Нижегородский государственный университет им. Н.И. Лобачевского

Проектирование подсистем. Основные принципы объектного подхода. Классы, объекты и их взаимодействие на примере

При поддержке компании Intel

Кустикова В.Д., кафедра математического обеспечения ЭВМ, факультет ВМК

Содержание

- Объектно-ориентированная разработка подсистем: анализ, проектирование и программирование
- Основные принципы объектного подхода: абстракция, инкапсуляция, наследование, полиморфизм
- Объекты и классы, природа и отношения между ними. Качество объектов и классов
- Признаки испорченной архитектуры
- Принципы объектно-ориентированного проектирования классов:
  - Принцип единственной ответственности (The Single Responsibility Principle, **S**RP)
  - Принцип открытия/закрытия (The Open Closed Principle, **O**CP)
  - Принцип подстановки Барбары Лисков (The Liskov Substitution Principle, **L**SP)
  - Принцип отделения интерфейса (The Interface Segregation Principle, **I**SP)
  - Принцип инверсии зависимости (The Dependency Inversion Principle **D**IP)

Цели

- Рассмотреть составляющие объектно-ориентированной разработки подсистем, изучить основные принципы объектного подхода, ввести понятия объекта и класса.
- Пояснить основные признаки испорченной архитектуры. Изучить принципы объектно-ориентированного проектирования классов (SOLID-принципы).

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ РАЗРАБОТКА

Объектно-ориентированная разработка подсистем…

- ***Объектно-ориентированное программирование (Object-oriented programming, OOP)*** – методология программирования, основанная на представлении программы в виде набора объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

- Ключевые моменты определения:
  - Базовые элементы – объекты, а не алгоритмы.
  - Каждый объект – экземпляр какого-либо класса.
  - Классы организованы иерархически.

Объектно-ориентированная разработка подсистем…

- ***Объектно-ориентированное проектирование (Object-oriented design, OOD)*** – методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

- Ключевые моменты определения:
  - Основа – объектно-ориентированная декомпозиция.
  - Применение приемов представления моделей, отражающих логическую (классы и объекты) и физическую (модули и процессы) структуру системы, а также ее статические и динамические аспекты.

Объектно-ориентированная разработка подсистем…

- ***Объектно-ориентированный анализ (Object-oriented analysis, OOA)*** – методология, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области.

- OOA обеспечивает модели, на основании которых разрабатывается архитектура с использованием OOD.
- OOD обеспечивает основание для реализации с использованием OOP.

ПРИНЦИПЫ ОБЪЕКТНОГО ПОДХОДА

Основные принципы объектного подхода. Абстракция…

- ***Абстракция*** выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.

- ***Клиентом*** называется любой объект, использующий ресурсы другого объекта (называемого ***сервером***).
- Поведение объекта характеризуется ***услугами***, которые он предоставляет другим объектам, и ***операциями***, которые он выполняет над другими объектами.
- Такой подход приводит к идее ***контрактного программирования***.

Основные принципы объектного подхода. Абстракция…

- ***Контрактное программирование:***
  - ***Контракт*** фиксирует все обязательства, которые объект-сервер имеет перед объектом-клиентом, определяет ответственность объекта – поведение, за которое он отвечает.
  - ***Операция***, предусмотренная контрактом, однозначно определяется ее формальными параметрами и типом возвращаемого значения.
  - Полный набор операций, которые клиент может осуществлять над другим объектом, вместе с правильным порядком, в котором эти операции вызываются, называется ***протоколом***. Протокол определяет поведение абстракции.

Основные принципы объектного подхода. Абстракция

- ***Примеры абстракций:***
  - Аэропорт
  - Самолет
  - Рейс
  - Роль пользователя системы

Основные принципы объектного подхода. Инкапсуляция…

- ***Инкапсуляция*** – процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение. Инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

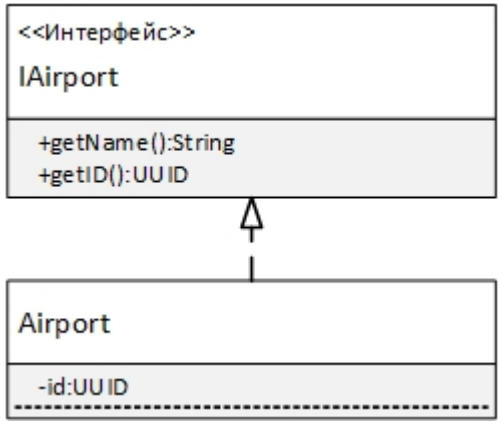
- Инкапсуляция проявляется в скрытии информации – внутренних деталей, не влияющих на внешнее поведение.
- Скрывается и внутренняя структура объекта, и реализация его методов.

Основные принципы объектного подхода. Инкапсуляция…

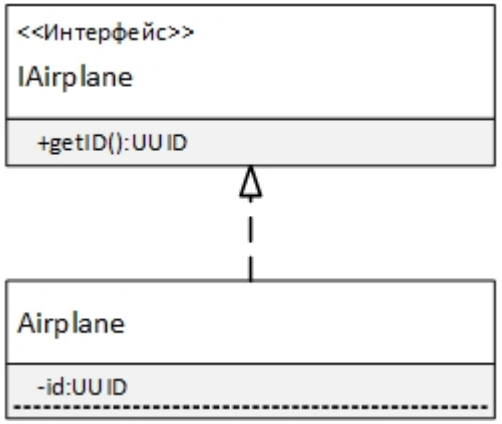
- ***Принцип разделения интерфейса и реализации:***
  - ***Интерфейс*** отражает внешнее поведение объекта, описывая абстракцию поведения всех объектов данного класса. Интерфейсная часть содержит все, что касается взаимодействия данного объекта с любыми другими объектами.
  - Внутренняя ***реализация*** описывает представление абстракции и механизмы достижения желаемого поведения объекта.

Основные принципы объектного подхода. Инкапсуляция

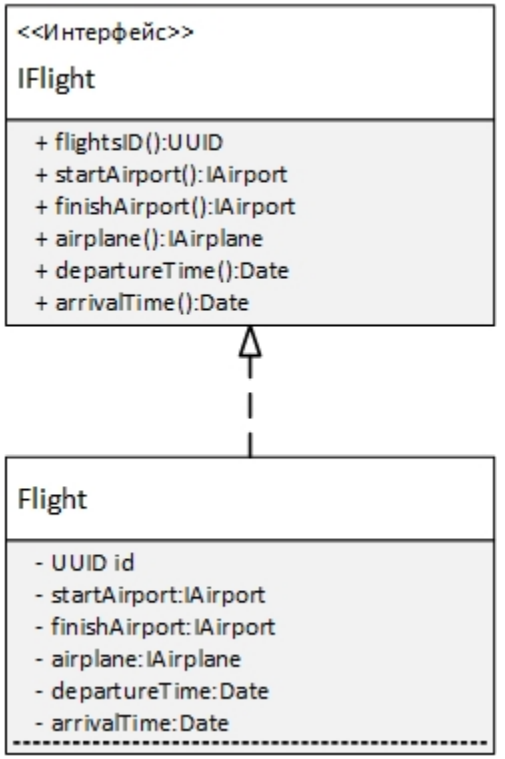
- ***Примеры проявления инкапсуляции:***



Интерфейс и класс «аэропорт»



Интерфейс и класс «самолет»



Интерфейс и класс «рейс»

Основные принципы объектного подхода. Наследование…

- ***Наследование*** – отношение между классами, при котором класс использует структуру или поведение другого (одиночное наследование) или других (множественное наследование) классов.

- Наследование вводит иерархию «общее/частное» в которой подкласс наследует структуру и поведение от одного или нескольких более общих суперклассов.
- Подклассы обычно дополняют или переопределяют унаследованную структуру и поведение.

