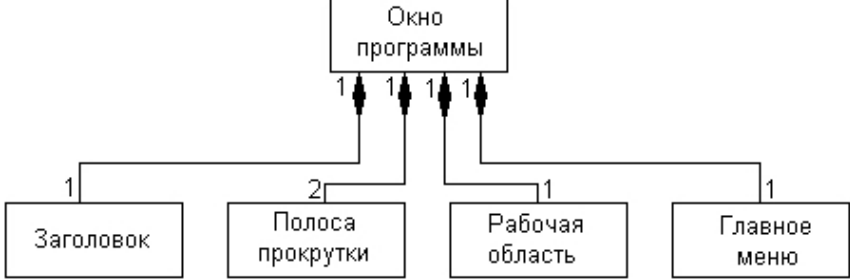

Варианты информационных систем

№	Информационная система
1.	Пассажир бронирует и покупает билет на рейс
2.	Клиент сдает автомобиль в автосервис
3.	Покупатель оформляет кредит на покупку товара
4.	Пассажир приходит на регистрацию рейса в аэропорт
5.	Клиент снимает квартиру через агентство недвижимости
6.	Определение списка студентов закрывших сессию в срок из указанной группы
7.	Формирование заказа на изготовление мебели
8.	Выдача книг в библиотеке
9.	Заправка автомобилей
10.	Формирование чека для оплаты покупок в супермаркете
11.	Учет автомобилей на автостоянке и расчет прибыли
12.	Формирование анкеты, проведение анкетирования и обработка результатов
13.	Диспетчер задач на компьютере
14.	Работа с группами пользователей, назначение прав доступа
15.	Формирование классного журнала в школе
16.	Печать фотографий и фотосувениров

Возможно, расширить функционал предлагаемой информационной системы.

Контрольные вопросы

1. Назначение диаграммы классов?
2. Как обозначается класс и что в себя включает?
3. Что такое атрибут в диаграмме классов?
4. Как обозначаются области видимости у атрибута?
5. Что такое операция в диаграмме классов?
6. Перечислить виды отношений между классами, охарактеризовать каждое из них?
7. Как обозначается объект (экземпляр) класса?

<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="border: 1px solid black; padding: 5px; width: 150px;"> <p>Прямоугольник</p> <p>p1.Point p2.Point</p> </div> <div style="border: 1px solid black; padding: 5px; width: 150px;"> <p>Окно</p> <p>показать() скрыть()</p> </div> <div style="border: 1px solid black; padding: 5px; width: 150px;"> <p>Счет</p> <p>проверить()</p> <p>исключения</p> <p>кредитная карточка просрочена</p> </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> (а) (б) (в) </div>	Классы
	Отношение зависимости (некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели)
<p>Символ порядка классов в ассоциации</p> <p>Имя ассоциации</p> <p>Кратность ассоциации</p> 	Отношение ассоциации
	Отношение агрегации
	Отношение композиции
	Отношение обобщения

<div>"interface"</div> <div>Датчик_температуры</div> <div>значение_температуры()</div>		Интерфейс
<div>квадрат: Прямоугольник</div> <div>(a)</div> <div>квадрат: Прямоугольник</div> <div>вершина = (1, 10)</div> <div>сторона = 15</div> <div>цвет_границы = черный</div> <div>цвет_заливки = белый</div> <div>(в)</div>	<div>квадрат</div> <div>(б)</div> <div>Прямоугольник</div> <div>(г)</div>	Объект