Объектно – ориентированное программирование

Алан Кей – фундаментальные характеристики ООП [Kay 1993]:

Все является объектом.

Объект как хранит информацию, так и способен ее преобразовывать. Любой элемент решаемой задачи (дом, собака, услуга, химическая реакция, город, космический корабль и т. д.) может представлять собой объект. Объект можно представить себе как швейцарский нож: он является набором различных ножей и «открывашек» (хранение), но в то же самое время им мы можем резать или открывать что-либо (преобразование).

Программа — совокупность объектов, указывающих друг другу что делать.

Для обращения к одному объекту другой объект «посылает ему сообщение».

Возможно и «ответное сообщение».

Программу можно представить себе как совокупность к примеру 3 объектов: писателя, ручки и листа бумаги. Писатель «посылает сообщение» ручке, которая в свою очередь «посылает сообщение» листу бумаги — в результате мы видим текст (посыл сообщения от листа к писателю).

Каждый объект имеет свою собственную «память», состоящую из других объектов.

Таким образом программист может скрыть сложность программы за довольно простыми объектами.

Пример: дом (сложный объект) состоит из дверей, комнат, окон, проводки и отопления. Дверь может состоять из собственно двери, ручки, замка и петель. Проводка состоит из проводов, розеток, щитка.

У каждого объекта есть тип (класс).

Класс (тип) определяет какие сообщения объекты могут посылать друг другу.

Например, аккумуляторная батарея может передавать электролампе ток, а вот физическое усилие - нет.

Каждый объект является представителем класса, который выражает общие свойства объектов (таких, как целые числа или списки).

В классе задается поведение (функциональность) объекта.

Тем самым все объекты, которые являются экземплярами одного класса, могут выполнять одни и те же действия.

Все объекты одного типа могут получать одинаковые сообщения.

Например, есть 2 объекта: синяя и красная кружки разные по форме и материалу. Но из обеих мы можем пить (или не пить, если они пустые). В данном случае кружка — это тип объекта.

Классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования.

Память и поведение, связанное с экземплярами определенного класса, автоматически доступны любому классу, расположенному ниже в иерархическом дереве.

1ый принцип ОО подхода - способ задания действий.

Действие в ООП инициируется посредством передачи сообщений агенту (объекту), ответственному за действия.

Сообщение содержит запрос на осуществление действия и сопровождается дополнительной информацией (аргументами), необходимой для его выполнения.

Получатель - это агент, посылается сообщение. Если он принимает сообщение, то на него автоматически возлагается ответственность за выполнение указанного действия.