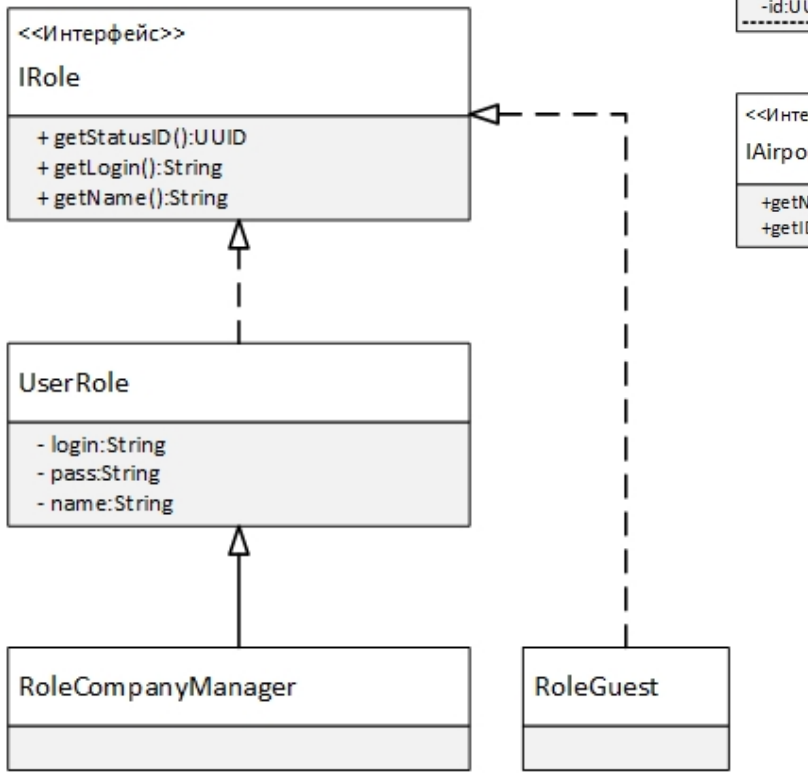
Основные принципы объектного подхода. Наследование

- ***Примеры иерархий:***

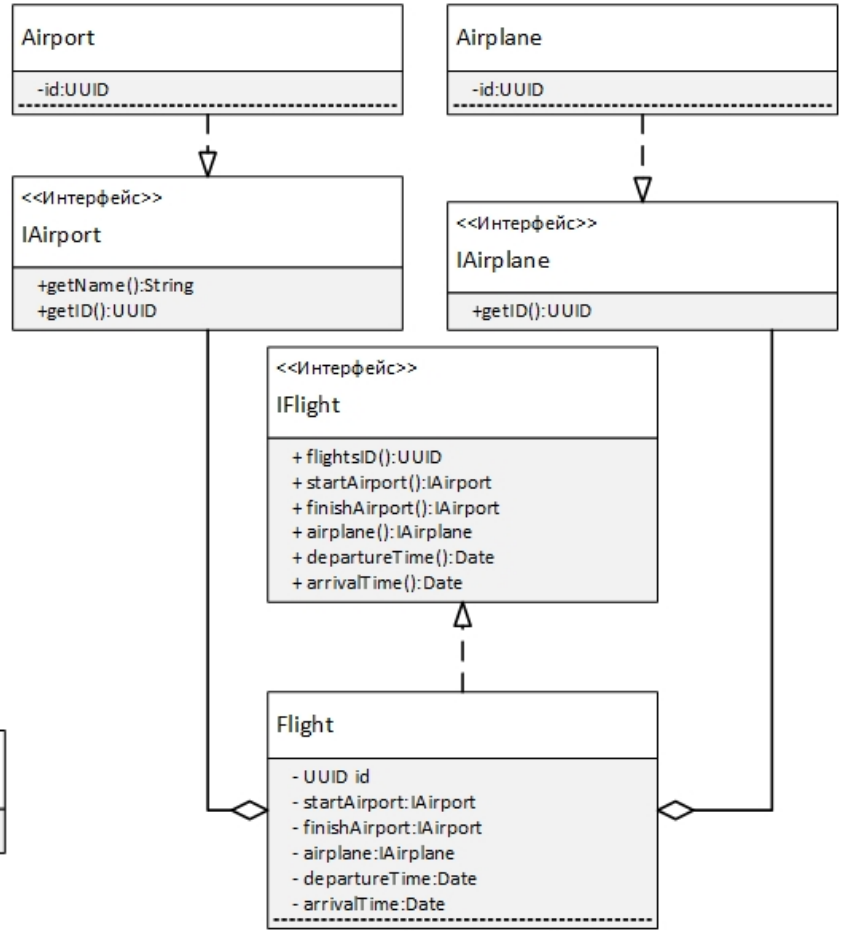
Интерфейс роли пользователя

Класс обычных пользователей

Класс менеджеров компании



Класс гостевой роли



Одиночное наследование

Агрегация

Основные принципы объектного подхода. Полиморфизм…

- ***Полиморфизм*** – положение, согласно которому имена (например, переменных) могут обозначать объекты разных (но имеющих общего родителя) классов. Следовательно, любой объект, обозначаемый полиморфным именем, может по-своему реагировать на некий общий набор операций.

- Динамический полиморфизм. Проявляется при взаимодействии наследования и механизма позднего связывания (или динамического связывания).
- Статический полиморфизм. Проявляется при использовании параметров-типов в шаблонных функциях и классах.

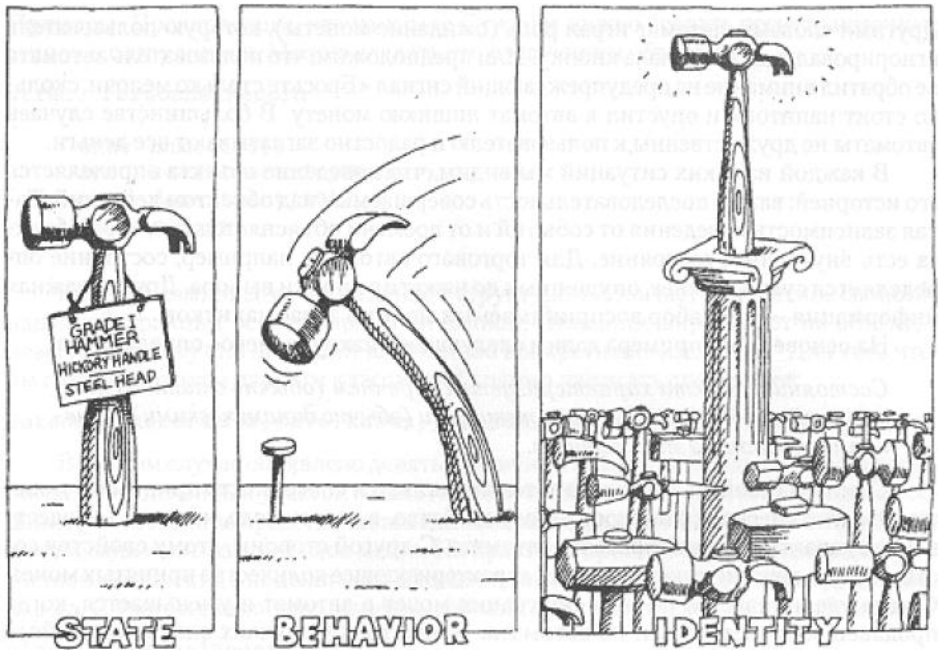
ОБЪЕКТЫ И КЛАССЫ

Объекты и классы. Природа объекта…

- С точки зрения восприятия человеком объектом может являться:
  - Осязаемый и/или видимый объект.
  - Нечто, воспринимаемое мышлением.
  - Что-то, на что направлена мысль или действие.
- Объект моделирует часть окружающей действительности, существует в пространстве и времени.
- ***Объект*** представляет собой конкретный опознаваемый предмет, единицу или сущность (реальную или абстрактную). Имеет четко определенное функциональное назначение в данной предметной области.

Объекты и классы. Природа объекта…

- ***Объект*** обладает состоянием, поведением и идентичностью. Структура и поведение схожих объектов определяет общий для них класс. Термины «экземпляр класса» и «объект» взаимозаменяемы.



**Буч Г. Объектно-ориентированный анализ и проектирование с примерами на C++. 2000.*

Объекты и классы. Природа объекта…

- ***Состояние*** объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из свойств.
- ***Поведение*** – это то, как объект действует и реагирует.
  - Поведение выражается в терминах состояния объекта и передачи сообщений.
  - В основном понятие «сообщение» совпадает с понятием «операция над объектом», хотя механизм передачи различен.
  - Состояние объекта представляет суммарный результат его поведения.
- ***Идентичность*** – свойство объекта, которое отличает его от всех других объектов.

