Лекция 13. Основы объектно-ориентированного представления программных систем

Учебные вопросы:

1. **Принципы объектно-ориентированного представления программных систем.**
2. **Объекты.**
3. **Классы.**

Литература: **[4], [8], [18].**

Декомпозиция программных систем

- **алгоритмическая**

В основе **алгоритмической** декомпозиции лежит разбиение по действиям – алгоритмам. Эта схема представления применяется в обычных ПС.

- **объектно-ориентированная**

Объектно-ориентированная декомпозиция обеспечивает разбиение по **автономным лицам** – объектам реального (или виртуального) мира. Эти лица (объекты) – более «крупные» элементы, каждый из них несет в себе и описания действий, и описания данных.

Абстрагирование

Создавая понятие в интересах какой-либо задачи, мы отвлекаемся (абстрагируемся) от несущественных характеристик конкретных объектов, определяя только существенные характеристики.

Абстрагирование сводится к формированию **абстракций**.

Каждая **абстракция** фиксирует основные характеристики объекта, которые отличают его от других видов объектов и обеспечивают ясные **понятийные границы**.

Абстракция концентрирует внимание на внешнем представлении объекта, позволяет отделить основное в поведении объекта от его реализации.

Абстракцию удобно строить путем выделения обязанностей объекта.

Инкапсуляция

Инкапсуляция и абстракция – взаимодополняющие понятия: абстракция выделяет внешнее поведение объекта**,** а **инкапсуляция** содержит и скрывает реализацию, которая обеспечивает это поведение.

Инкапсуляция достигается с помощью информационной закрытости. Обычно скрываются структура объектов и реализация их методов.

Модульность

Модульность – это свойство системы, которая может подвергаться декомпозиции на ряд внутренне связанных и слабо зависящих друг от друга модулей.

Общая **цель** декомпозиции на модули – уменьшение сроков разработки и стоимости ПС за счет выделения модулей, которые проектируются и изменяются независимо.

Каждая модульная структура должна быть достаточно простой, чтобы быть полностью понятой.

Изменение реализации модулей должно проводиться без знания реализации других модулей и без влияния на их поведение.

Свойства модулей

- **Информационная закрытость**
- **Связность модуля**
- **Сцепление модулей**

Информационная закрытость означает следующее:

- все модули независимы, обмениваются только информацией, необходимой для работы;
- доступ к операциям и структурам данных модуля ограничен.

Свойства модулей

- **Информационная закрытость**
- **Связность модуля**
- **Сцепление модулей**

Связность модуля – это мера зависимости его частей.

Связность – внутренняя характеристика модуля.

Чем выше связность модуля, тем лучше результат проектирования.

Измерение связности – сила связности (СС)

1.	Связность по совпадению (СС=0)	В модуле отсутствуют явно выраженные внутренние связи.
2.	Логическая связность (СС=1)	Части модуля объединены по принципу функционального подобия.
3.	Временная связность (СС=3)	Части модуля не связаны, но необходимы в один и тот же период работы системы.
4.	Процедурная связность (СС=5)	Части модуля связаны порядком выполняемых ими действий, реализующих некоторый сценарий поведения.
5.	Коммуникативная связность (СС=7)	Части модуля связаны по данным (работают с одной и той же структурой данных).
6.	Информационная (последовательная) связность (СС=9)	Выходные данные одной части используются как входные данные в другой части модуля.
7.	Функциональная связность (СС=10)	Части модуля вместе реализуют одну функцию.

