# Базовые понятия ООП

# Понятие объектно-ориентированного программирования

- ***Объектно-ориентированное программирование*** (Object-Oriented Programming) — совокупность принципов, технологии и инструментальных средств для создания программных систем, в основу которых закладывается архитектура взаимодействия объектов
- ***Абстракция*** — характеристика сущности, которая отличает ее от других сущностей
- ***Наследование*** — принцип, в соответствии с которым знание о более общей категории разрешается применять для более частной категории
- ***Инкапсуляция*** — сокрытие отдельных деталей внутреннего устройства классов от внешних по отношению к нему объектов или пользователей
- ***Полиморфизм*** — свойство элементов модели с одинаковыми именами иметь различное поведение

# Объектно-ориентированное программирование

- ООП – это парадигма программирования, в которой базовым является понятие объекта

- Объект имеет
  - Состояние
  - Поведение
  - Уникальность
- Объект умеет
  - Получать сообщения
  - Обрабатывать данные
  - Отправлять сообщения

- Программа в ходе работы представляет собой набор взаимодействующих объектов

# Основные понятия объектно-ориентированного программирования

- Класс – тип данных для создания объектов
- Объект – экземпляр класса
- Метод – процедура над объектом
- Конструктор и деструктор – особые методы
- Свойство – виртуальное поле
- Наследование – расширение класса
- Виртуальный метод – переопределяемый метод
- Делегат – ссылка на метод
- Событие – список делегатов
- Интерфейс – описание класса без реализации
- Шаблон – параметризованный класс
- Атрибут – метаданные, механизм рефлексии

# Понятие класса, объекта и компонента

*Класс* и *объект* — два общепринятых термина. Какова же разница между ними?

Термин *класс* объединяет объекты с одинаковыми возможностями (данными и методами). Он описывает общее поведение и характеристики набора аналогичных друг другу объектов. *Объект — это экземпляр класса* или, другими словами, переменная, тип которой задается классом. Объекты в отличие от классов реальны, т. е. существуют и хранятся в памяти во время выполнения программы. Соотношения между объектом и классом аналогичны соотношениям между переменной и типом.

***Компоненты.*** Использование библиотек классов повышает скорость разработки программ, но, с другой стороны, требует определенных усилий для изучения этих библиотек и понимания того, как они устроены. Кроме того, библиотека классов должна быть написана на том же языке программирования, что и разрабатываемая программа. Конечно, существуют способы сопряжения разных языков программирования, но все равно, для того чтобы использовать, например, для программы, написанной на языке Pascal, библиотеку классов C++, необходимо написать программу с вызовами нужных функций или порождением необходимых классов.

Подобные неудобства привели к появлению концепции *компонента* — программного модуля или объекта, который готов для использования в качестве составного блока программы и которым можно визуально манипулировать во время разработки программы.

*Компонент — это объект, объединяющий состояние и интерфейс* (*способ взаимодействия*). Состояние компонента может быть изменено только с помощью изменения его свойств и вызова методов.

# Типы интерфейсов у компонентов

У компонента имеются два типа интерфейсов: интерфейс стадии проектирования и интерфейс стадии выполнения. Интерфейс проектирования позволяет включать компоненты в современные среды разработки приложений, а интерфейс выполнения управляет работой компонента во время выполнения программы. При этом неважно, на каком языке программирования реализован компонент. Он должен просто удовлетворять определенным внешним параметрам и быть нейтрален по отношению к языку программирования, чтобы его можно было использовать в программе на любом языке, поддерживающем компонентную технологию. Так, например, компоненты стандарта ActiveX могут быть одинаково успешно включены в программу, реализованную в среде Visual Basic, и в приложение, разработанное средствами Delphi.