Объекты и классы. Природа объекта

- ***Примеры объектов:***
  - Конкретный аэропорт
  - Конкретный самолет
  - Конкретный вылет
  - Роль конкретного пользователя, вошедшего в систему

Объекты и классы. Отношения между объектами…

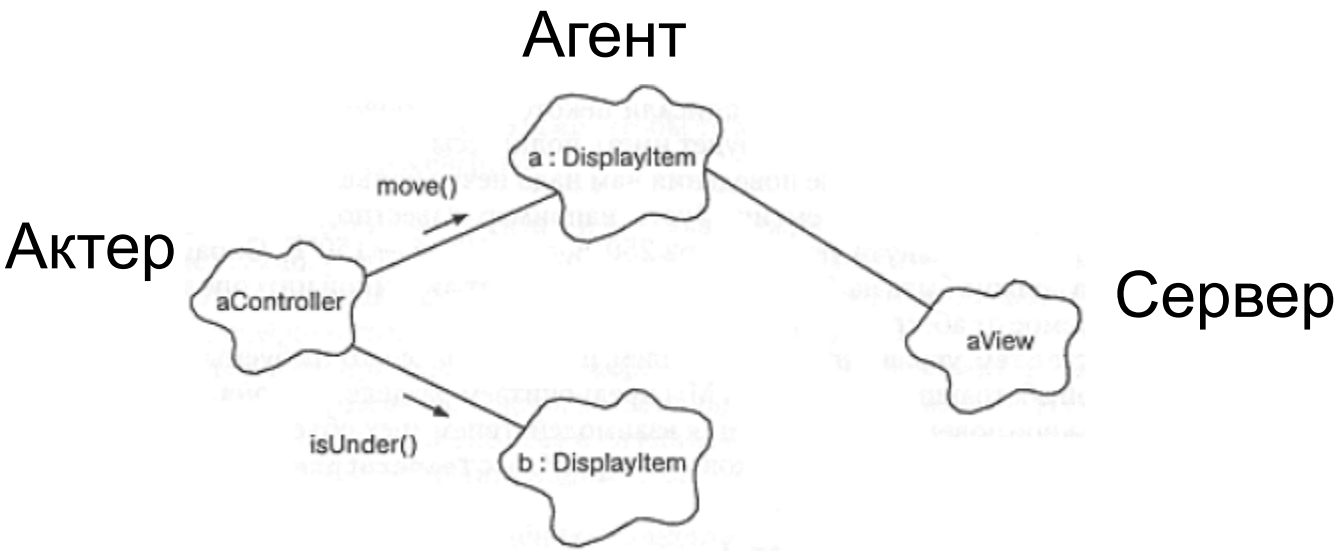
- Типы отношений между объектами:
  - ***Агрегация*** описывает отношения целого и части, приводящие к соответствующей иерархии объектов, причем, идя от целого (агрегата), можно прийти к его частям (атрибутам).
    - Агрегация = физическое вхождение (например, крылья являются частью самолета).
    - Агрегация = концептуальное вхождение (например, акционер монопольно владеет акциями, но они физически в него не входят).
  - ***Связи*** – специфическое сопоставление, через которое клиент запрашивает услугу у объекта-сервера или через которое один объект находит путь к другому.

Объекты и классы. Отношения между объектами…

- Роли объектов при реализации отношения типа «связь»:
  - ***Актер***. Объект может воздействовать на другие объекты, но сам никогда не подвергается воздействию других объектов.
  - ***Сервер***. Объект может только подвергаться воздействию со стороны других объектов, но он никогда не выступает в роли воздействующего объекта.
  - ***Агент***. Такой объект может выступать как в активной, так и в пассивной роли. Как правило, объект-агент создается для выполнения операций в интересах какого-либо объекта-актера или агента.

Объекты и классы. Отношения между объектами…

- ***Пример распределения ролей:***



**Буч Г. Объектно-ориентированный анализ и проектирование с примерами на C++. 2000.*

Объекты и классы. Природа классов

- ***Класс*** – некоторое множество объектов, имеющих общую структуру и общее поведение.
- Любой конкретный объект является ***экземпляром класса***.

- Объекты, не связанные общностью структуры и поведения, нельзя объединить в класс, так как по определению они не связаны между собой ничем, кроме того, что все они объекты.
- Классы необходимы, но не достаточны для декомпозиции сложных систем. Некоторые абстракции так сложны, что не могут быть выражены в терминах описания класса (абстракции GUI, база данных – объекты, но не классы).

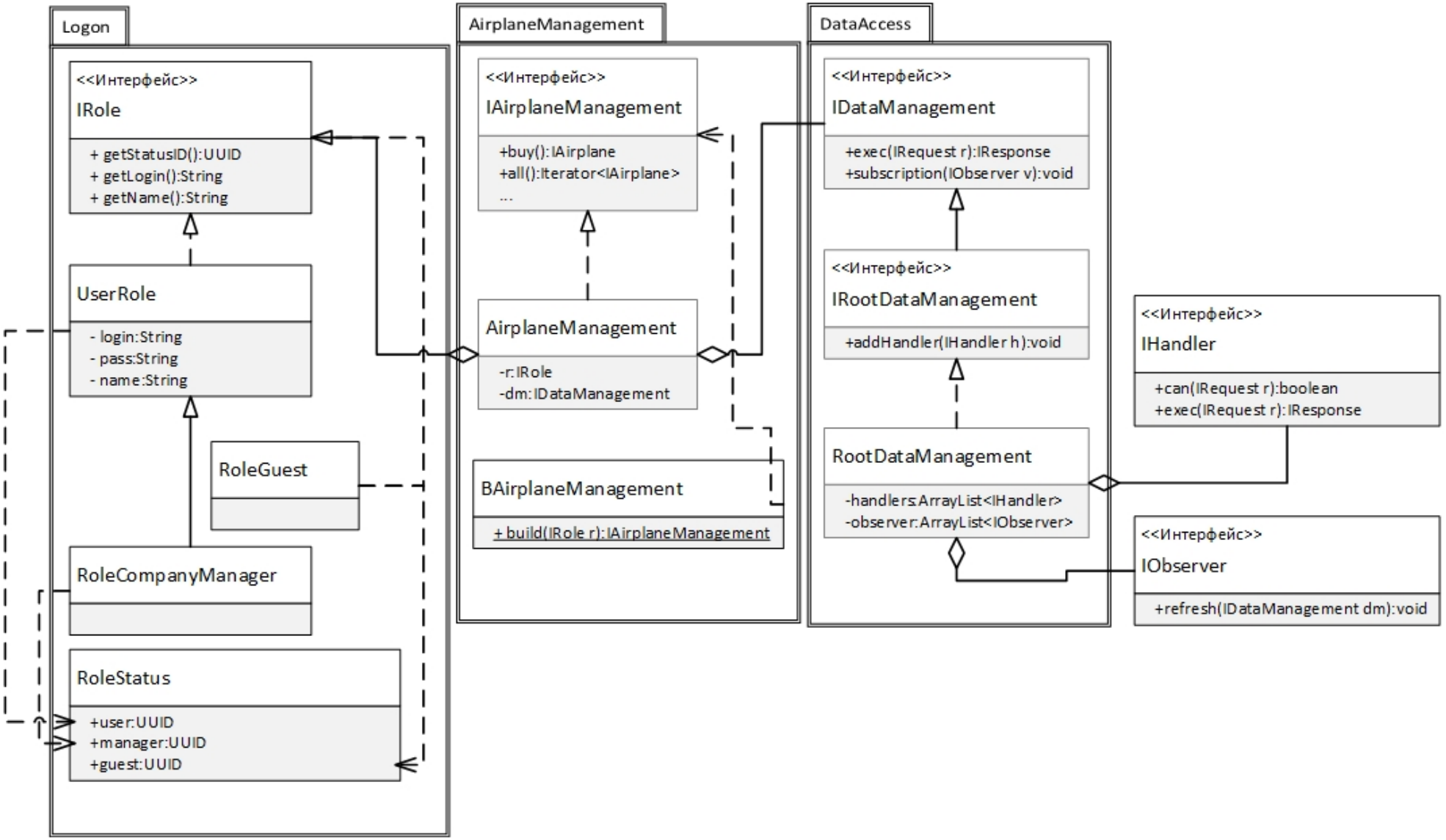
Объекты и классы. Отношения между классами…

- Типы отношений между классами:
  - Отношение «обобщение/специализация» (общее и частное, «is-a»).
  - Отношение «целое/часть» («part of»).
  - Семантические, смысловые отношения, ассоциации.
- Подходы к выражению отношений между классами в языках программирования:
  - Ассоциация – смысловая связь (один-к-одному, один-ко-многим или многие-ко-многим). По умолчанию, не имеет направления (если не оговорено противное, ассоциация подразумевает двустороннюю связь) и не объясняет, как классы общаются друг с другом. Требуется на ранней стадии анализа.

Объекты и классы. Отношения между классами

- Наследование – отношение общего и частного.
  - Альтернатива наследованию – делегирование, объекты рассматриваются как прототипы, которые делегируют свое поведение родственным им объектам.
- Агрегация – отношение «целое/часть» (например, физическое включение), направленное отношение.
- Использование – наличие равноправной связи между экземплярами классов.
- Инстанцирование. Введение параметризованных классов (шаблонные классы).
- Метакласс – класс, экземпляры которого суть классы. Возможность изменять следование классов, обобщенные функции и методы.

Объекты и классы. Отношения между классами. Примеры



Объекты и классы. Отношение между классами и объектами

- Каждый объект представляет собой экземпляр класса.
- В программе может создаваться любое число объектов определенного класса.

- В большинстве случаев классы статичны, т.е. их содержание определено на этапе компиляции.
- Сами объекты динамичны, т.к. создаются и удаляются в процессе исполнения программы.

Объекты и классы. Качество классов и объектов

- Критерии оценки качества классов и объектов системы:
  - Зацепление – степень глубины (количество) связей между отдельными модулями.
  - Связность – степень взаимодействия между элементами отдельного модуля (а для OOD еще и отдельного класса или объекта).
  - Достаточность – наличие в классе или модуле всего необходимого для реализации логичного и эффективного поведения.
  - Полнота – наличие в интерфейсной части класса всех необходимых характеристик абстракции.
  - Примитивность операций. Операции, которые требуют доступа только к внутренней реализации абстракции.

