# Основные принципы ООП

*Основные принципы ООП* – это методология программирования, основана на представлении программы в виде совокупности объектов.

*Каждый объект* – это экземпляр определённого класса. Классы образуют иерархию наследований.

Принципы ООП:

*Абстракция* – это выделение главного, наиболее значимой характеристики объекта и отбрасывание второстепенных.

*Полиморфизм* – это решение одной и той же задачи, разными методами и способами.

*Наследование* – это способность объекта или класса быть созданным на основе другого. Наследственные отношения класса часто определяет их иерархию. *Наследование* – это главный механизм повторного использования кода.

*Инкапсуляция* – это разрешения объектов или класса для разграничения доступа к ним.

## Инкапсуляция

Данные и функции для их обработки объединяются в один тип данных, класс, структуру.

Реализация скрывается при помощи модификатора доступа.

*private* – доступ только внутри класса. (в C# по умолчанию)

*protected* – доступ классу и его наследникам.

*public* – доступ всем.

*internal* – доступ всем уровням сборки.

Поля рекомендуется делать закрытыми

private string \_name;

Функции класса лучше делать закрытыми и открывать по мере необходимости.

## Наследование

В С# создание нового класса может происходить на основе только одного класса.

Новому классу доступно то, что и в родительском :protected и public

Класс Object – родительски для всех типов данных в С#.

Классы образуют иерархию типов.

## Полиморфизм

*pole* – много

*morphe* – форма

Типы полиморфизма:

Ad-hoe

Перегрузки (**overload**) функции и методов вызывают разные системные параметры.

Все конструкции являются перегруженными, если у них одинаковое имя и разные типы и/или кол-во параметров.

Тип возврата в перегрузке не учитывается

Компилятор видит Sum(int, int)

Перегрузка смотрится в пределах класса.

*Перегрузка подтипов* – это объекты классов могут быть приведены в родительском классе.

Переопределение(**override**) функции и методов в данном классе можно переопределить виртуальные методы родительского класса.

virtual

override – переопределение метода может использовать поведение их родительского класса.

Поведение родительского класса можно получить, использовав up cast (**up casting** – проверяющая поведение, то есть приведение объектов дочернего типа к одному родительскому объекту с классом)

var Murka = new Cat();

Явное приведение:

Animal barsic = (Animal)murka;

Animal barsic = murka as Anumal;

Animal barsik = murka;

Animal barsic = new Cat();

object x = murka

object y = (object)murka

*down cost/downcosting* – понижающее приведение. Приведение объекта к его предшественникам.

Возможны ошибки если у объекта производного класса будут вызваны методы, отсутствующие в родительском классе.

Cat barsik = Animal as Cat

чтобы выполнить проверку или преобразование типов используется **is** и **as**

is проверяет: совместен ли тип переменных и вариант с указанием типов.

Общая форма: *Объект is тип*

**as** явно преобразует выразив в указателе тип, если типы совмещены. Если преобразование невозможно, то возвращается значение NULL, никогда не создаёт исключений.

*Параметрический полиморфизм* – это использование обобщений.

void Print(T obj) {…}

*Обобщение* (универсальный шаблон)– позволяет использовать параметры обобщенного типа, использование одного алгоритма для разных программ.