Класс и объект -  это базовые понятия объектно-ориентированного программирования (ООП). **Объект** (object) – это структурированная переменная, которая содержит всю информацию о реализуемом в программе понятии. **Класс** (class) – это описание таких объектов и действий, которые можно с ними выполнять.

 В PHP синтаксис класса определяется следующим образом:

class имя\_класса {  
   var $имя\_свойства1;  
   var $имя\_свойства2="значение свойства";

   ...

   var $имя\_свойстваN;

  function имя\_метода1 ($параметр1, параметр2, ..., параметрN) {  
    [код метода]  
  }  
}

Т.о. имена свойств класса определяются внутри класса с помощью ключевого слова var. Далее идет наименование свойства. Если определенному свойству необходимо присвоить значение по умолчанию при создании объекта, то значение описывается через "=" после имени свойства. Методы, применимые к объектам данного класса, описываются [функциями](http://phpprogs.ru/article/funkcii-v-php). Доступ к свойствам и методам внутри класса осуществляется с помощью переменной $this.

Для примера создадим класс, описывающий человека.

class Human {

  var name;

  var city;

  var age;

  function setData($name, $age, $city) {

    $this->name = $name;

    $this->age = $age;

    $this->city = $city;

  }

  function getData() {

    echo "Имя: ".$this->name."<BR>";

    echo "Возраст: ".$this->age."<BR>";

    echo "Город проживания: ".$this->city."<BR>";

  }

}

И так, мы создали класс Human, характирезующий человека. Класс состоит из 3-х свойств: имя, возраст и город проживания. Также в классе присутствуют 2 метода, позволяющие нам записать данные о человеке в сваойства класса и затем вывести их.

Теперь перейдем от теории к практике. Для объявления объекта необходимо использовать оператор new. Чтобы добраться до определенного свойства класса, необходимо использовать синтаксис имя\_объекта->имя\_свойства или же имя\_объекта->имя\_метода() при обращении к методу класса.

  $data = new Human;

  $data->setData("Вася Пупкин", 25, "Москва");

  $data->getData();

  После выполнения на экране появится:

  Имя: Вася Пупкин

  Возраст: 25

  Город проживания: Москва

Создадим свой **класс по работе с базой данных**. Данный класс позволит нам реализовывать в наших программах почти все функции MySQL.

Назовем наш класс **sql\_db** и создадим для него конструктор, который будет принимать настройки соединения:

class sql\_db {  
    var $db\_connect\_id;  
    var $user;  
    var $password;  
    var $server;  
    var $database;  
  
    function sql\_db($sqlserver, $sqluser, $sqlpassword, $database) {  
        $this->user = $sqluser;  
        $this->password = $sqlpassword;  
        $this->server = $sqlserver;  
        $this->dbname = $database;  
          
        $this->db\_connect\_id = mysql\_connect($this->server, $this->user, $this->password);  
  
        if( $this->db\_connect\_id ) {  
            if( $database != "" ) {  
                $this->dbname = $database;  
                $dbselect = mysql\_select\_db($this->dbname);  
  
                if( !$dbselect ) {  
                    mysql\_close($this->db\_connect\_id);  
                    $this->db\_connect\_id = $dbselect;  
                }  
            }  
  
            return $this->db\_connect\_id;  
        }  
        else {  
            return false;  
        }  
    }  
}

Разберем этот код. Изначально объявляем свойства нашего класса:

db\_connect\_id - идентификатор соединения  
user - пользователь БД  
password - пароль к БД  
server - сервер БД  
database - имя базы данных

Подсоединение к базе данных происходит путем вызова функции **mysql\_connect**, принимающей параметры сервера, пользователя и пароля. Далее обязательно проверяем произошло ли соединение с базой данных, если да, то выбираем базу данных через функцию **mysql\_select\_db**, в противном случае возвращаем ошибку.