Объекты и классы. Выбор операций…

- ***Функциональность***
  - Первая версия интерфейса класса создается, исходя из структурного смысла класса. Когда появляются клиенты класса, интерфейс уточняется, модифицируется и дополняется.
  - В каждом классе принято выделять только примитивные операции, отражающие отдельные аспекты поведения.
  - Следует разделять методы, не связанные между собой.
  - Методы класса необходимо рассматривать как единое целое, т.к. они взаимодействуют друг с другом для реализации протокола абстракции.

Объекты и классы. Выбор операций

- ***Расходы памяти и времени***
  - Для каждого метода необходимо принять решение об использовании памяти и времени.
  - Использование понятий «лучшего», «среднего» и «худшего» (верхний допустимый предел) вариантов.

Объекты и классы. Выбор отношений…

- ***Сотрудничество***
  - Отношения связаны с конкретными действиями.
  - Одна абстракция должна видеть другую и иметь доступ к ее открытым ресурсам.
  - Абстракции доступны одна другой только тогда, когда перекрываются их области видимости и даны необходимые права доступа.
  - Закон Деметера: «Методы любого класса не должны зависеть от структуры других классов, а только от структуры (верхнего уровня) самого класса. В каждом методе посылаются сообщения только объектам из предельно ограниченного множества классов».

Объекты и классы. Выбор отношений

- ***Механизмы и видимость***
  - Отношения между объектами определяется в основном механизмами их взаимодействия (кто и о чем должен знать?).
  - Способы, обеспечивающие видимость объектов:
    - Сервер является глобальным.
    - Сервер передается клиенту в качестве параметра операции.
    - Сервер является частью клиента в смысле классов.
    - Сервер локально объявляется в области видимости клиента.

Объекты и классы. Выбор реализации…

- ***Выбор способа представления класса или объекта***
  - Представление классов должно быть скрыто. Обеспечивает возможность изменения реализаций без нарушения связей с другими классами и объектами.
  - Наличие нефункциональных требований (например, скорость выполнения определенных операций или объем памяти для хранения представления) приводит к необходимости создания семейства классов с одинаковым интерфейсом, но принципиально разной реализацией.

Объекты и классы. Выбор реализации

- ***Выбор способа размещения класса или объекта в модуле***
  - Решение о месте декларирования классов и объектов в языках является компромиссом между требованиями видимости и скрытия информации.
  - В общем случае модули должны быть функционально связными внутри и слабо связанными друг с другом.
  - Необходимо учитывать такие факторы, как повторное использование, безопасность, документирование.

SOLID-принципы

ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ КЛАССОВ

Признаки неудачной архитектуры

- ***Отсутствие гибкости***, сложность модификации. Модификация влечет каскад последовательных изменений в зависимых модулях.
- ***Хрупкость*** – падение системы во многих местах при каждом небольшом изменении.
- ***Неподвижность*** – отсутствие возможности повторного использования подсистем или компонент при разработке подобных проектов.
- ***Вязкость*** проявляется в двух формах:
  - Вязкость архитектуры наблюдается, когда решение проблемы требует модификации архитектуры системы.
  - Вязкость окружения проявляется при использовании медленной и неэффективной среды разработки.

Принципы объектно-ориентированного проектирования классов

- Принцип единственной ответственности (The Single Responsibility Principle, **S**RP)
- Принцип открытия/закрытия (The Open Closed Principle, **O**CP)
- Принцип подстановки Барбары Лисков (The Liskov Substitution Principle, **L**SP)
- Принцип отделения интерфейса (The Interface Segregation Principle, **I**SP)
- Принцип инверсии зависимости (The Dependency Inversion Principle **D**IP)

Принцип единственной ответственности…

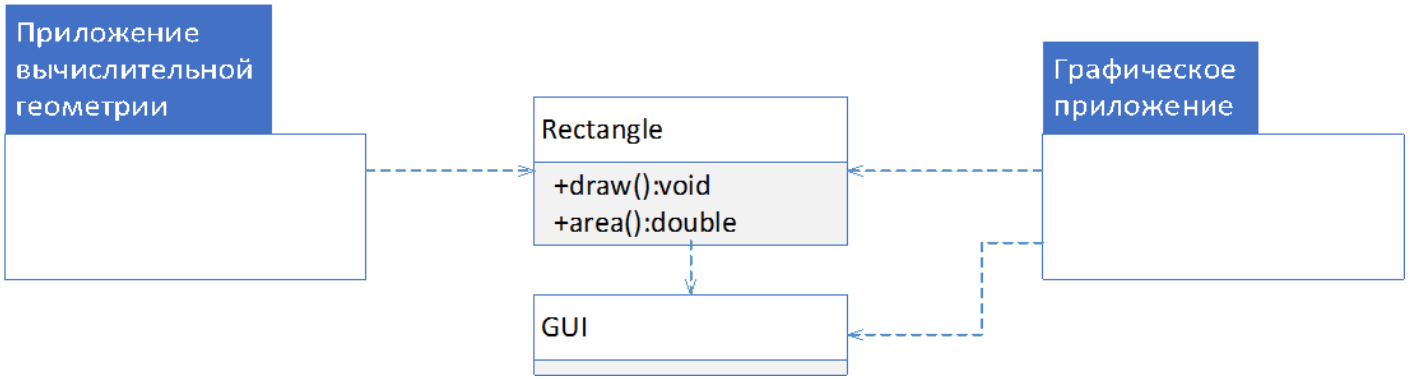
- ***Должна существовать только одна причина для изменения класса.***

- Ответственность = причина для изменения.

- Если имеется две причины, чтобы внести изменения в класс, то необходимо разделить функциональность на два класса, и после этого модифицировать.
- В результате каждый класс будет содержать только одну ответственность, как следствие, одну причину для изменения.

Принцип единственной ответственности…

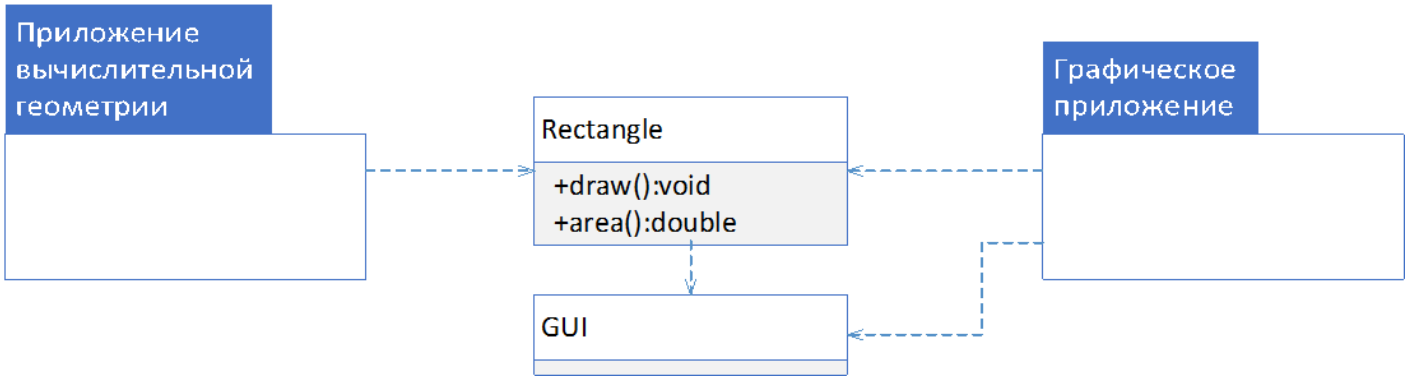
- Типичный пример нескольких ответственностей:



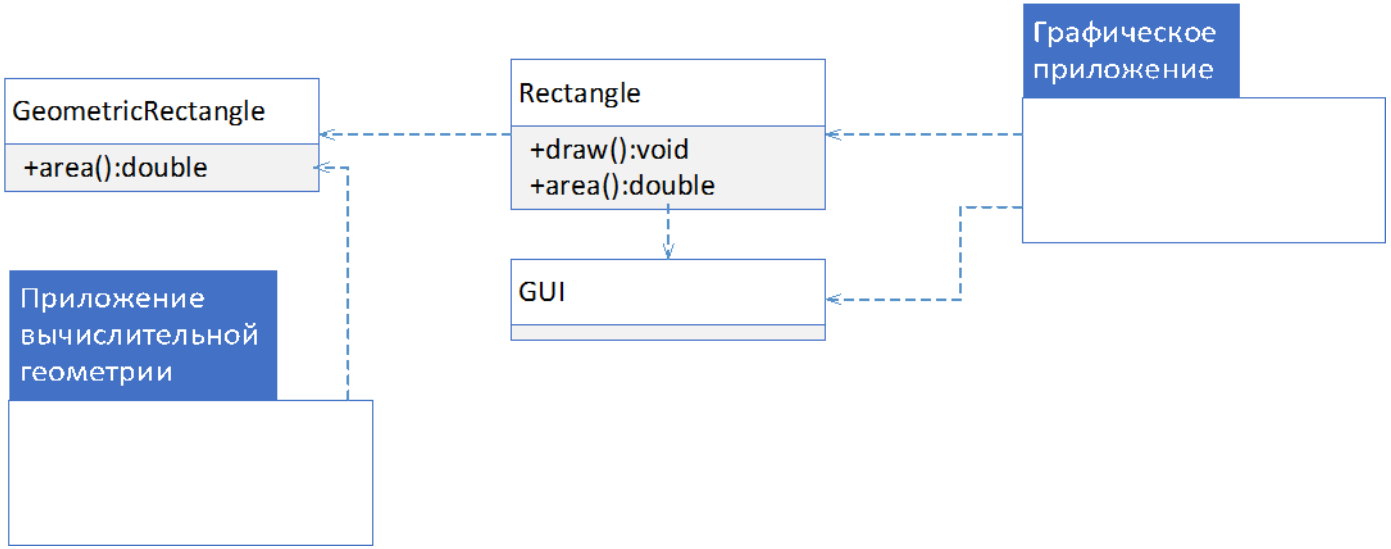
- Класс Rectangle содержит две ответственности:
  - Математическая модель геометрии прямоугольника.
  - Отображение прямоугольника.
- Проблемы:
  - Необходимо включать GUI в приложение вычислительной геометрии.
  - Изменения в графической части влекут необходимость повторной сборки и развертывания приложения вычислительной геометрии.

Принцип единственной ответственности…

- Типичный пример нескольких ответственностей:



- Решение:



Принцип открытия/закрытия…

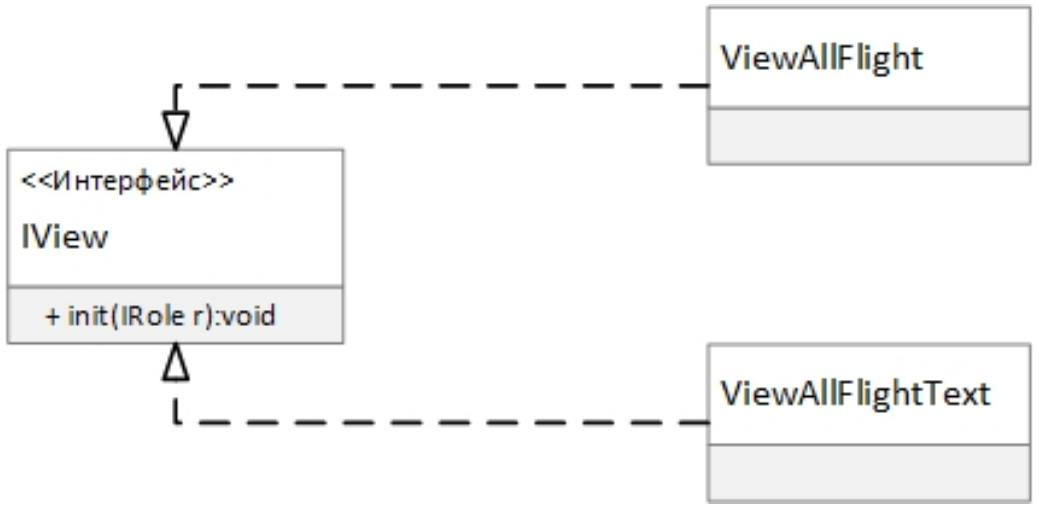
- ***Модуль должен быть открыт для расширения, но закрыт для модификации.***

- Техники, позволяющие следовать принципу:
  - Динамический полиморфизм (наследование + механизм позднего связывания). Определение интерфейса, несколько реализаций. Расширение за счет разработки новых реализаций.
  - Статический полиморфизм. Использование шаблонов функций и классов. Расширение с помощью параметров-типов.

Принцип открытия/закрытия

- Пример проявления принципа:

Визуальный компонент отображения таблицы рейсов



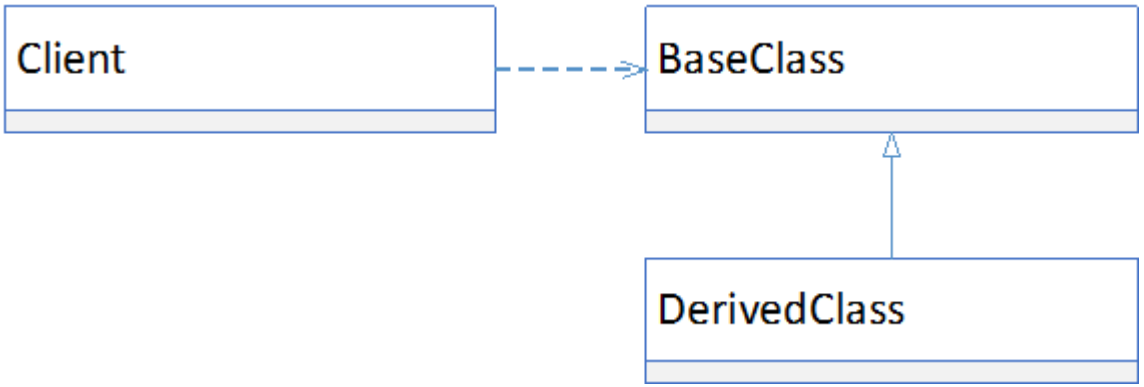
Интерфейс компонента отображения

Текстовый компонент отображения таблицы рейсов

Принцип подстановки Барбары Лисков…

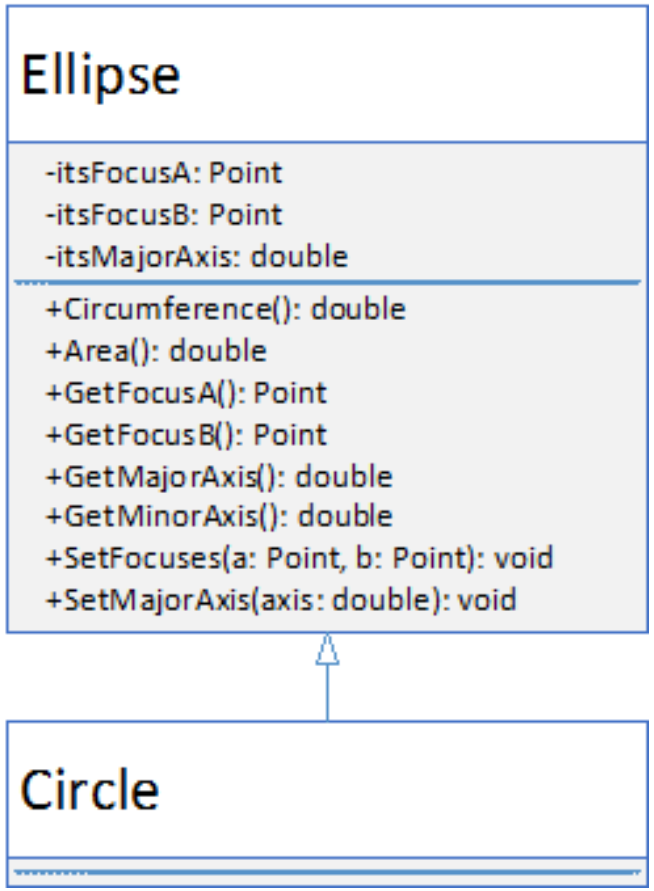
- ***Классы-наследники должны подставляться вместо родительских классов.***

- Клиенты базового класса должны правильно функционировать, если им отдавать экземпляры любого производного класса.



Принцип подстановки Барбары Лисков…

- Дилемма «окружность/эллипс».