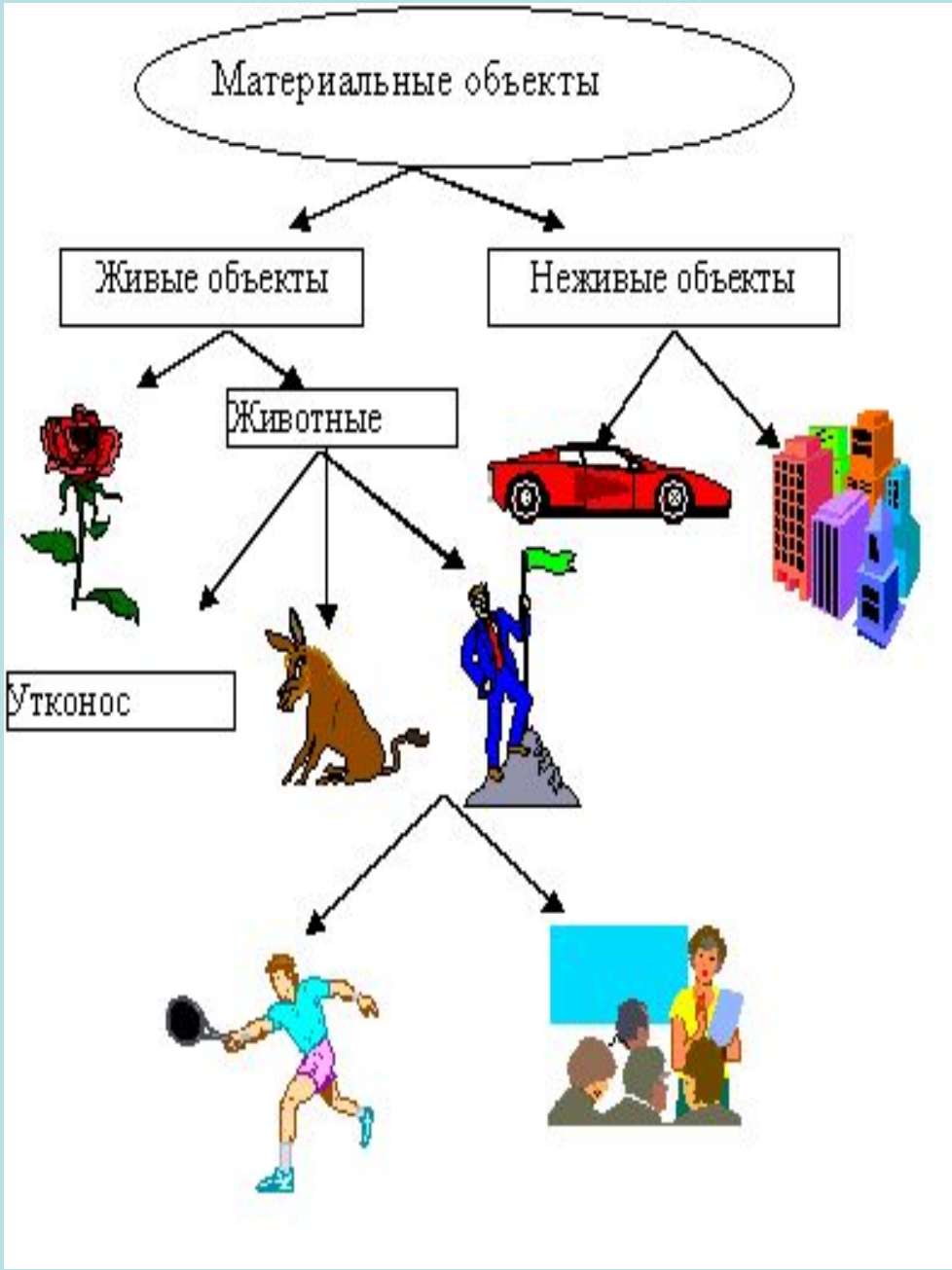
В качестве реакции на сообщение получатель запустит некоторый метод, чтобы удовлетворить принятый запрос.

2ой принцип ООП: *Все объекты являются представителями (экземплярами) классов. Метод активизируемый объектом в ответ на сообщение, определяется классом, к которому принадлежит получатель сообщения. Все объекты одного класса используют одни и те же методы в ответ на одинаковые сообщения.*

Весь мир - это объекты, которые передают друг другу сообщения

Зий принцип, в соответствии с которым знание о более общей категории разрешается использовать для более узкой категории, называется *наследованием.*

Классы могут быть организованы в иерархическую структуру с наследованием свойств. Дочерний класс (подкласс) наследует атрибуты родительского класса (надкласса), расположенного выше в иерархическом дереве.



Определение ООП

ООП - совокупность принципов разработки программ, понятий и элементов языка, позволяющих успешно создавать программы большого объема.

ООП базируется на связи в единое целое свойств и поведения предмета или процесса.

Свойства предмета - это данные, которые его характеризуют. Поведение задается функциями.

Преимущества ООП

Преимущества (при создании больших программ):

- использование при программировании понятий, более близких к предметной области;
- локализация свойств и поведения объекта о одном месте, позволяющая лучше структурировать и отлаживать программу;
- возможность создания библиотеки объектов и создания программы из готовых частей;
- исключение избыточного кода - можно многократно не описывать повторяющиеся действия;
- простая возможность внесения изменений в программу без изменения уже написанных частей, а в ряде случаев и без их перекомпиляции.

