Практическая работа № 5.

Построение UML – модели системы. Диаграмма классов.

(задания №1-2 часа, №2 - 2 часа)

Цель работы: изучить структуру модели проектирования, правила построения диаграммы классов.

Задачи: описать сервисные функции исследуемой системы.

ПО: Visual Paradigm, Draw.io, Rational Rose.

Теоретический материал:

Диаграмма классов представляет собой логическую модель статического представления моделируемой системы. Задача заключается в том, чтобы представить поведение более детально на логическом уровне.

О**писание методологии моделирования классов в языке UML.**

Объект представляет собой экземпляр класса – особую сущность, которая имеет заданные значения атрибутов и операций.

Атрибуты

Атрибут – это свойство класса. Атрибуты описывают перечень значений, в рамках которых указываются свойства объектов (т.е. экземпляров) этого класса. Класс может не иметь атрибутов или содержать любое их количество. Имена атрибутов, состоящие из одного слова, принято обозначать строчными буквами. Если имя состоит из нескольких слов, то эти слова объединяются, и каждое слово, за исключением первого, начинается с прописной буквы. UML позволяет отображать дополнительную информацию об атрибутах. В изображении класса можно указать тип для каждого значения атрибута. Перечень возможных типов включает строку, число с плавающей точкой, целое число, логическое значение и другие перечислимые типы. Для отображения типа используется двоеточие, которое отделяет имя атрибута от его типа. Здесь же можно указать значение атрибута по умолчанию.

Операции

Операция – это то, что может выполнять класс, либо то, что вы (или другой класс) можете выполнять над данным классом. Подобно имени атрибута, имя операции записывается строчными буквами, если это одно слово. Если имя состоит из нескольких слов, они соединяются, и все слова, кроме первого, пишутся с прописной буквы. Список операций начинается ниже линии, отделяющей операции от атрибутов.

Помимо дополнительной информации об атрибутах, можно отобразить дополнительную информацию об операциях. В скобках, следующих за именем операции, можно указать параметр операции и его тип. Один из типов операций, *функция,* по окончании работы возвращает значение. В этом случае можно указать возвращаемое значение и его тип.

Для ассоциации, агрегации и композиции может указываться кратность (англ. multiplicity), характеризующая общее количество экземпляров сущностей, участвующих в отношении. Она, как правило, указывается с каждой стороны отношения около соответствующей сущности.

Кратность может указываться следующими способами:

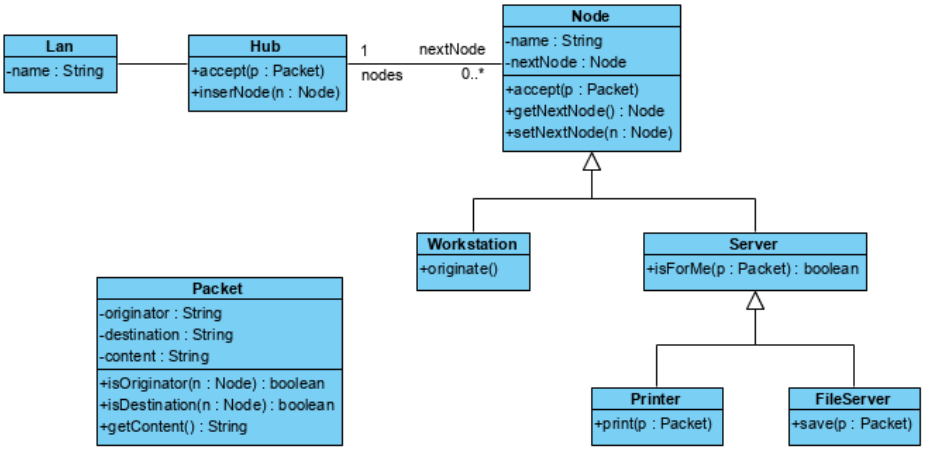
* – любое количество экземпляров, в том числе и ни одного;

целое неотрицательное число – кратность строго фиксирована и равна указанному числу (например: 1, 2 или 5);

диапазон целых неотрицательных чисел "первое число .. второе число" (например: 1..5, 2..10 или 0..5);

диапазон чисел от конкретного начального значения до произвольного конечного "первое число .. *" (например: 1..*, 5..* или 0..*);

перечисление целых неотрицательных чисел и диапазонов через запятую (например: 1, 3..5, 10, 15..*).



