Программная инженерия. Методы проектирования

Классические методы проектирования

Метод структурного проектирования

Этот метод поддерживает проектирование, ориентированное на потоки данных.

Назначение метода — обеспечить систематический подход к созданию программной структуры, фундаменту архитектурного проектирования.

Результат структурного проектирования — иерархическая структура ПС.

Шаги метода:

- определение типа информационного потока;
- выделение границ потока;
- отображение диаграммы потока данных в начальную структуру системы;
- определение иерархии управления (разложением на элементы);
- уточнение полученной структуры (используются проектные эвристики, то есть продиктованные опытом рекомендации, ориентированные на повышение качества результата).

Действия на шаге 3 зависят от типа информационного потока в модели анализа.

Типы информационных потоков:

- поток преобразований;
- поток запросов.

Типы информационные потоков

- Элементы потока преобразований

- Структура потока запроса

Метод проектирования Джексона

Метод Джексона включает следующие шагов:

- Анализ: $\begin{cases} \text{Объект} - \text{действие} \\ \text{Объект} - \text{структура} \\ \text{Начальное моделирование} \end{cases}$
- Доопределение функций. Джексон выделяет три типа сервисных функций:
  - Встроенные функции;
  - Функции впечатления;
  - Функции диалога. Они решают следующие задачи:
    - наблюдают вектор состояния процесса-модели;
    - формируют и выводят поток данных, влияющий на действия в процессе-модели;
    - выполняют операции для выработки некоторых результатов.
- Учет системного времени.

Основы объектно-ориентированного представления программных систем

Принципы ОО представления ПС

Декомпозиция – разбиение на составляющие элементы.

Две схемы декомпозиции: алгоритмическая и объектно-ориентированная декомпозиции.

- **Алгоритмы**
- **Абстрагирование**
- **Инкапсуляция**
- **Модульность**

Иерархическая организация: абстракция + инкапсуляция + модульность

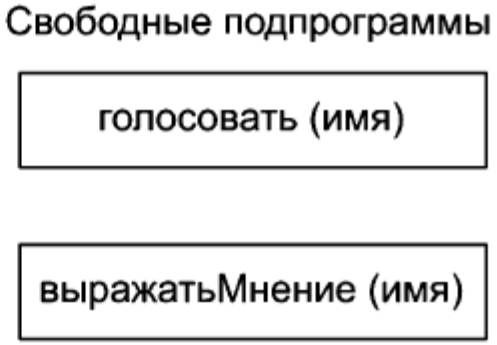
Объекты

Объект – конкретное представление абстракции.

Объект обладает индивидуальностью, состоянием и поведением.

Операции, методы и подпрограммы

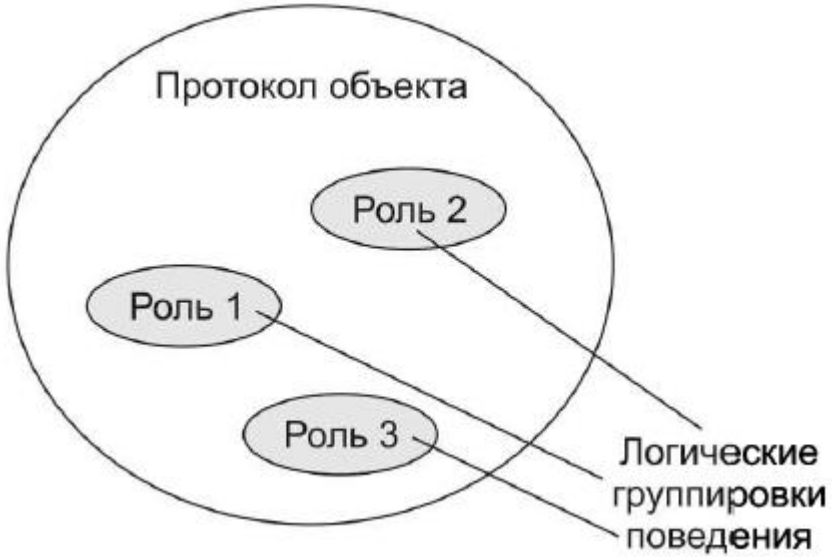
Вид операции	Пример операции
Модификатор	пополнеть (кг)
Селектор	какойВес () : integer
Итератор	показатьАссортиментТоваров () : string
Конструктор	создатьРобот (параметры)
Деструктор	уничтожитьРобот ()