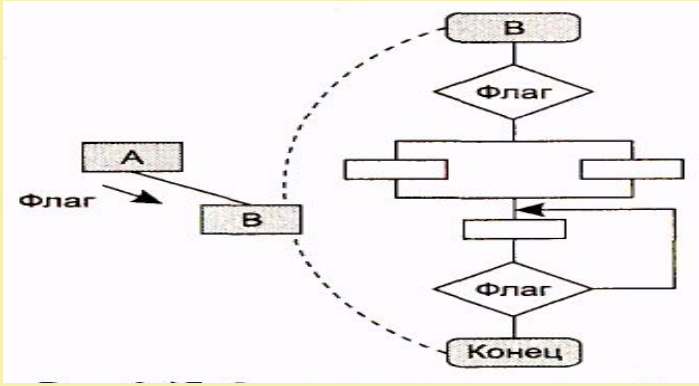
Свойства модулей

- **Информационная закрытость**
- **Связность модуля**
- **Сцепление модулей**

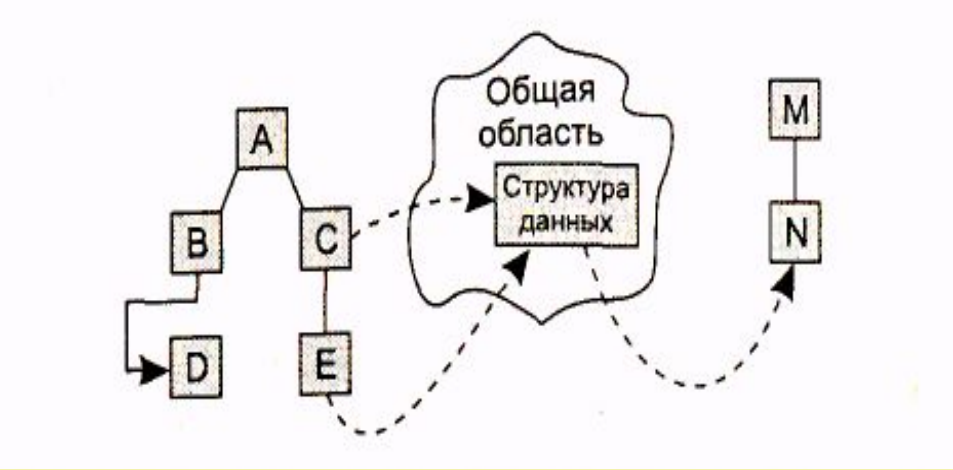
Сцепление – это мера взаимозависимости модулей по данным.

Сцепление – это внешняя характеристика модуля, которую желательно уменьшать.

Измерение сцепления – степень сцепления (СЦ)

1.	**Сцепление по данным (СЦ=1)**	**Модуль А вызывает модуль В. Все входные и выходные параметры вызываемого модуля – простые элементы данных.** 
2.	**Сцепление по образцу (СЦ=3)**	**В качестве параметров используются структуры данных.** 
3.	**Сцепление по управлению (СЦ=4)**	**Модуль А явно управляет функционированием модуля В (с помощью флагов или переключателей), посылая ему управляющие данные.** 

Измерение сцепления – степень сцепления (СЦ)

4.	**Сцепление по внешним ссылкам (СЦ=5)**	**Модули A и B ссылаются на один и тот же глобальный элемент данных.**
5.	**Сцепление по общей области (СЦ=7)**	**Модули разделяют одну и ту же глобальную структуру данных.**
6.	**Сцепление по содержанию (СЦ=9)**	**Один модуль прямо ссылается на содержание другого модуля (не через его точку входа). Например, коды их команд перемежаются друг с другом.** 

