Тема лекции №8

Объектно-ориентированное программирование как инструмент для моделирования транспортных процессов

Цель лекции - изучить особенности объектно-ориентированного программирования

План лекции.

1. Основные понятия.
2. Принципы ООП.
3. Особенности реализации ООП.
4. Объектно-ориентированные языки.

1. Основные понятия

Объе́ктно-ориенти́рованное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Особенности определения:

1) объектно-ориентированное программирование использует в качестве основных логических конструктивных элементов объекты, а не алгоритмы;

2) каждый объект является экземпляром определенного класса;

3) классы образуют иерархии.

Программа считается объектно-ориентированной, только если выполнены все три указанных требования.

Класс - универсальный, комплексный тип данных, состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями), то есть он является моделью информационной сущности с внутренним и внешним интерфейсами для оперирования своим содержимым (значениями полей).

Класс - это шаблон, описание ещё не созданного объекта. Класс содержит данные, которые описывают строение объекта и его возможности, методы работы с ним.

Объект - некоторая сущность в компьютерном пространстве, обладающая определённым состоянием и поведением, имеющая заданные значения свойств (атрибутов) и операций над ними (методов). Как правило, при рассмотрении объектов выделяется то, что объекты принадлежат одному или нескольким классам, которые определяют поведение (являются моделью) объекта.

Объект — экземпляр класса. То, что «рождено» по «чертежу», то есть по описанию из класса.

В качестве примера объекта и класса можно привести технический чертёж для изготовления детали — это класс. Выточенная же на станке по размерам и указаниям из чертежа деталь - объект.

Модульность - это такая организация объектов, когда они заключают в себе полное определение их характеристик, никакие определения методов и свойств не должны располагаться вне его, это делает возможным свободное копирование и внедрение одного объекта в другие.

Событие в объектно-ориентированном программировании - это сообщение, которое возникает в различных точках исполняемого кода при выполнении определённых условий.

События предназначены для того, чтобы иметь возможность предусмотреть реакцию программного обеспечения.

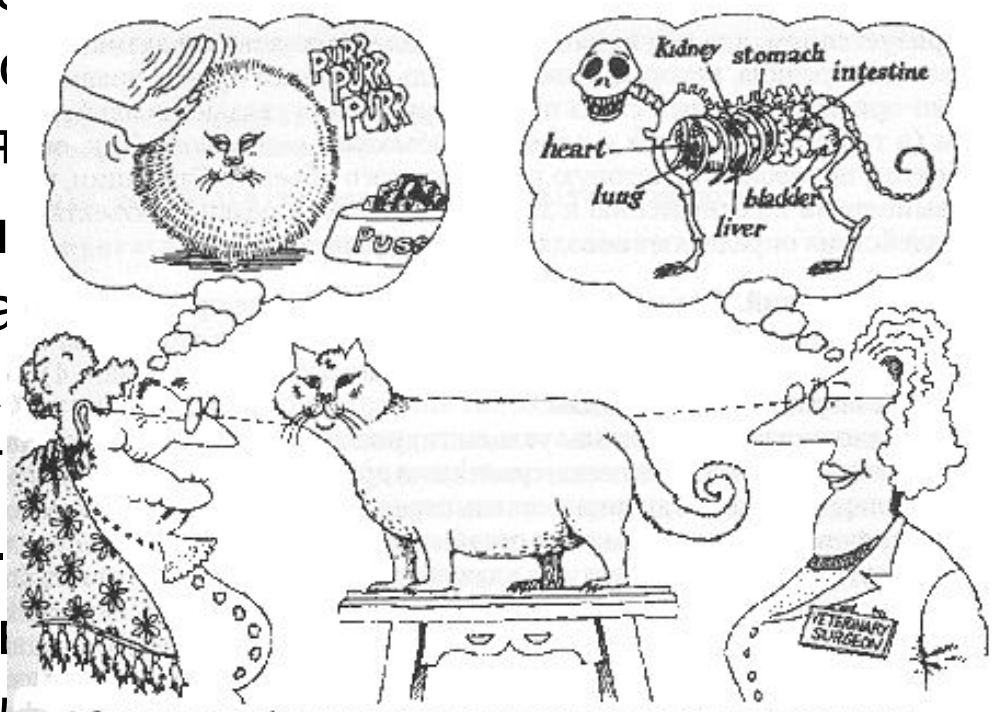
Для решения поставленной задачи создаются обработчики событий: как только программа попадает в заданное состояние, происходит событие, посылается сообщение, а обработчик перехватывает это сообщение.