Пространство поведения объекта

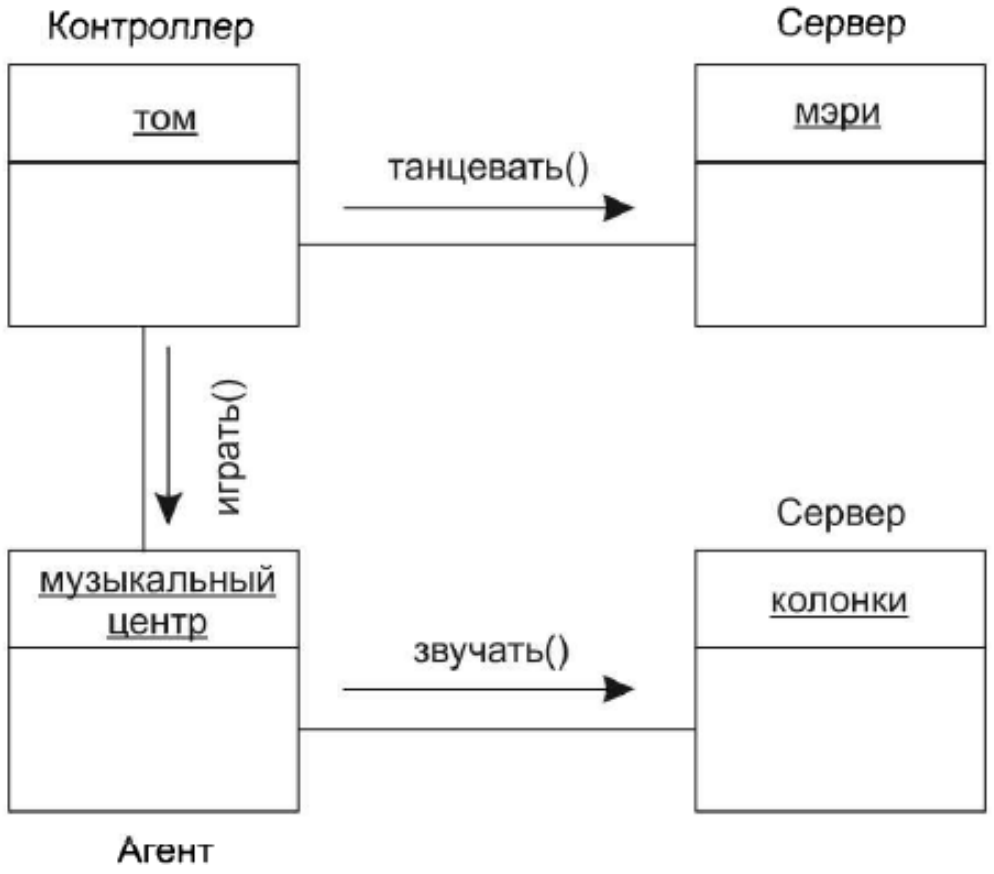
Все методы и свободные подпрограммы, ассоциированные с конечным объектом, образуют его ***протокол***.
Протокол определяет оболочку допустимого поведения объекта и поэтому заключает в себе цельное (статическое и динамическое) представление объекта.

Роли – это группировки, разделяющее пространство поведения объекта.



Виды отношений между объектами

Связь – физическое или понятийное соединение между объектами.



Классы

Класс – описание множества объектов, которые разделяют одинаковые атрибуты, операции, отношения и семантику (смысл).

Любой объект-простой экземпляр класс.

Структура представления класса

Виды отношений между классами:

- Ассоциация;
- Зависимость;
- Обобщение-специализация;
- Целое-часть.