Иерархическая организация

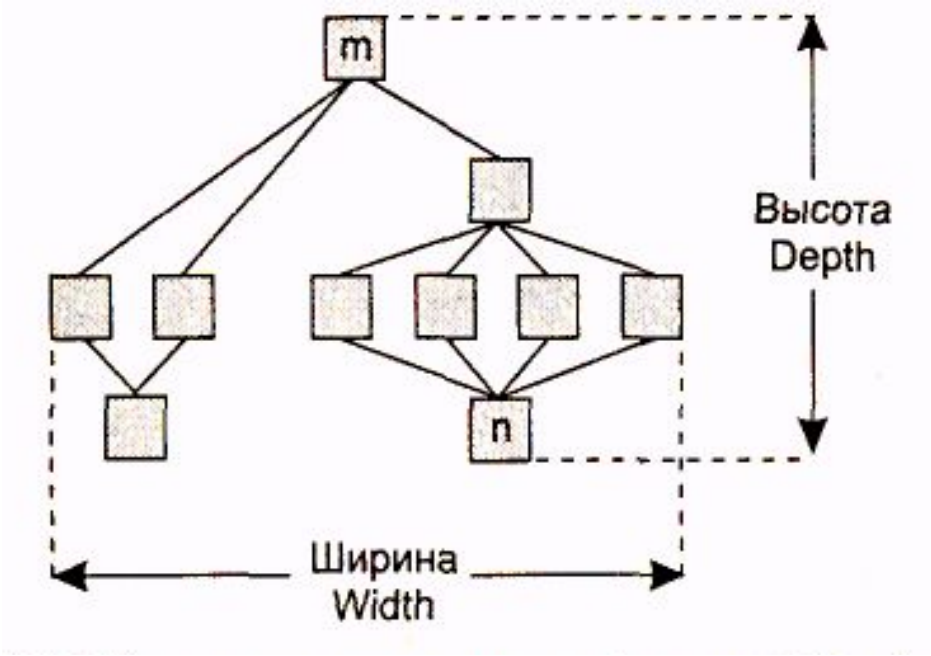
Иерархическая организация – это формирование из абстракций иерархической структуры.

Иерархическая организация задает размещение абстракций на различных уровнях описания системы.

Двумя важными инструментами иерархической организации в объектно-ориентированных системах являются:

- структура из классов («***is a***»-***иерархия***);
- структура из объектов (***«part of»-иерархия***).

Основные характеристики иерархической структуры



Иерархическая структура программной системы – это основной результат предварительного проектирования. Она определяет состав модулей ПС и управляющие отношения между модулями.

В этой структуре модуль более высокого уровня (**начальник**) управляет модулем нижнего уровня (**подчиненным**).

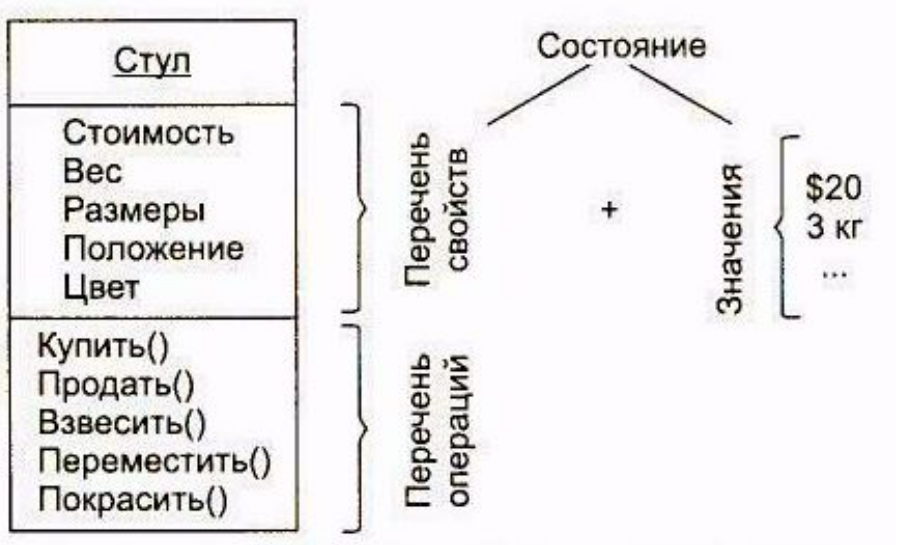
Первичные характеристики – ***количество вершин*** (модулей) и ***количество ребер*** (связей между модулями).

Глобальные характеристики – ***высота*** и ***ширина***.

Локальными характеристиками модулей структуры являются ***коэффициент объединения по входу*** **Fan_in(i)** и ***коэффициент разветвления по выходу*** **Fan_out(i)**.