2. Принципы ООП.

Абстракция — спо… …ых значимых характеристик … …ие отбрасываются

В ООП абстракци… характеристика… …тобы отделить соста… …ньших» объектов, от эт… составляющих.

Такой подход поз… …с объектом, не в… …е он состоит и как работает.



Абстракция фокусирует внимание на существенных характеристиках объекта с точки зрения наблюдателя

Инкапсуляция — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе.

Инкапсуляция - принцип **объектно-ориентированного программирования**, позволяющий собрать объект в пределах одной структуры или массива, убрав способ обработки данных и сами данные от «чужих глаз».

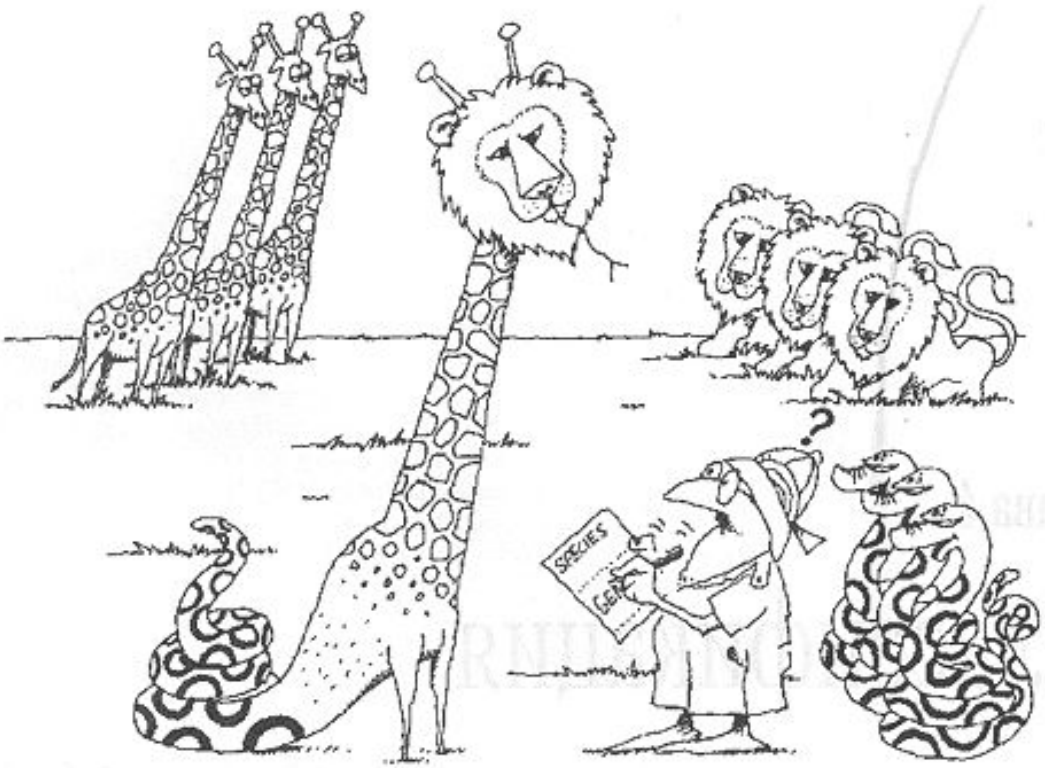
Инкапсуляцию применяют:

- когда нужно сохранить некоторый участок кода без изменений со стороны пользователя;
- когда нужно ограничить доступ к коду - в связи с уникальностью используемых техник, которые автор хочет оставить «при себе»;
- когда изменение кода повлечёт за собой неработоспособность программы или её взлом.

Наследование — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью.

Класс, от которого производится наследование, называ[ется] перклассом. Новый [класс] очерним или произв[одным]

Наследов[ание] ориент[ировано] роить новый класс н[а] ункционал может [быть] чаться. Класс-[родитель] дительским или ба[зовым] ком, дочерн[им]



Наследование позволяет приобретать свойства родителей

Полиморфизм — свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Полимор
опред ить в
завис
наход
завис
дейст дить
лишн йс —
множ
Полимо орного
испол змер
прогр ние.

В связи со своими особенностями объектно-ориентированное программирование имеет ряд преимуществ перед другими видами программирования:

1) Использование одного и того же программного кода с разными данными. Классы позволяют создавать множество объектов, каждый из которых имеет собственные значения атрибутов.