Виды отношений в ООЯП:

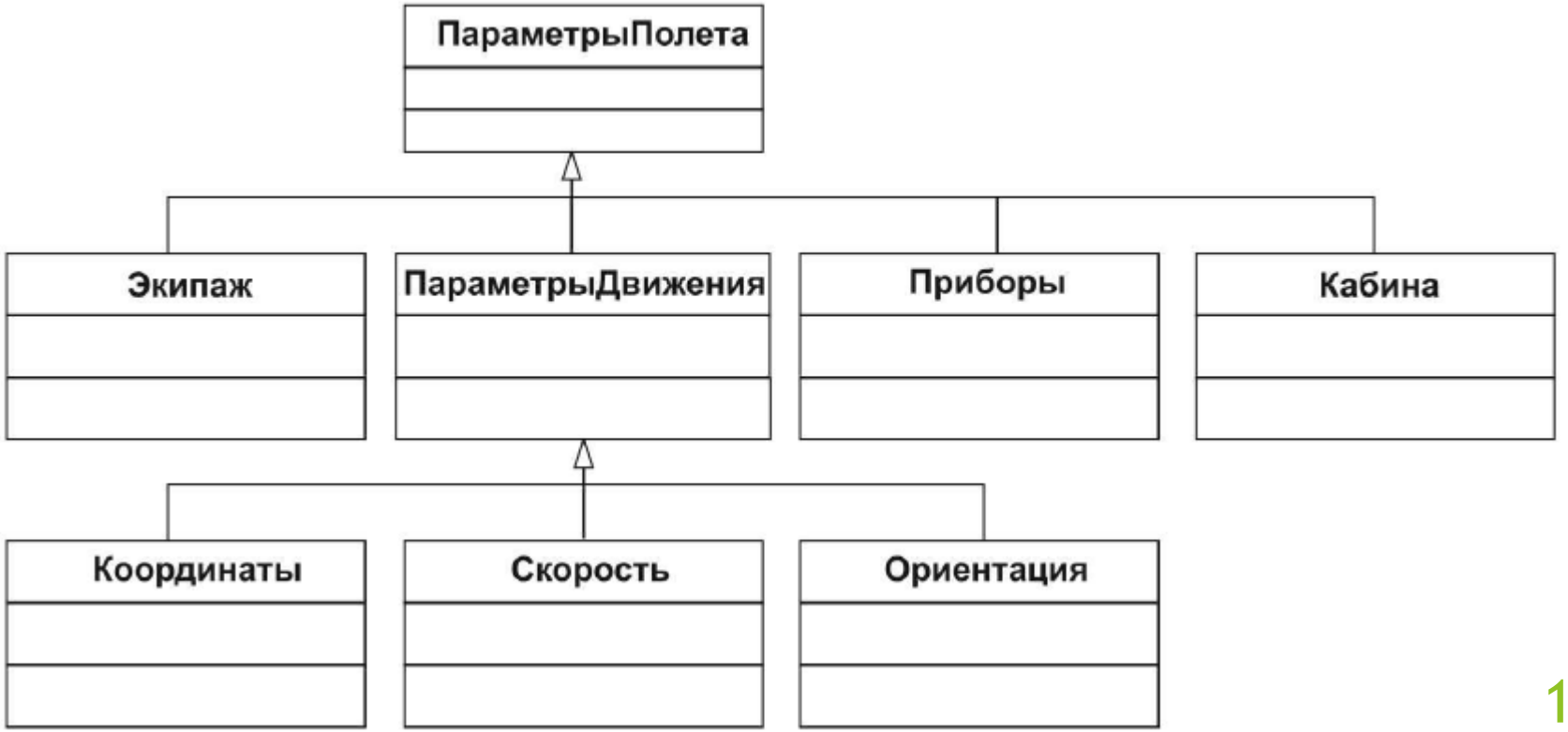
- Ассоциация;
- Наследование;
- Агрегация;
- Зависимость;
- Конкретизация4
- Метакласс;
- Реализация.

Ассоциации классов

Ассоциация обозначает семантическое соединение классов.

Наследование

Наследование - это отношение, при котором один класс разделяет структуру и поведение, определенные в одном другом (простое наследование) или во многих других (множественное наследование) классах.



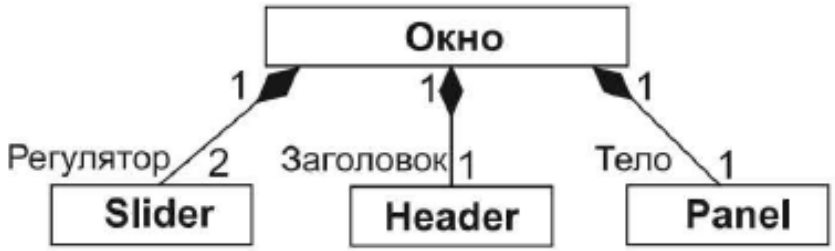
Виды отношений

Полиморфизм- возможность с помощью одного имени обозначать операции из различных классов.