Недостатки ООП

Недостатки ООП:

- снижение быстродействия программы, связанное с использованием виртуальных методов;
- идеи ООП не просты для понимания и в особенности для практического использования;
- для эффективного использования существующих ОО систем требуется большой объем первоначальных знаний.

Свойства ООП

- ***Инкапсуляция*** - скрытие деталей реализации; объединение данных и действий над ними.

- ***Наследование*** позволяет создавать иерархию объектов, в которой объекты-потомки наследуют все свойства своих предков. Свойства при наследовании повторно не описываются. Кроме унаследованных, потомок обладает собственными свойствами. Объект в C++ может иметь сколько угодно потомков и предков.

- ***Полиморфизм*** - возможность определения единого по имени действия, применимого ко всем объектам иерархии, причем каждый объект реализует это действие собственным способом.

Класс (объект) – инкпасулированная абстракция с четким протоколом доступа

Объекты

Объект - 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

Стул	
Стоимость Вес Размеры Положение Цвет	Перечень Свойств
Купить() Продать() Взвесить() Переместить() Покрасить()	Перечень операций

Объекты не существуют изолированно друг от друга. Они подвергаются воздействию или сами воздействуют на другие объекты.

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

Поведение объекта является функцией как его состояния, так и выполняемых им операций (Купить, Продать, Взвесить, Переместить, Покрасить).

Говорят, что состояние объекта представляет суммарный результат его поведения.

Операция обозначает обслуживание, которое объект предлагает своим клиентам.

5 видов операций клиента над объектом:

1) модификатор (изменяет состояние объекта);
2) селектор (дает доступ к состоянию, но не изменяет его);
3) итератор (доступ к содержанию объекта по частям, в строго определенном порядке);
4) конструктор (создает объект и инициализирует его состояние);
5) деструктор (разрушает объект и освобождает занимаемую им память).

Примеры операций:

Вид операции	**Пример операции**
Модификатор	Пополнеть (кг)
Селектор	КакойВес (): integer
Итератор	Показать Ассортимент Товаров (): string
Конструктор	СоздатьРобот (параметры)
Деструктор	УничтожитьРобот ()

В чистых ОО ЯП операции могут объявляться только как методы - элементы классов, экземплярами которых являются объекты.

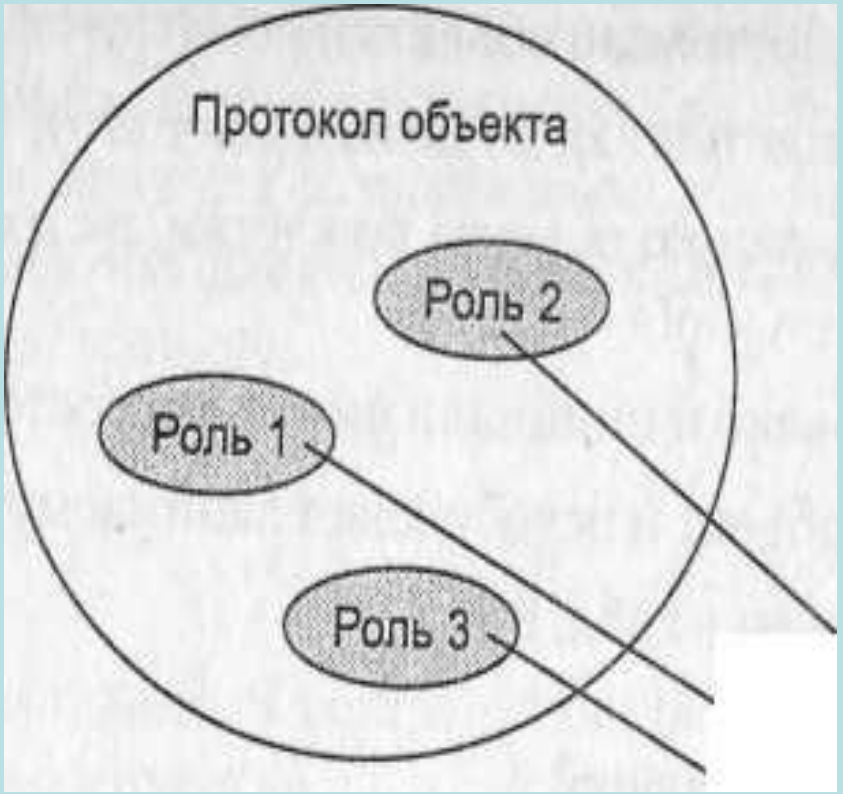
Гибридные языки (C++, Ada 95) позволяют писать операции как свободные подпрограммы (вне классов).