Компоненты графического интерфейса, управляемые событиями, являются основным «строительным» материалом при разработке приложений средствами графических редакторов. Разработка любого приложения состоит из двух взаимосвязанных этапов:

- проектирование и создание функционального интерфейса приложения (т.е. набора визуальных компонентов, которые будут обеспечивать взаимодействие пользователя и вычислительной среды);
- программирование процедур обработки событий, возникающих при работе пользователя с приложением.

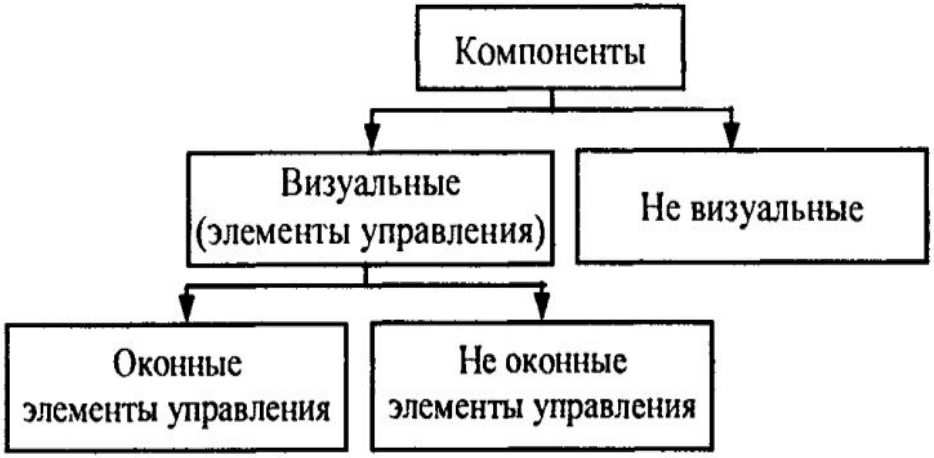
*На первом этапе* (т.е. на этапе проектирования интерфейса — формирования общего вида главного окна при выполнении приложения и способов управления работой приложения) для каждого компонента необходимо определить его внешний вид, размеры, способ и место размещения в области окна приложения (т.е. реализовать интерфейс разработки и интерфейс выполнения).

Компоненты, доступные проектировщику на этапе разработки приложения, разбиты на функциональные подгруппы.

# Типы интерфейсов у компонентов

*Визуальные* компоненты (*элементы управления*) характеризуются наличием свойств размеров и положения в области окна и на стадии разработки приложения обычно находятся на форме в том же месте, что и во время выполнения приложения (например, кнопки, списки, переключатели, надписи). Визуальные компоненты имеют две разновидности — «оконные» и «неоконные» (графические).

- *«Оконные»* визуальные компоненты (самая многочисленная группа компонентов) — это компоненты, которые могут получать *фокус ввода* (т.е. становиться активными для взаимодействия с пользователем) и содержать другие визуальные компоненты.



- *«Неоконные»* (графические) визуальные компоненты не могут получать фокус и содержать другие визуальные компоненты (например, надписи и графические кнопки).

*Невизуальные* компоненты на стадии разработки не имеют своего фиксированного местоположения и размеров. Во время выполнения приложения некоторые из них иногда становятся видимыми (например, стандартные диалоговые окна открытия и сохранения файлов), а другие остаются невидимыми всегда (например, таблицы базы данных).

Важной характеристикой компонента являются его *свойства*. Свойства компонента — это атрибуты, определяющие его состояние и поведение. Различают три типа свойств.

Первые — свойства *времени проектирования*. Установленные для них значения будут использоваться в момент первого отображения компонента и в дальнейшем могут быть изменены во время выполнения приложения.

Вторые — *динамические* свойства. Изменением их значений можно управлять только изнутри программного кода (во время выполнения приложения).

Третьи — так называемые свойства *только-для-чтения*, которые могут быть прочитаны и использованы при выполнении программы, но не могут быть изменены.

*Второй этап* — непосредственное программирование процедур обработки событий, исходящих от компонентов. Основная задача при разработке таких процедур — запрограммировать реакцию на все возможные изменения состояний объектов.