Общая характеристика объектов

Объект – это конкретное представление абстракции.
Объект обладает **индивидуальностью**, **состоянием** и **поведением**. Структура и поведение подобных объектов определены в их общем ***классе***. Термины «**экземпляр класса**» и «**объект**» взаимозаменяемы.



Индивидуальность – это характеристика объекта, которая отличает его от всех других объектов.

Состояние объекта характеризуется перечнем всех свойств объекта и текущими значениями каждого из этих свойств.

Поведение характеризует то, как объект воздействует на другие объекты (или подвергается воздействию) в терминах изменений его состояния и передачи сообщений.

Виды отношений между объектами

Связь – это физическое или понятийное соединение между объектами. Объект сотрудничает с другими объектами через соединяющие их связи.

Связь обозначает соединение, с помощью которого:

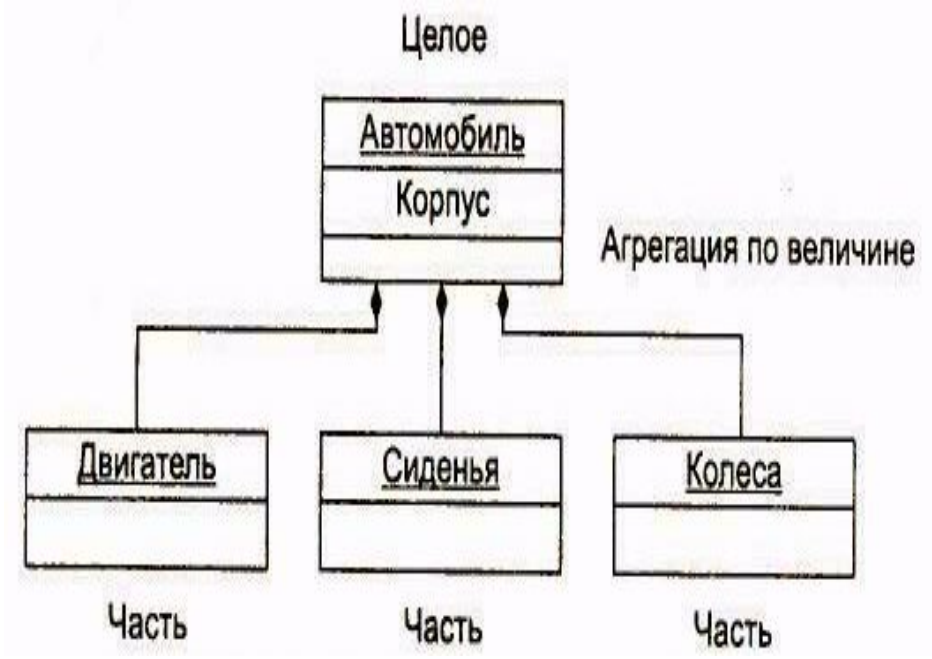
- объект-клиент вызывает операции объекта-поставщика;
- один объект перемещает данные к другому объекту.

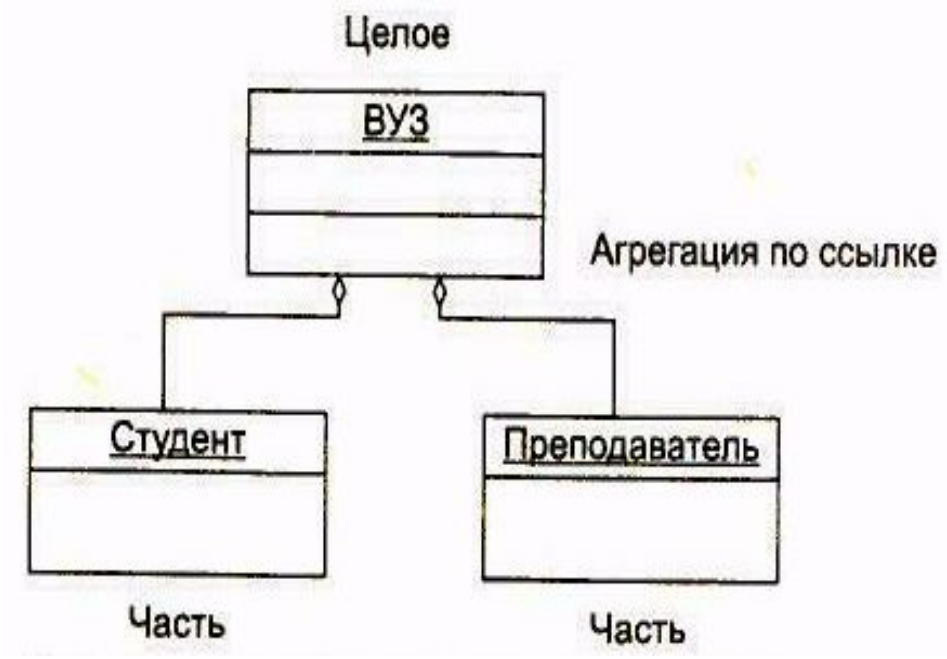
Как участник связи объект может играть одну из трех ролей:

- **актер** – объект, который может воздействовать на другие объекты, но никогда не подвержен воздействию других объектов;
- **сервер** – объект, который никогда не воздействует на другие объекты, он только используется другими объектами;
- **агент** – объект, который может как воздействовать на другие объекты, так и использоваться ими. Агент создается для выполнения работы от имени актера или другого агента.

Агрегация

- **Агрегация** обозначает отношения объектов в иерархии «целое/часть».
- **Агрегация** обеспечивает возможность перемещения от целого (агрегата) к его частям (свойствам).
- **Агрегация** может обозначать, а может и не обозначать физическое включение части в целое.





Физическое включение частей в агрегат

Нефизическое включение частей в агрегат

Общая характеристика классов

Класс – это описание множества объектов, которые разделяют одинаковые свойства, операции, отношения и семантику (смысл). Любой объект – просто экземпляр класса.

Интерфейс объявляет возможности (услуги) класса, но скрывает его структуру и поведение.

Интерфейс может быть разделен на 3 части:

1) **публичную** *(public),* объявления которой доступны всем клиентам;
2) **защищенную** *(protected),* объявления которой доступны только самому классу, его подклассам и друзьям;
3) **приватную** *(private),* объявления которой доступны только самому классу и его друзьям.

Реализация класса описывает секреты поведения класса. Она включает реализации всех операций, определенных в интерфейсе класса.

Виды отношений между классами

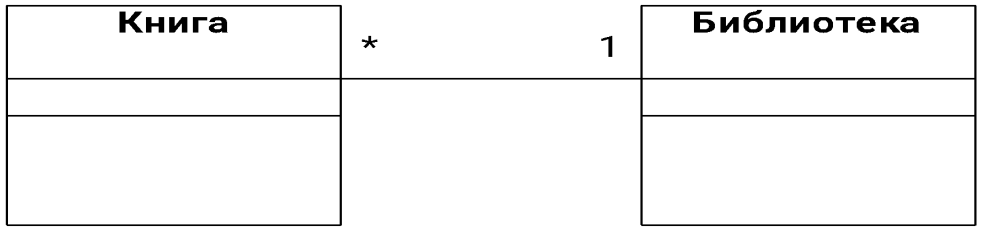
Всего существует четыре основных вида отношений между классами:

- **ассоциация** (фиксирует структурные отношения – связи между экземплярами классов);
- **зависимость** (отображает влияние одного класса на другой класс);
- **обобщение-специализация** («is a»-отношение);
- **целое-часть** («part of»-отношение).