В общем случае все методы и свободные подпрограммы, ассоциированные с конкретным объектом, образуют его ***протокол***. Таким образом, протокол определяет оболочку допустимого поведения объекта и поэтому заключает в себе цельное (статическое и динамическое) представление объекта.

Принцип выделения ролей:

Большой протокол разделяют на логические группировки поведения.

Эти группировки, разделяющие пространство поведения объекта, обозначают ***роли***, которые может играть объект.



С точки зрения внешней среды важное значение имеет понятие - обязанности объекта.

Обязанности означают обязательства объекта обеспечить определенное поведение.

Обязанностями объекта являются все виды обслуживания, которые он предлагает клиентам. В мире объект играет определенные роли, выполняя свои обязанности.

Активный объект имеет собственный канал (поток) управления, пассивный - нет.

Активный объект автономен, он может проявлять свое поведение без воздействия со стороны других объектов.

Пассивный объект, наоборот, может изменять свое состояние только под воздействием других объектов.

Виды отношений между объектами

В поле зрения разработчика ПО находятся не объекты-одиночки, а взаимодействующие объекты, именно взаимодействие объектов реализует поведение системы.

Г. Буч (цитата из Галла): ***«Самолет - это набор элементов, каждый из которых по своей природе стремится упасть на землю, но ценой совместных непрерывных усилий преодолевает эту тенденцию».***

Отношения между парой объектов основываются на взаимной информации о разрешенных операциях и ожидаемом поведении.

Интересны два вида отношений между объектами: ***связи и агрегация.***

Связи

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.

Связи являются рельсами между станциями-объектами, по которым ездят «трамвайчики сообщений».

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 пользоваться ими. Агент создается для выполнения работы от имени актера или другого агента.