При разработке диаграммы следует придерживаться следующих правил:

1. За основу диаграммы классов при ее разработке берется диаграмма классов анализа.

2. Для классов должны быть определены и специфицированы все атрибуты и методы. Их спецификация, как правило, выполняется с учетом выбранного языка программирования.

3. При определении методов рекомендуется использовать сообщения с ранее разработанных диаграмм последовательности и коммуникации.

4. Детальное проектирование граничных классов, как правило, не требуется. Большинство современных средств разработки поддерживает визуальную разработку интерфейса системы – меню, диалоговых форм, элементов диалоговых окон, панелей инструментов и т. д. В качестве исходных данных для их проектирования служат прототипы пользовательских интерфейсов. В связи с этим при проектировании таких классов основное внимание следует уделять особенностям отображения информации и специфичным операциям, которые возникают при диалоге пользователя с системой. Граничные классы, определяющие интерфейс взаимодействия с другими системами, требуют детального проектирования.

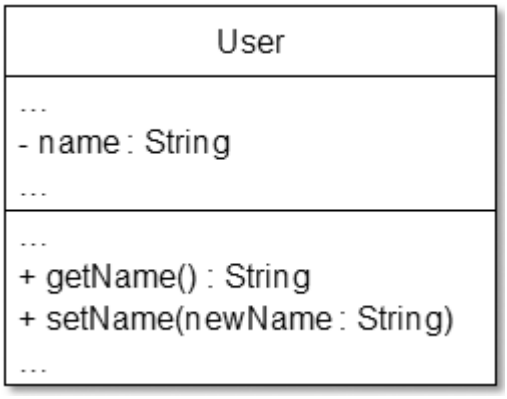
5. Для проектирования классов-сущностей можно применять подходы, используемые при проектировании БД, особенно в том случае, если данные

будут храниться в таблицах БД. Если представление данных в БД и классах отличается друг от друга и в качестве хранилища информации будет применяться реляционная база данных, то рекомендуется разработать отдельную диаграмму классов, описывающую состав и структуру БД.

6. Несмотря на то, что каждому объекту при выполнении программы автоматически назначается уникальный идентификатор, рекомендуется для классов-сущностей явно определять атрибуты, хранящие значения первичного ключа.

7. В отличие от реляционных БД поощряется использование в классах многозначных атрибутов в виде массивов, множеств, списков и т. д.

8. Управляющие классы следует проектировать только в случаях крайней необходимости – управления сложным взаимодействием объектов, реализации сложной бизнес-логики и вычислений, контроля целостности объектов и т. п. В противном случае функциональность этого класса лучше распределить между соответствующими граничными классами и классами-сущностями.



Пример спецификации закрытого атрибута и методов для работы с ним

9. Ввиду большого количества классов в системе рекомендуется диаграммы классов разрабатывать отдельно для каждого пакета.

Порядок выполнения работы:

Построить диаграмму классов рассматриваемой системы (вариант учебного проекта).

Заполнить таблицы 1,2 на основе полученной диаграммы в п.1:

Таблица 1 — Описание классов диаграммы

Название класса	**Описание**

Таблица 2 — Взаимодействие между классами

Класс	**Кратность**	**Тип отношения**	**Класс**

Содержание отчета:

1. Титульный лист.
2. Цель работы, задание (вариант индивидуального проекта).
3. Описание этапов выполнения работы.
4. Выводы о проделанной работе.