- “Сбор” свойств и методов в классе называется Инкапсуляция.
- Наследование – это передача “по наследству” свойств и методов.
- Предок у класса может быть только один, а потомков много (т.е. нет множественного наследования).
- Классы-потомки получат в наследство все свойства и методы от классов-родителей.
- Унаследованные методы можно переписать заново в классах-потомках.
- К методу родителя можно обратиться с помощью конструкции parent::method().
- Конструктор (__construct()) – это “магический” метод, который выполняется автоматически при создании объекта класса.
- Запись $methodName = new ClassName('argument'); означает передачу 'argument' в конструктор.
- methodName(…$args) означает, что переданные в данный метод аргументы будут храниться в массиве $args; еще вариант: methodName($color, …$args).
- Все объекты в PHP имеют тип object.
- Оператор instanceof ($obj instanceof SomeClass) проверяет, является ли объект объектом заданного класса. Также, такая проверка даст true, если имя класса будет одним из предков класса данного объекта.
- Type hinting (Контроль типа). Это указание ожидаемого типа аргумента. function sendMail(User $user, $message) {}
- Буфер вывода. Всё, что мы выводим – уже нельзя вернуть назад. Однако, в PHP есть способ взять вывод в “буфер”, чтобы потом с ним работать.

  ```
  ob_start();
  echo 'Hello!';
  $output = ob_get_contents();
  ob_end_clean();

  echo $output; // Hello
  ```
- Функцию *__autoload()* нужно только объявить. Вызывать ее не требуется. Название функции начинается с двух знаков подчеркивания – это признак “магии” в PHP. Функция будет вызвана автоматически, когда код встретит ранее неизвестное имя класса. Это имя будет передано функции. Ее задача – подключить файл, где этот класс определен.

- Абстрактный класс – это класс, который не позволяет создание объектов класса.

  В абстрактном классе могут быть как обычные свойства и методы, так и абстрактные методы:
  - они содержат только заголовок;
  - отсутствует собственно код метода;
  - дочерний класс обязан такой метод реализовать в точном соответствии с заголовком из абстрактного класса, иначе будет ошибка.

  Любой класс, в котором есть хотя бы один абстрактный метод, обязан быть абстрактным.

  Ключевое слово “self” значит “тот класс, где оно написано”:

```
abstract class Model {
	public static $table = 'some table';
	public static function getTable() {
		return self::$table;
	}
}

class User extends Model {
	public static $table = 'users';
}

echo User::getTable();  // some table
```

  Такое происходит потому, что “self” – это “раннее” связывание, на этапе разбора кода.

  Чтобы связаться с тем классом, в котором будет выполняться код, используйте static вместо self: return static::$table

  Это называется Late Static Binding (позднее статическое связывание), на этапе выполнения.

- Интерфейс – это специальная сущность языка PHP, своего рода “контракт”, который обязан выполнить класс.

  Интерфейс содержит в себе описание публичных методов, (*и констант*) которые обязан иметь класс, реализующий интерфейс.

```
interface Orderable {
    public function getPrice();
    public function getWeight();
}

class Item implements Orderable {
    public function getPrice() {
        //
    }

    public function getWeight() {
        //
    }
}
```

  - Интерфейсы могут, как классы, наследоваться друг от друга.

```
interface HasPrice {
    //
}

interface Orderable extends HasPrice, HasWeight {
    //
}
```

  - Класс может реализовывать несколько интерфейсов.

```
class Item implements HasPrice, HasWeight {
    //
}
```

- Трейт – это специальная сущность языка PHP, своего рода “заготовка”, которую можно вставить в класс. Трейт может содержать в себе методы и свойства, динамические и статические.