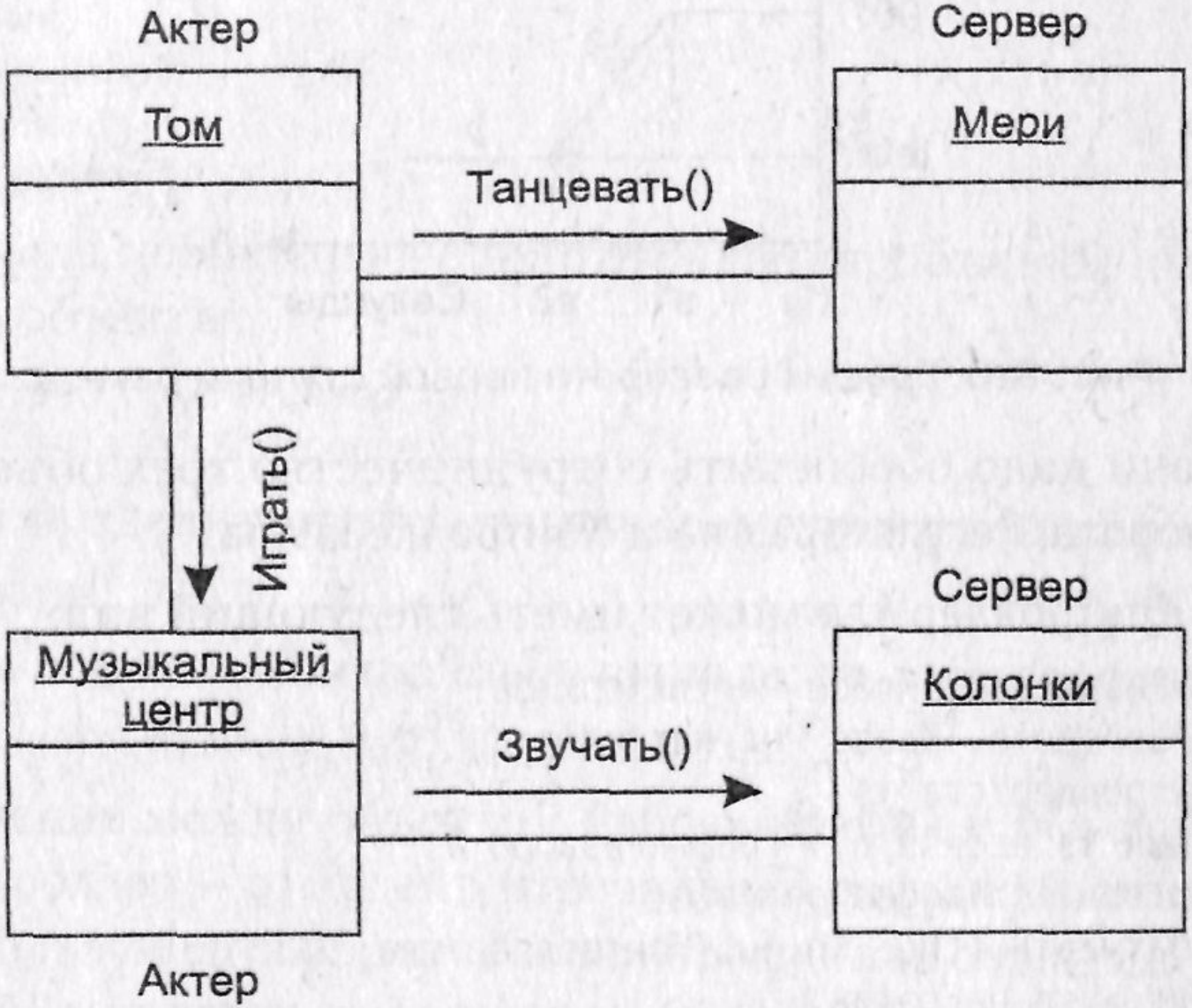


Рис. 9.5. Связи между объектами

Видимость объектов

Рассмотрим два объекта, A и B, между которыми имеется связь. Для того чтобы объект A мог послать сообщение в объект B, надо, чтобы B был виден для A.

4 формы видимости между объектами.

1. Объект-поставщик (сервер) глобален для клиента.
2. Объект-поставщик (сервер) является параметром операции клиента.
3. Объект-поставщик (сервер) является частью объекта-клиента.
4. Объект-поставщик (сервер) является локально объявленным объектом в операции клиента.

На этапе анализа вопросы видимости обычно опускают. На этапах проектирования и реализации вопросы видимости по связям обязательно должны рассматри ваться.

Агрегация

Связи обозначают равноправные (клиент-серверные) отношения между объектами. 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.

При выборе вида отношения - учитывать факторы:

- связи обеспечивают низкое сцепление между объектами;
- агрегация инкапсулирует части как секреты целого.