Отношения ***агрегации*** между ***классами*** аналогичны отношениям агрегации между объектами.

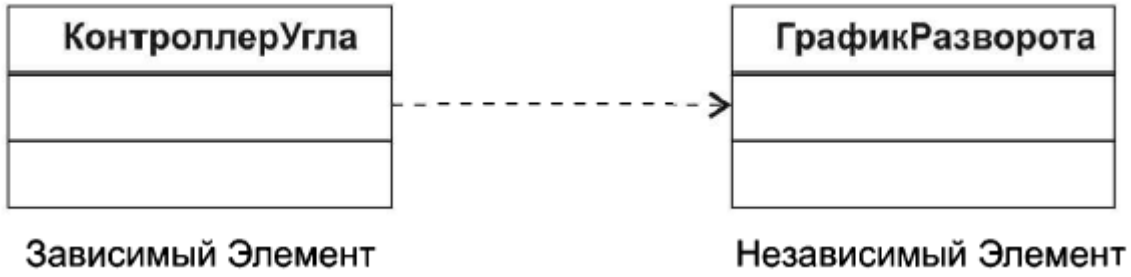
По ссылке

По величине
(композиция)

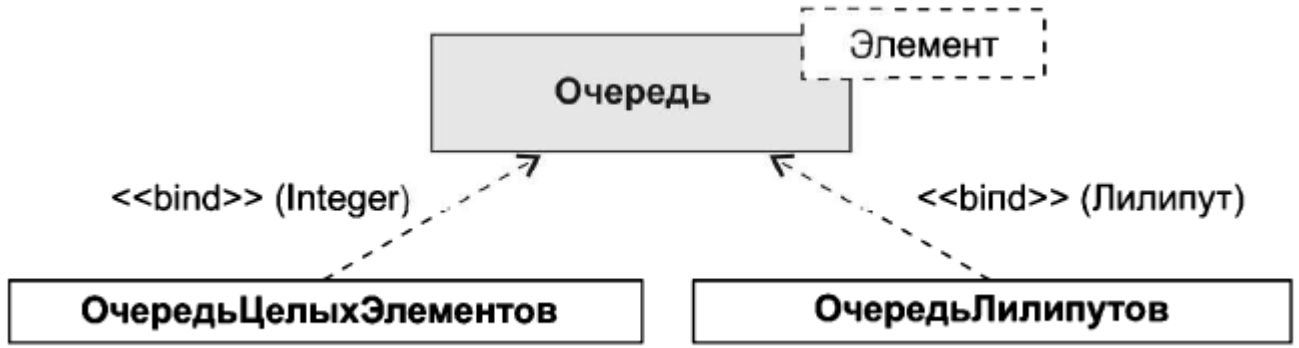


Виды отношений

Зависимость - это отношение, которое показывает, что изменение в одном классе (независимом) может влиять на другой класс (зависимый), который использует его.



Конкретизация - процесс наполнения шаблона (родового или параметризованного класса). Целью является получение класса, от которого возможно создание экземпляра.



Унифицированный язык моделирования

UML - стандартный язык для написания моделей анализа, проектирования и реализации объектно-ориентированных программных систем.

Диаграмма UML - это нагруженный связанный граф. Вершины графа нагружаются элементами модели, а дуги (ребра) – отношениями между элементами.

Структурные диаграммы:

1. Архитектурный уровень
    1. Диаграмма пакетов
    2. Диаграмма компонентов
2. Уровень детального проектирования
    1. Диаграмма классов
    2. Диаграмма объектов
    3. Диаграмма композитной структуры
3. Уровень физического размещения
    1. Диаграмма развертывания

Диаграммы поведения:

1. Формирование требований
    1. Диаграмма Use Case
    2. Диаграмма деятельности
2. Анализ требований
    1. Диаграмма последовательности
    2. Диаграмма коммуникации
    3. Диаграмма обзора взаимодействий
    4. Диаграмма синхронизации
3. Детальное проектирование
    1. Диаграмма конечного автомата

