ООП

http://delphidevelop.ru/publ/28

http://www.delphi-manual.ru/interface.php

Классификация языков программирования

- **Языки низкого уровня**
  - ***Язык Ассемблера*** (язык низкого уровня – яз.прогр. близкий к программированию в машинных кодах, используемых в работе данного процессора – разный для разных типов процессоров) Ассемблер – программа для работы с языком программирования ассемблера
- **Языки высокого уровня**
  - ***Процедурные языки*** - программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи. (Ада – ЯП разработан Мин Обороны США в военных целях для управления военными комплексами, названный в честь первого программиста Ады Лавлейз; Basic, Pascal – ЯП были предназначены для обучения программированиию; Си – язык разработан для использования в системе UNIX, портирован на другие системы)
  - ***Функциональные языки*** – в функциональном программировании процесс вычисления происходит через вычисления значений функций, в математическом понимании. (LISP)
  - ***Логические языки*** – программы (база знаний) основаны на выводе новых фактов на основе уже существующих фактов согласно заданным логическим правилам. (PROLOG) (Саша – человек, Маша – человек, Даша – человек, Саша любит спорт, Маша любит спорт, Даша любит книги, Два человека друзья если они любят одно и то же. Вопрос программе: Саша друг Маши? – да. Кто является друзьями? – Саша, Маша)
  - ***Объектно-ориентированное программирование*** – программа на ООЯП представляет собой набор процедур, описывающих взаимодействие объектов. (C++, Java, JavaScript, Delphi (Lazarus))

Основные понятия

Объект – это некая сущность, воспринимаемая человеком или машиной как единое целое

Класс – это тип, определяющий устройство и поведение объектов (это тип данных для объектов)

Приложения на ООП строятся из объектов как дом из различных кирпичиков и блоков. Основная единица в ООП – **объект**.

Свойства - какие-либо качества, параметры объекта. (Ширина, Высота)

Методы - процессы, меняющие какие-либо свойства объекта.


КЛАСС
ОБЪЕКТ
ОБЪЕКТ
ОБЪЕКТ
...
...
...
...
...
...
...
СВОЙСТВА
МЕТОДЫ
СОБЫТИЯ

Пример:

- ***Состояние объекта*** *определяется набором значений свойств в данный момент времени. Набор свойств у двух объектов класса люди одинаковые, но значения этих свойств разные.*

- **Класс** – люди
  Объект (экземпляр класса – Гаврилов, Глазырина)
  Пол: [Муж, Жен]
  Свойства:
  X: Real; (Real - это тип данных для действительных чисел, например, переменная X может быть равна 0,32 или 91,2123)
  Y: Real;
  Z: Real; - положение в пространстве
  Возраст: Integer;
  Цвет кожи: Color;
  Видимость: Да.

Методы:

(методы меняют состояние объекта, т.е. меняют значения каких-либо свойств.)
Шагнуть вперед (меняет значение X и Y)
Прыгнуть (меняет значение Z)
Стать невидимым
Стать видимым

Конкретные методы меняют свойства объекта конкретным образом. Два разных объекта отреагируют одинаково на применение одного метода.

События:

- (события отражают поведение объектов при их взаимодействии)
  Один человек толкнул другого. (Объект которого толкнули изменит свои X,Y,Z)
  День рождение
  Поздороваться
  С помощью событий можно самому описать как объект отреагирует. Два разных объекта могут отреагировать по разному на одно и то же событие.

Пример 2:

Класс – собаки

- Объект (Шарик, Бобик)
- Свойства
- Методы
- События (если собаку пнуть, то одна укусит, другая заскулит)

Придумать свои примеры классов и описать их по вышеуказанной структуре.

Шаблон описания объекта

Класс: Имя класса

Объект: Имя объекта

Свойства (перечисляем свойства)

Имя свойства: *Значение свойства*

Имя свойства: *Значение свойства*

Метод (должен менять какие-либо свойства из вышеперечисленных):

Имя метода: Какие свойства изменились

События (как объект отреагирует на внешние раздражители):

Может измениться свойство или может вызвать метод, который в свою очередь изменит какие-то свойства

Три основных принципа ООП

1. ***Полиморфизм*** – одно и то же действие, но реализация может быть различная (Например: нахождение корней уравнения, классический метод через дискриминант, теорема Виета и т.п. или решение системы уравнений классический метод или метод Гаусса)
2. ***Инкапсуляция*** – сложный механизм действий скрыт от глаз наблюдателя и выглядит как простое действие в один шаг (Например: завести машину – повернуть ключ, далее электрич импульс, искра, двигатель и пр. или скрыть объект на экране – нужно скрыть все пиксели объекта, а выглядит просто)
3. ***Наследование*** – свойства родительского класса передаются классу-наследнику, также к классу-наследнику могут придаваться свои специфические свойства. (например: класс собаки -> класс бульдог и класс овчарка –

Объект

Основой объектно-ориентированного программирования является объект.

Объект состоит из трёх основных частей:

1) Имя (например, автомобиль);

2) Состояние, или переменные состояния (например, марка автомобиля, цвет, масса, число мест и т. д.);

3) Методы, или операции, которые выполняют некоторые действия над объектами и определяют, как объект взаимодействует с окружающим миром.

Объекты

Объекты характеризуются

- ***свойствами*** (цвет, положение на экране и пр.)
- ***методами*** (действия или задачи которые выполняет объект)
- ***событиями*** (на какое событие должен реагировать объект).