2) Наследование и полиморфизм позволяют не писать новый код, а настраивать уже существующий, за счет добавления и переопределения атрибутов. Это ведет к сокращению объема исходного кода.

3. Особенности реализации ООП.

В современных объектно-ориентированных языках программирования каждый объект является значением, относящимся к определённому классу. Класс представляет собой объявленный программистом составной тип данных, имеющий в составе:

Поля данных

– Параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние (свойства объекта предметной области). Физически поля представляют собой значения (переменные, константы), объявленные как принадлежащие классу.

Методы

– Процедуры и функции, связанные с классом. Они определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.

Классы могут наследоваться друг от друга. Класс-потомок получает все поля и методы класса-родителя, но может дополнять их собственными либо переопределять уже имеющиеся.

Интерфейс — это класс без полей и без реализации, включающий только заголовки методов.

Если некий класс наследует (или, как говорят, реализует) интерфейс, он должен реализовать все входящие в него методы.

Использование интерфейсов предоставляет относительно дешёвую альтернативу множественному наследованию.

Инкапсуляция обеспечивается следующими средствами:

Контроль доступа

– Поскольку методы класса могут быть как чисто внутренними, обеспечивающими логику функционирования объекта, так и внешними, с помощью которых взаимодействуют объекты, необходимо обеспечить скрытость первых при доступности извне вторых. Для этого в языки вводятся специальные синтаксические конструкции, явно задающие область видимости каждого члена класса.

Методы доступа

– Поля класса в общем случае не должны быть доступны извне, поскольку такой доступ позволил бы произвольным образом менять внутреннее состояние объектов. Поэтому поля обычно объявляются скрытыми (либо язык в принципе не позволяет обращаться к полям класса извне), а для доступа к находящимся в полях данным используются специальные методы, называемые методами доступа.

Свойство - способ доступа к внутреннему состоянию объекта, имитирующий переменную некоторого типа. Обращение к свойству объекта реализовано через вызов функции. При попытке задать значение данного свойства вызывается один метод, а при попытке получить значение данного свойства — другой.

При применении свойств:

- можно задать значение по умолчанию, которое будет храниться в данном свойстве (или указать, что никакого значения по умолчанию не предполагается);
- можно указать, что это свойство только для чтения.

Как правило, свойство связано с некоторым внутренним полем объекта. Но свойству вообще может не быть сопоставлена ни одна переменная объекта, хотя пользователь данного объекта имеет дело с ним так, как если бы это было настоящее поле.

4. Объектно-ориентированные языки

Объектно-ориентированный язык программирования - язык, построенный на принципах ООП.

Как правило, объектно-ориентированный язык (ООЯ) содержит следующий набор элементов:

- Объявление классов с полями (данными — членами класса) и методами (функциями — членами класса).
- Механизм расширения класса (наследования) — порождение нового класса от существующего с автоматическим включением всех особенностей реализации класса-предка в состав класса-потомка. Большинство ООЯ поддерживают только единичное наследование.
- Полиморфные переменные и параметры функций (методов), позволяющие присваивать одной и той же переменной экземпляры различных классов.
- Полиморфное поведение экземпляров классов за счёт использования виртуальных методов. В некоторых ООЯ все методы классов являются виртуальными.

Некоторые языки добавляют к указанному минимальному набору дополнительные средства:

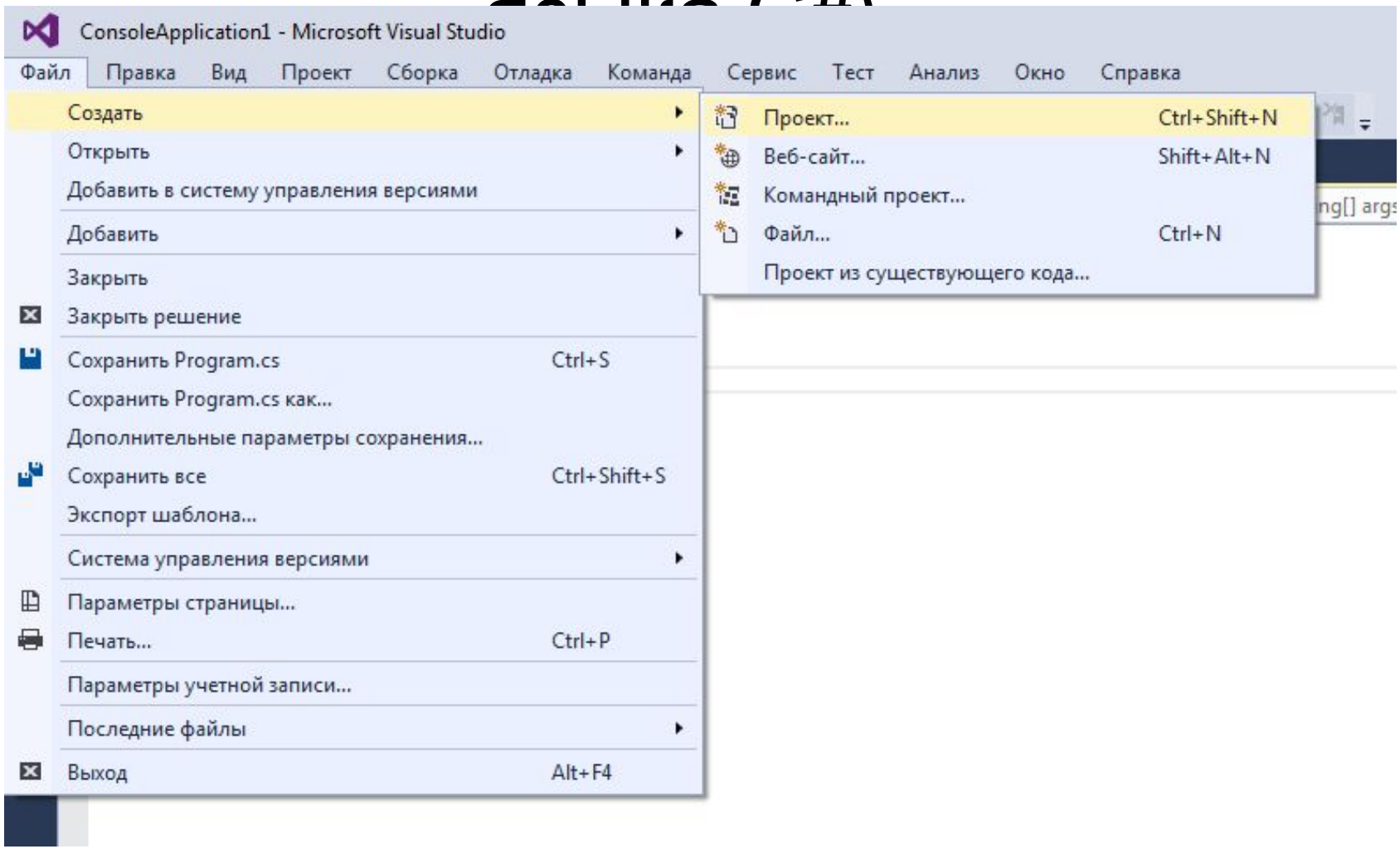
- Конструкторы, деструкторы, финализаторы;
- Свойства (аксессоры);
- Индексаторы;
- Средства управления видимостью компонентов классов

Одни языки отвечают принципам ООП в полной мере — в них все основные элементы являются объектами, имеющими состояние и связанные методы. Примеры подобных языков — SmalltalkОдни языки отвечают принципам ООП в полной мере — в них все основные элементы являются объектами, имеющими состояние и связанные методы. Примеры подобных языков — Smalltalk, Eiffel.

Существуют гибридные языки, совмещающие объектную подсистему в целостном виде с подсистемами других парадигм как «два и более языка в одном», позволяющие совмещать в одной программе объектные модели с иными, и размывающие грань между объектно-ориентированной и другими парадигмами за счёт нестандартных возможностей, балансирующих между ООП и другими парадигмами (таких как множественная диспетчеризация, параметрические классы, возможность манипулировать методами классов как самостоятельными объектами, и др.). Примеры таких языков: CLOSCLOS, DylanCLOS, Dylan, OCamlCLOS, Dylan, OCaml, PythonCLOS, Dylan, OCaml, Python, RubyCLOS, Dylan, OCaml, Python, Ruby, Objective-C.

Однако, наиболее распространены языки, включающие средства эмуляции объектной модели поверх более традиционной императивной семантики. Примеры таких языков — СимулаОднако, наиболее распространены языки, включающие средства эмуляции объектной модели поверх более

Создание проекта в среде MS Visual Studio (Пример работы



https: // www.visualstudio.com / ru-ru / downloads / download-visual-studio-vs.aspx