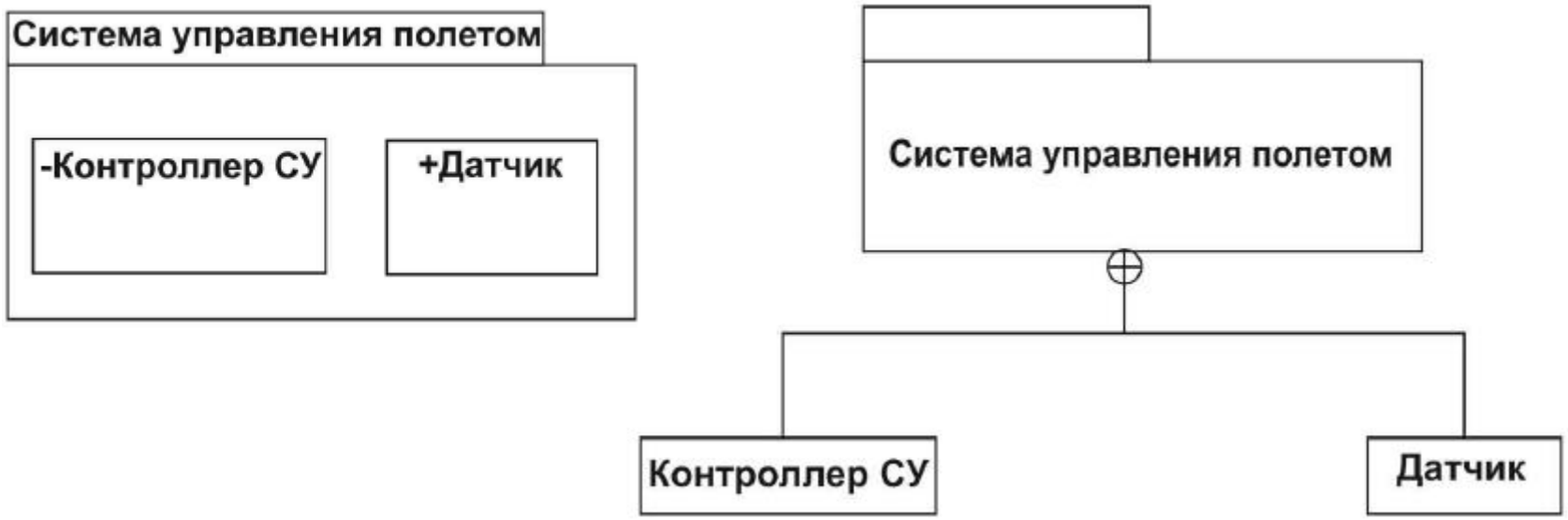
Объектно-ориентированная проектирование и реализация

Основные понятия

Формирование архитектуры - первый и основополагающий шаг в решении задачи проектирования, закладывающий фундамент представления программной системы, способной выполнять весь спектр детальных требований.

Диаграмма пакетов- это структурная диаграмма, в которой основными элементами являются пакеты и зависимости между ними.

Пакет- хранилище элементов, изображается в виде прямоугольника с закладкой в одном из углов (обычно в левом верхнем).



Основные понятия

Диаграммы компонентов показывает определения, внутреннюю структуру и зависимости набора компонентов.

Компонент- модульная и заменяемая часть системы, которая соответствует набору интерфейсов и обеспечивает реализацию этого набора интерфейсов.

Интерфейс- список операций, которые определяют услуги компонента (или класса).

Интерфейсы бывают ***обеспеченными*** (описывает услуги, исполнение которых компонент предлагает другим компонентам) и ***требуемыми*** (описывает услуги, которые поставляются другими компонентами).

Порт является окном в капсулу компонента. Через это окно происходит взаимодействие внешней среды с закрытым содержимым компонента.

Детальное проектирование

Детальное проектирование - вторая ступень проектирования, которая следует за созданием архитектуры.

Формирование требований => Анализ требований => Детальное проекирование

Диаграммы классов

Класс = имя +

+атрибуты +

Видимость Имя: Тип [Множественность] = НачальноеЗначение {Свойства}

+ операции

Видимость Имя (Список параметров): ВозвращаемыйТип {Свойства}

СПАСИБО ЗА ВНИМАНИЕ!

Если остались вопросы, задавайте их в ЭИОС МТУСИ или пишите на почту