Наследование

sd@freematiq.com

static class

```
class User {
   private static $passLen = 6;
   static function register($name, $password) {
      $this-> // error!
   }

   public getLen() {
      return $passLen; // error!
   }
}
```

self

```
class User {
   private static $passLen = 6;
   static function register($name, $password) {
      if (self::$passLen < mb_strlen($password))
         { … }
   }

   static public getCount() {
      return User::$passLen;
   }
}
```

Магические методы

Не статичные

- __get($name); - получение
- __set($name, $value); - установка
- __call($name, $args); - вызов метода

Если такие методы/свойства существует - вызова магического метода не произойдёт

clone

```
$user2 = clone $user;
```

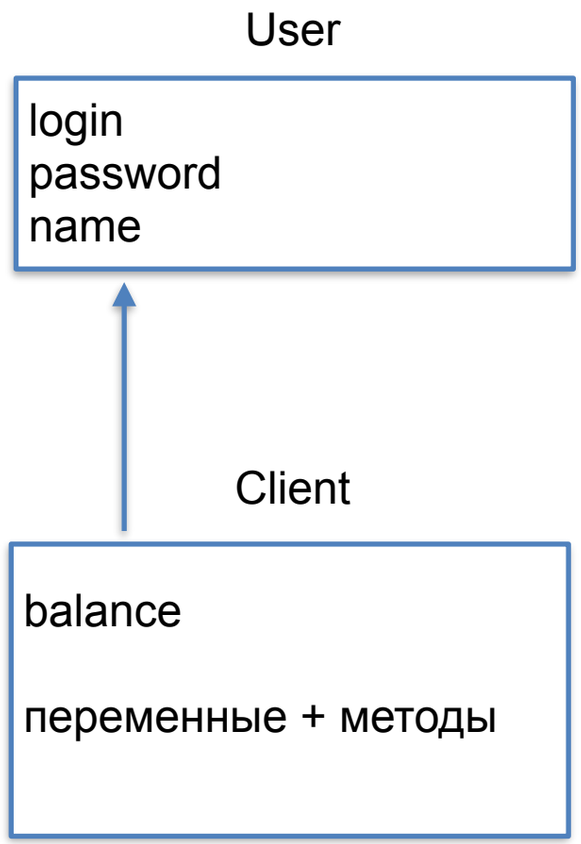
```
__clone();
```

- Копирование побитно
- Можно запретить, если сделать метод private

Наследование

- Создание нового класса на основе существующего (базового)
- Расширение функционала, добавление методов и переменных
- Переопределение работы методов
- Для создания однотипных объектов

Наследование



?

Необходимо сделать наследование для фигур

Прямоугольник и квадрат.

Что от чего наследуется?

extends

```
class Manager extends User { … };

__construct($name, $password) {
    parent::__construct($name, $password);
};
```

parent - для вызова родительского метода

Полиморфизм

```
class Manager extends User { … };

User $users[];

$users[] = new User();
$users[] = new Manager();

foreach($users as $user) {
  $user->generate_report();
}
```

Позднее статическое связывание

Нужно, для переопределения статичного метода

```
<?php
class A {
    public static function who() {
        echo __CLASS__;
    }
    public static function test() {
        self::who();
    }
}

class B extends A {
    public static function who() {
        echo __CLASS__;
    }
}

B::test();
?>

A
```

Позднее статическое связывание

Нужно, для переопределения статичного метода

```
<?php
class A {
    public static function who() {
        echo __CLASS__;
    }
    public static function test() {
        static::who();
    }
}

class B extends A {
    public static function who() {
        echo __CLASS__;
    }
}

B::test();
?>
```

```
B
```

Абстрактные методы и классы

- Абстрактный метод нельзя вызвать, если он не переопределен в производном классе
- Объект абстрактного класса невозможно создать
- Если класс содержит абстрактный метод, то он считается абстрактным
- abstract public function getSquare();

Интерфейсы

- Не содержат свойств!
- Не содержат реализаций методов!
- Могут содержать константы
- interface InterfaceName { };
- Класс, реализующий интерфейс обязан содержать реализацию всех методов интерфейса - контракт
- Но если реализованы не все, то класс должен быть абстрактным
- Интерфейсы могут наследоваться с помощью extends, несколько раз!

Interface

```
interface UserIdentity {
  public function getId();
};

class User implements UserIdentity { … };
```

Можно имплементировать несколько интерфейсов

Итого

- Для общих данных используйте static
- self — обращение к классу
- __get, __set, __call
- Наследование для расширения
- Полиморфизм для динамики
- Абстрактные для архитектуры
- Интерфейсы - контракты

Домашняя работа

- Добавить разных классов
- Наследовать друг от друга
- Сделать вывод баланса для каждого типа: если пользователь администратор, вывести баланс для всех пользователей