  При компиляции вашей программы текст трейта будет вставлен в текст класса.

```
trait DateTime {
    protected $started;
    protected $finished;

    public function getDuration() {
        return $finished - $started;
    }
}

class Order {
     use DateTime;
}
```

- В PHP выделяют три стадии исполнения:
  1. Парсер (специальный движок), который превращает код в специальное так называемое “абстрактное синтаксическое дерево”, т.е. разбирает код и понимает что там написано;
  2. Компиляция. Когда “абстрактное синтаксическое дерево” превращается в байт-код;
  3. Исполнение. Когда код запускается на исполнение.
- Исключение – это ситуация, при которой дальнейшее нормальное выполнение невозможно или бессмысленно. Это ситуация, возникающая во время исполнения программы. Примеры исключений:
  - Деление на ноль;
  - Разрыв соединения с БД;
  - Отсутствие нужного файла;
  - Неверный пароль в форме входа…

- Анонимная функция – это функция, которая в отличие от обычной, не имеет имени.

  ```
  function ($x) {
      return $x * 2;
  }
  ```

  Анонимную функцию можно присвоить переменной и так вызвать:

  ```
  $test = function ($x) {
      return $x * 2;
  };

  echo $test(2); // 4
  ```

  Анонимная функция – это “объект первого класса”.

  Объектами первого класса (first-class object) в контексте конкретного языка программирования называются элементы, которые могут быть переданы как параметр, возвращены из функции, присвоены переменной.

  Лямбда-выражение в программировании — специальный синтаксис для определения функциональных объектов, заимствованный из λ-исчисления. Применяется как правило для объявления анонимных функций по месту их использования, и обычно допускает замыкание на лексический контекст, в котором это выражение использовано.
- Замыкание – это способ передачи функции контекста.

  Замыкание позволяет “замкнуть” на анонимную функцию какое-либо значение из родительского контекста. Значение замыкания “фиксируется” на момент создания анонимной функции.

  ```
  $prefix = 'project';
  $test = function ($name) use ($prefix) {
      return $prefix . '_' . $name;
  }
  echo $test('users'); // project_users
  ```

- Генераторы – это способ создания итераторов.

  Выглядит генератор как функция. Но вместо return – yield. Единственное применение генератора, это его использование в контексте foreach.

```
function generate() {
    for ($x=1; $x<=10; $x++) {
        yield $x;
    }
}

foreach (generate() as $number) {
    echo $number; // 12345678910
}
```

  Главный смысл: использование памяти. Даже в случае последовательности из 1000 элементов требуется память для одного.

- PSR (PHP Standards Recommendations) – это стандарты кодирования на PHP.
- ORM – Object Relational Mapping
  или, иначе говоря, принцип отображения объектов реального мира (и их связей) на объекты вашего языка программирования.

  ООП прекрасно подходит для реализации этого шаблона проектирования:
  - Класс описывает какие объекты данных у нас могут быть;
  - Сами данные представлены объектами заданных классов;
  - И мы сразу можем определить “поведение данных” в виде методов этих объектов;
- БД – совокупность хранимых данных.
  СУБД – система управления этими данными.

  Реляционные базы данных:
  - Все данные хранятся в таблицах;
  - Каждая запись – это строка таблицы;
  - Столбцы таблицы – это поля. Поля имеют тип.
- PDO – PHP Data Objects.
  Класс для PHP, предоставляющий разработчику простой и универсальный интерфейс для доступа к различным базам данных.
- Active Record – это архитектурный паттерн “Активная запись”.
  - Записи в БД соответствует объект в языке программирования;
  - Запись в базе данных может “сама себя” сохранить и удалить, используя методы объекта.

  ```
  $user = User::findById(1);
  $user->password = 'password';
  $user->save();

  $user = User::findByEmail('test@mail.com');
  $user->delete();
  ```
- SPL: Standard PHP Library.
  Стандартная библиотека PHP — коллекция классов и интерфейсов для решения стандартных проблем в PHP. Основное содержание библиотеки — классы-итераторы, решающие задачи итерации по каталогу, массиву, дереву XML.