Пример физического включения (композиции) частей (Двигателя, Сидений, Колес) в агрегат Автомобиль - части включены ***в агрегат по величине.***

Рис. 9.7. Физическое включение частей в агрегат

Пример нефизического включения частей (Студента, Преподавателя) в агрегат Вуз. Студент и Преподаватель являются элементами Вуза, но они не входят в него физически. Части включены в ***агрегат по ссылке.***

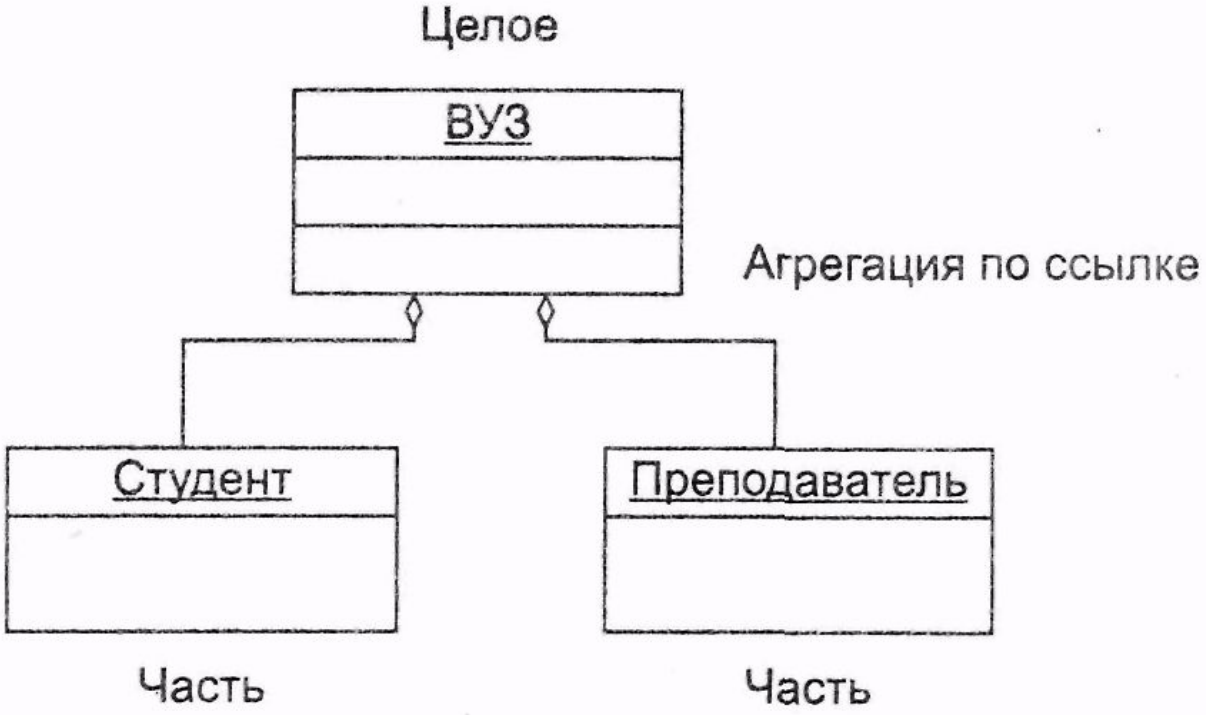


Рис. 9.8. *Нефизическое включение частей в агрегат*

Классы

Понятия объекта и класса тесно связаны.

Важное различие:

Класс - это абстракция существенных характеристик объекта.

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

Любой объект - просто экземпляр класса.

Различают внутреннее представление класса (реализацию) и внешнее представление класса (интерфейс).

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

Интерфейс демонстрирует внешнему миру абстракцию класса, его внешний облик.

Интерфейс состоит из объявлений всех операций, применимых к экземплярам класса. Он может включать объявления типов, переменных, констант и исключений, необходимых для полноты данной абстракции.

Интерфейс - 3 части:

1. публичная (public) - объявления доступны всем клиентам;
2. защищенная (protected) - объявления доступны только самому классу, его подклассам и друзьям;
3. приватная (private) - объявления доступны только самому классу и его друзьям.

КЛАСС	
интерфейсные:	
Части	**Публичная**
	Защищенная
	Приватная
Реализация	

Друг класса - класс, который имеет доступ ко всем частям класса (публичной, защищенной и приватной). От друга у класса нет секретов. Реализация класса описывает секреты поведения класса. Она включает реализации всех операций, определенных в интерфейсе класса.