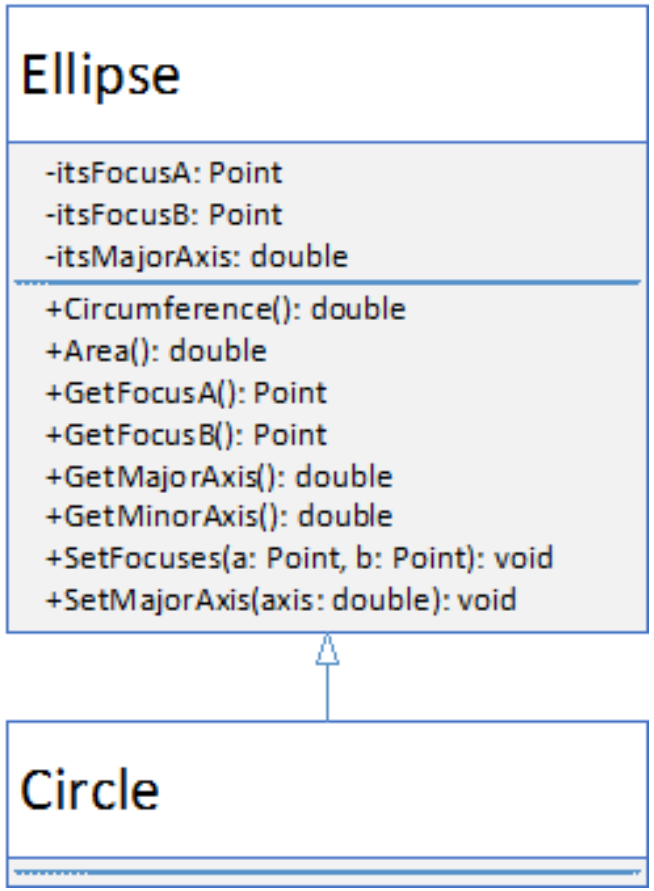



```
void f(Ellipse& e)
{
    Point a(-1,0);
    Point b(1,0);
    e.SetFocuses(a,b);
    e.SetMajorAxis(3);
    assert(e.GetFocusA() == a);
    assert(e.GetFocusB() == b);
    assert(e.GetMajorAxis() == 3);
}
```

Функция работает корректно,
если передан объект типа Ellipse,
но не Circle (второй фокус игнорируется)

Принцип подстановки Барбары Лисков…

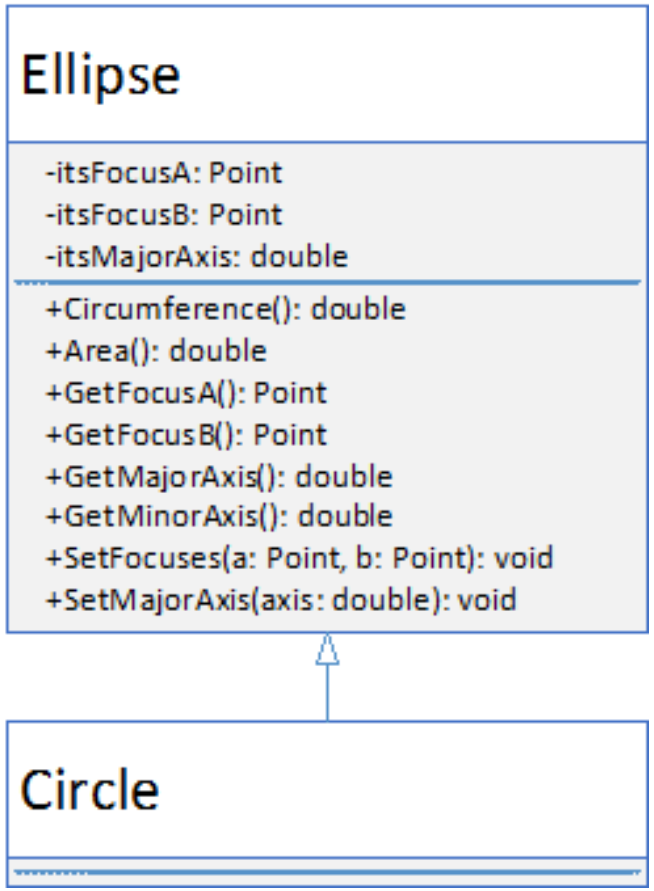
- Дилемма «окружность/эллипс». Плохое решение.




```
void f(Ellipse& e)
{
    if (typeid(e) == typeid(Ellipse))
    {
      Point a(-1,0);
      Point b(1,0);
      e.SetFocuses(a,b);
      e.SetMajorAxis(3);
      assert(e.GetFocusA() == a);
      assert(e.GetFocusB() == b);
      assert(e.GetMajorAxis() == 3);
    }
    else
      throw NotAnEllipse(e);
}
```

Принцип подстановки Барбары Лисков…

- Дилемма «окружность/эллипс». Использование контракта.




```
// Предусловия:
// Если не выполнено предусловие,
// результат работы не определен и
// функцию нецелесообразно вызывать.
void f(Ellipse& e)
{
    // тело функции
}
// Постусловия:
// Условия, которые выполнены, если
// функция завершила работу.
// Значение постусловия не
// возвращается, если функция
// завершилась некорректно.
```

Контракты поддерживаются некоторыми языками (например, Eiffel).

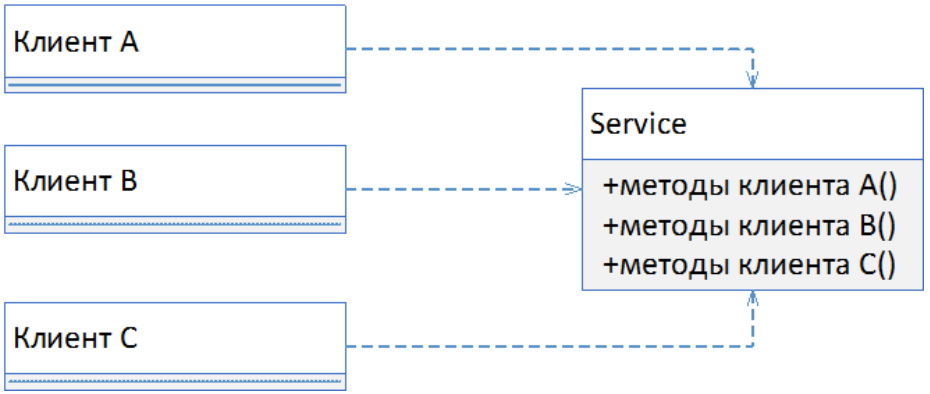
Принцип отделения интерфейсов…

- ***Лучше разработать специальные интерфейсы для многих типов клиентов, чем один интерфейс общего назначения.***

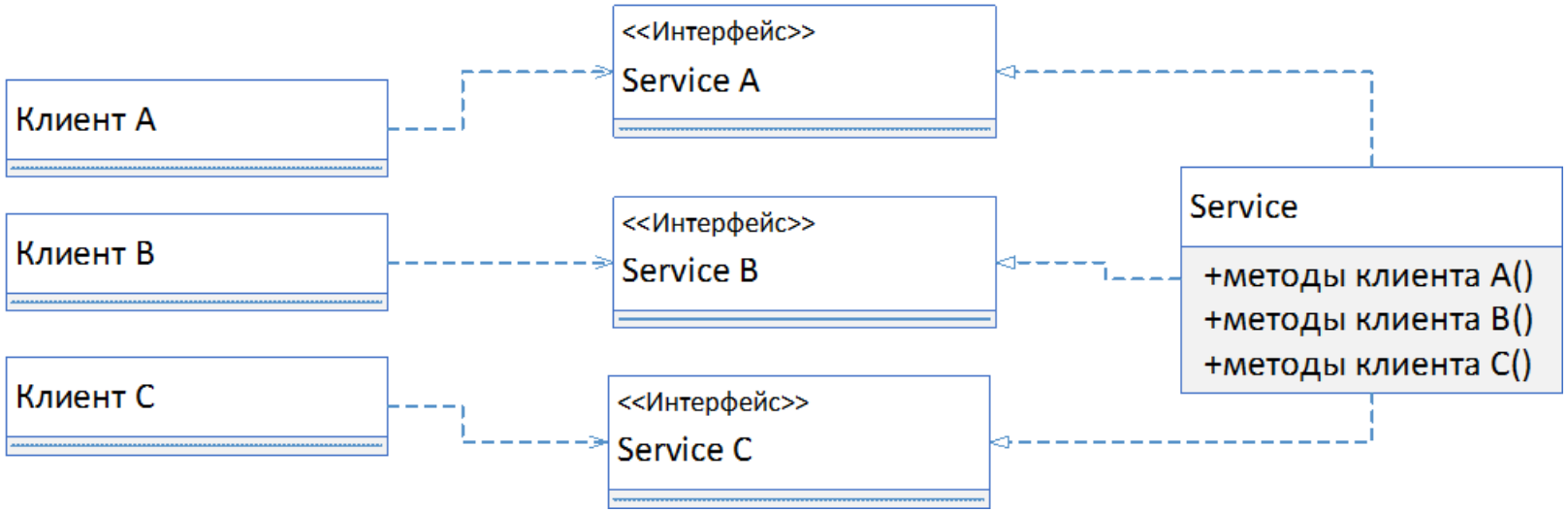
- Если есть класс, имеющий несколько клиентов, вместо того, чтобы загружать класс со всеми методами, которые необходимы всем клиентам в совокупности, имеет смысл создавать интерфейс для каждого клиента и использовать механизм множественного наследования при реализации класса.