Обработка событий

Многие объекты, в том числе и кнопки, должны как-то реагировать на различные события, например на пользовательское щелканье мышкой. Чтобы сделать их способными на ответные действия, нужно написать **процедуру обработки события**. Самое распространенное событие – Click (Щелчок), пользователь навел курсором на объект и щелкнул кнопкой мыши.

Структура процедур обработки событий

PROCEDURE TForm1.Button1Click( );

CONST *{описание постоянных}*

<имя постоянной>=<значение>;

VAR {определение лок. переменных}

<имя переменной>:<тип данных>;

<имя переменной>:<тип данных>;

BEGIN

{код процедуры обработки события}

END;

Классом называется составной тип данных,

членами (элементами) которого являются функции и переменные (поля).

В основу понятия класс положен тот факт, что "над объектами можно совершать различные операции".

Свойства объектов описываются с помощью переменных (полей) классов, а действия над объектами описываются с помощью подпрограмм, которые называются методами класса.

Объекты называются экземплярами класса.

Для объявления класса используется конструкция:

type
<название класса> = c l a s s (<имя класса родителя>
<поля и методы класса>
private
<поля и методы, доступные только в пределах модуля>
protected
<поля и методы, доступные только в классах-потомках>
public
<поля и методы, доступные из других модулей>
published
<поля и методы, видимые в инспекторе объектов>
end;

Структуры отличаются от классов тем, что поля структуры доступны всегда.

При использовании классов могут быть члены, доступные везде публичные (описатель public), и приватные (описатель private), доступ к которым возможен только с помощью публичных методов. Это также относится и к методам класса.

Поля, свойства и методы секции public не имеют ограничений на видимость.

Поля, свойства и методы, находящиеся в секции private, доступны только в методах класса и в функциях, содержащихся в том же модуле, что и описываемый класс. Это позволяет полностью скрыть детали внутренней реализации класса. Вызов приватных методов осуществляется из публичных.

Поля и методы, описанные до указания имени секции, являются публичными (public).

Структура проекта в Lazarus

Любой *проект* в Lazarus — это совокупность файлов, из которых создаётся единый *выполняемый файл*. В простейшем случае список файлов проекта имеет вид:

- файл описания проекта (`.lpi`);
- файл проекта (`.lpr`);
- файл ресурсов (`.lrs`);
- модуль формы (`.lfm`);
- программный модуль (`.pas`);

После *компиляции* программы из всех файлов проекта создаётся единый выполняемый файл с расширением .exe, имя этого файла совпадает с именем проекта.

Программный модуль, или просто модуль — это отдельно компилируемая программная единица, которая представляет собой набор типов данных, констант, переменных, процедур и функций. Любой модуль имеет следующую структуру:

```
unit имя_модуля;      //Заголовок модуля.
interface
//Раздел описаний.
implementation
//Раздел реализаций.
end.                  //Конец модуля.
```

Заголовок модуля — это зарезервированное слово `unit`, за которым следует имя модуля и точка с запятой. В разделе описаний, который открывается служебным словом `interface`, описывают *программные элементы* — типы, классы, процедуры и функции:

```
interface
uses список_модулей;
type список_типов;
const список_констант;
var список_переменных;
procedure имя_процедуры;
...
function имя_функции;
...
```

Раздел `implementation` содержит *программный код*, реализующий механизм работы описанных программных элементов (тексты процедур обработки событий, процедуры и функции, созданные программистом). *Процедуры* и *функции* в Lazarus также построены по модульному принципу[1].

Наряду с визуальными приложениями, Lazarus позволяет разрабатывать и обычные консольные приложения, которые также могут быть созданы в оболочке Free Pascal и в текстовом редакторе Geany. Авторы настоятельно рекомендуют начинать изучение программирования именно с создания консольных приложений. Поэтому рассмотрим подробно структуру консольного приложения.

```
program имя_программы;
uses modul1, modul2, ..., moduln;
const описания_констант;
type описания_типов;
var описания_переменных;
begin
операторы_языка;
end.
```

Модуль кода (unit) делится на две части – интерфейс (**interface) и** реализацию
(**implementation).**
Раздел interface:
– uses: подключаемые модули, в которых содержатся используемые процедуры, функции, классы и т.п. Их список формируется автоматически в
зависимости от добавленных в форму компонентов.
– type: описания типов. Автоматически в этом разделе описан класс вашей
формы- перечислены все компоненты расположенные на форме
– var: описание глобальных переменных. Автоматически тут описана
переменная типа описанного класса формы. В этой переменной во время работы программы хранится указатель на экземпляр формы.

Раздел implementation:

– Раздел содержит описание реализаций процедур и функций.

Автоматически тут

создаются обработчики событий. Здесь описывается реализация

необходимых в

программе процедур, функций и методов классов.

Раздел implementation предназначен не только для обработчиков событий. В этот раздел добавлять в него свои собственные функции и процедуры .

Существует также важная тонкость использования собственных функций: все обработчики событий имеют в своем названии имя формы:

procedure TForm1.FormCreate(Sender: TObject);

Это указывает на то, что работают с классом формы TForm1 и описывают методы для нее.

1. Что такое ООП?
2. Назовите 3 источника ООП.
3. Назовите 3 составные части ООП.
4. Что такое «наследование»?
5. Что такое класс?
6. Что такое объект?
7. Что характеризует объект?
8. Что нужно сделать, чтобы объекты стали способными на ответные действия?
9. Приведите пример объектов в ООП.
10. Перечислите название окон в Lazarus.