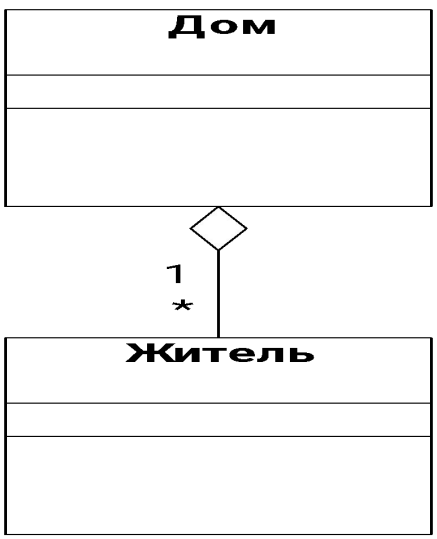
Для покрытия основных отношений большинство объектно-ориентированных языков программирования поддерживает следующие отношения:

1) **ассоциация**;
2) **наследование**;
3) **агрегация**;
4) **зависимость**;
5) **конкретизация**;
6) **метакласс**;
7) **реализация**.

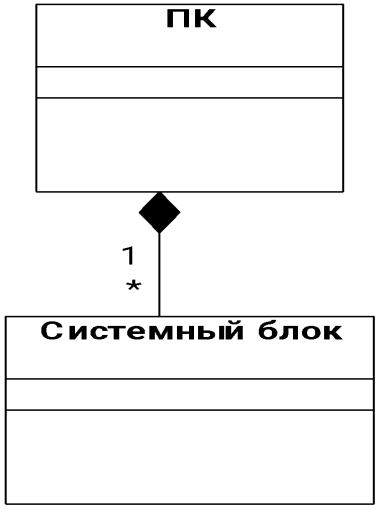
Примеры отношений между классами



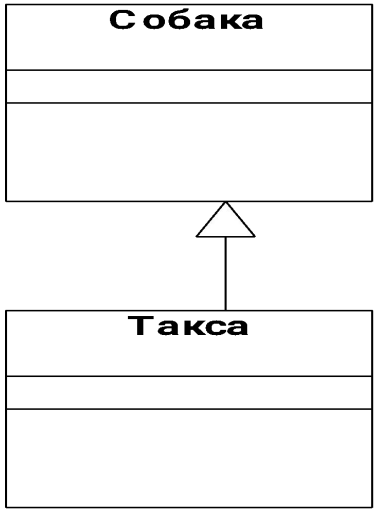
Ассоциация



Агрегация

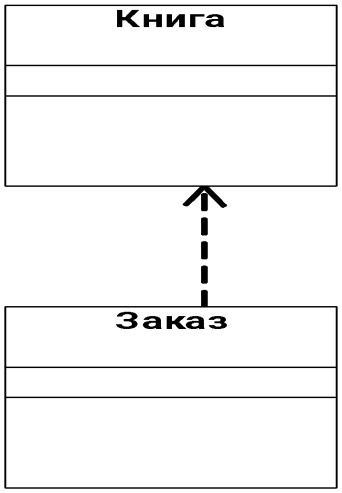


Композиция



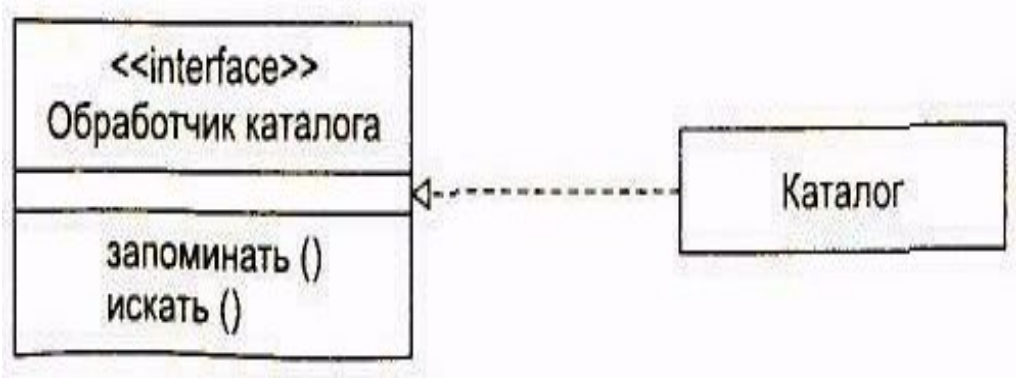
Наследование

Примеры отношений между классами



Зависимость

Конкретизация родового класса



Реализация