Принцип отделения интерфейсов…

- «Толстый» сервис с единым интерфейсом:



- Выделенные интерфейсы:



Принцип инверсии зависимости…

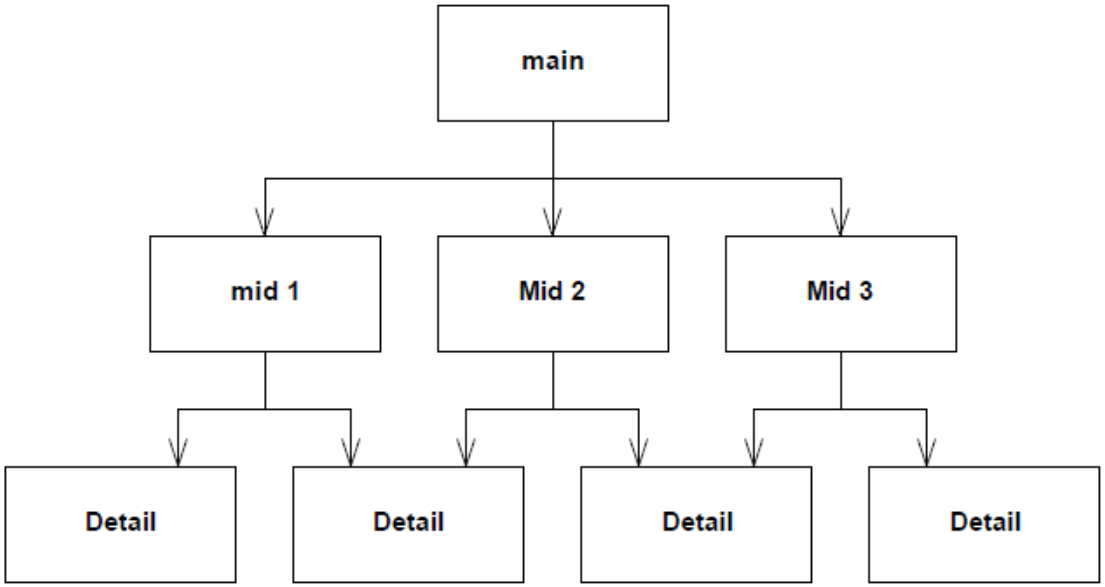
- ***Зависеть от абстракций и не зависеть от конкреций.***

- Инверсия зависимости – стратегия, согласно которой клиент зависит от интерфейсов, абстрактных функций и классов, а не от их конкретных реализаций.
- Зависимость от конкретных реализаций интерфейсов присутствует только в местах создания экземпляров.
- Создание экземпляров может происходить в любой момент внутри системы. Решение – шаблон проектирования «Абстрактная фабрика» (Abstract factory).
- Принцип является основой для компонентного архитектурного стиля.

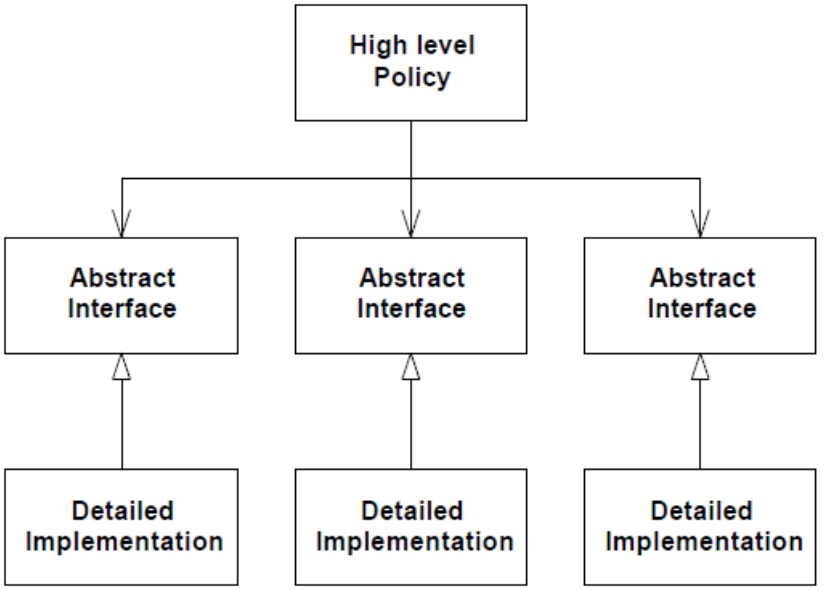
Принцип инверсии зависимости…

- Структура зависимости:

Процедурная архитектура



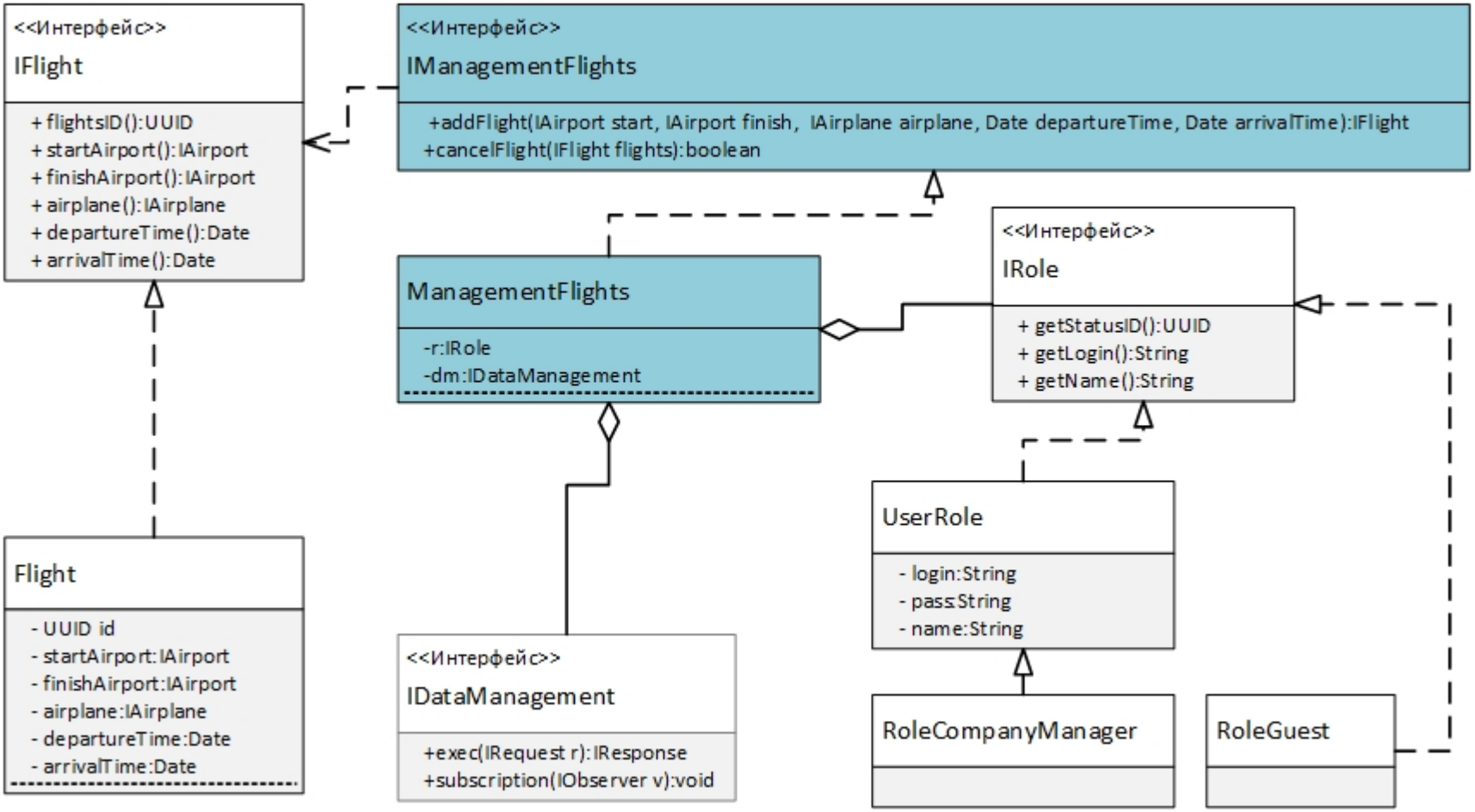
Объектно-ориентированная архитектура



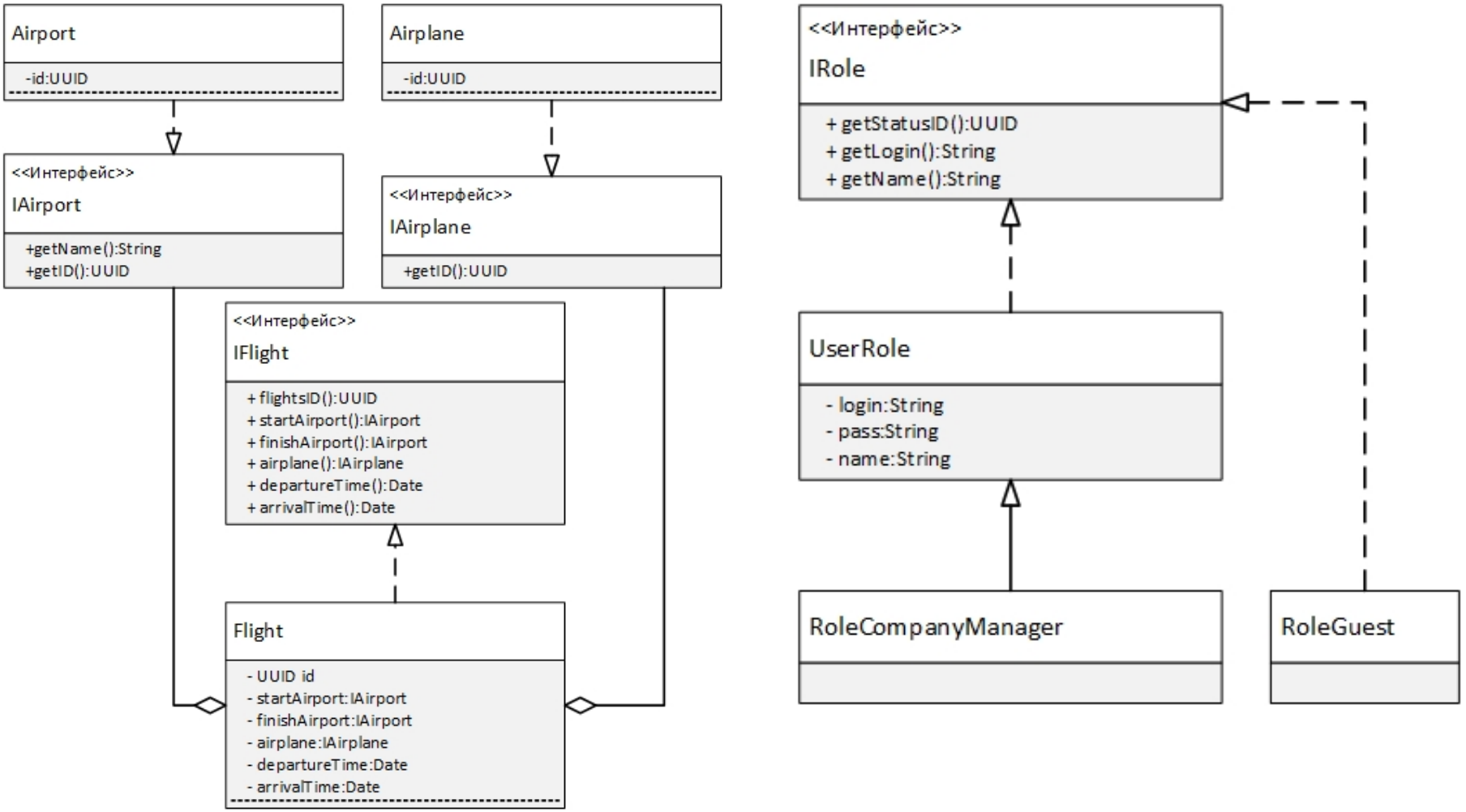
**Martin R.C. Design Principles and Design Patterns [http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf].*

Принцип инверсии зависимости

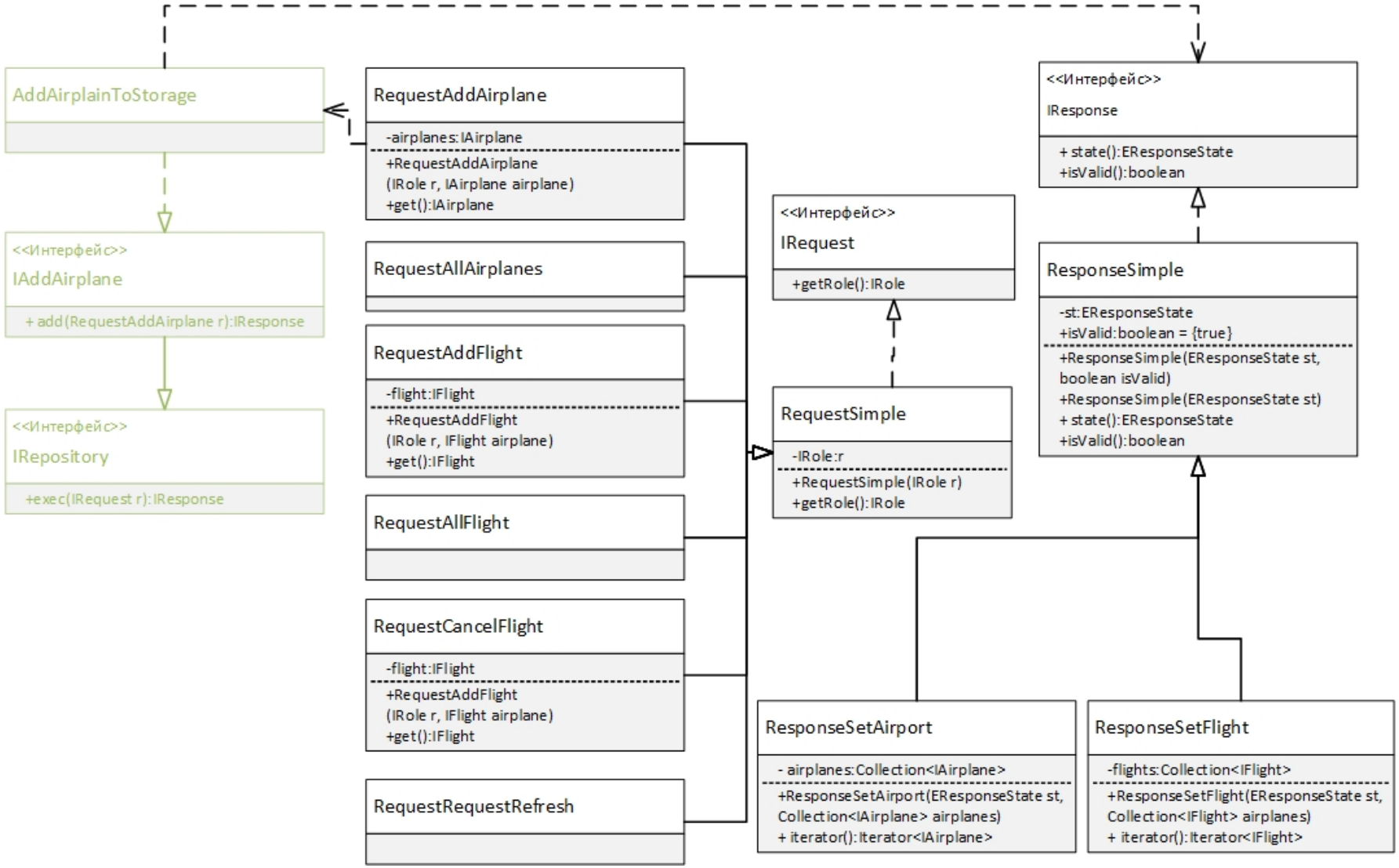
- Пример проявления принципа:



Пример проектирования подсистем. Сущности системы



Пример проектирования подсистем. Подсистема организации доступа к данным



Заключение

- Рассмотрены составляющие объектно-ориентированной разработки подсистем (анализ, проектирование, программирование).
- Изучены основные принципы объектного подхода: абстракция, инкапсуляция, полиморфизм, наследование.
- Введены понятия объекта и класса. Рассмотрены возможные отношения между объектами, между классами, между объектами и классами.
- Рассмотрены основные признаки испорченной архитектуры.
- Изучены принципы объектно-ориентированного проектирования классов (SOLID-принципы).

Контрольные вопросы…

- Какие методологии включает в себя объектно-ориентированная разработка?
- Приведите основные принципы объектного подхода. Как они проявляются на практике? Приведите примеры.
- Что такое состояние, поведение и идентичность объекта?
- Какие существуют типы отношений между объектами? Приведите примеры.
- Какие существуют типы отношений между классами? С помощью каких подходов указанные типы отношений выражаются в языках программирования?
- Приведите основные признаки неудачной архитектуры системы.

Контрольные вопросы

- В чем состоит принцип единственной ответственности (The Single Responsibility Principle, SRP)? Приведите пример.
- В чем состоит принцип открытия/закрытия (The Open Closed Principle, OCP)? Приведите пример.
- В чем состоит принцип подстановки Барбары Лисков (The Liskov Substitution Principle, LSP)? Приведите пример.
- В чем состоит принцип отделения интерфейса (The Interface Segregation Principle, ISP)? Приведите пример.
- В чем состоит принцип инверсии зависимости (The Dependency Inversion Principle DIP)? Приведите пример.

Литература к лекции

1. **Sommerville I.** Software engineering (9th edition). – Pearson, 2011. – 790 p.
2. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами на C++. – Изд-во: Бином, Невский диалект, 1998. – 560с.
3. **Martin R.C.** Design Principles and Design Patterns [http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf].
4. **Martin R.C.** SRP: The Single Responsibility Principle [http://objectmentor.com/resources/articles/srp.pdf].