Теперь мы имеем соединение с базой данных, но этого мало, реализуем функцию отсылки запросов к БД.

    function sql\_query($query = "") {  
        unset($this->query\_result);          
        $this->query\_result = mysql\_query($query, $this->db\_connect\_id);  
  
        if( $this->query\_result ) {  
            return $this->query\_result;  
        }  
        else {  
            $this->query\_err .= "<br>n" . $this->sql\_error\_msg()    ;  
            return false;  
        }  
    }

На вход нашей функции sql\_query передается запрос, которая выполняет mysql-функцию **mysql\_query**. Здесь также проверяем что возвращает функция, если есть результат, то возвращаем его пользователю, в противном случае возвращаем ошибку.

Для связки подключение->запрос->ответ нам не хватает одной функции - это функция обработки ответа от MySQL-сервера. Создадим 2 функции, которые будут отдавать [ассоциативные массивы](http://phpprogs.ru/article/arrays_2part). Функция sql\_fetchrow будет обрабатывать ряд запроса и возвращать текущий ассоциативный массив, а функция sql\_fetchrowset возращать полный ответ от сервера, т.е. многомерный ассоциативный массив.

    function sql\_fetchrow($query\_id = 0) {  
        if( !$query\_id ) {  
            $query\_id = $this->query\_result;  
        }  
  
        if( $query\_id ) {  
            return @mysql\_fetch\_array($query\_id, MYSQL\_ASSOC);  
        }  
        else {  
            return false;  
        }  
    }  
  
    function sql\_fetchrowset($query\_id = 0) {  
        if( !$query\_id ) {  
            $query\_id = $this->query\_result;  
        }  
  
        if( $query\_id ) {  
            unset($this->rowset);  
            unset($this->row[$query\_id]);  
  
            while($this->rowset = @mysql\_fetch\_array($query\_id, MYSQL\_ASSOC)) {  
                $result[] = $this->rowset;  
            }  
  
            return isset($result) ? $result : false ;  
            unset($result);  
        }  
        else {  
            return false;  
        }  
    }

Также есть другие функции, которые нам помогут в работе: sql\_numrows - функция отдает количество возвращаемых БД-сервером рядов (для SELECT), sql\_affectedrows - количество записей, которые были затронуты последним запросом (Для UPDATE, DELETE, INSERT), sql\_numfields - количество возвращаемых полей в ответе, sql\_fieldname - возвращает наименование поля, порядковый номер которого задается параметом offset, sql\_fieldtype - возвращает тип поля, порядковый номер которого задается параметом offset, sql\_rowseek - смещение указателя на rownum, sql\_nextid - возвращает последний добавленный ID командой INSERT, sql\_freeresult - освобождение памяти, sql\_error - возвращает код последней ошибки и ее описание, sql\_close - закрыть соединение с базой данных.

Теперь давайте попробуем поработать с нашим классом. Для примера будем использовать базу данных, созданную в уроках про [phpMyAdmin](http://phpprogs.ru/article/mysql-rabota-s-phpmyadmin) и данные, которые мы добавляли в статье про [основные типы запросов к MySQL](http://phpprogs.ru/article/mysql-yazyk-sql). Создадим объект $db.

<?php  
    include "mysql.php";  
    $db = new sql\_db("localhost", "root", "", "test");  
?>

Теперь попробуем из таблицы books выбрать все записи двумя способами, с помощью функций sql\_fetchrow и sql\_fetchrowset

<?php  
    $db->sql\_query("select \* from books;");

    while ($row = $db->sql\_fetchrow()) {

      echo "Автор: {$row['autor']}. Книга: {$row['name']}<BR/>";

    }  
?>

<?php  
    $db->sql\_query("select \* from books;");

    $rowset = $db->sql\_fetchrowset();

    foreach ($rowset as $row) {

      echo "Автор: {$row['autor']}. Книга: {$row['name']}<BR/>";

    }  
?>

Чтобы узнать сколько записей возвращается из запроса, выполним следующий код:

<?php  
    $db->sql\_query("select \* from books;");  
    echo "Количество записей: ".$db->sql\_numrows()."<BR/>";

?>

Узнаем ID последней добавленной записи

<?php  
    $db->sql\_query("INSERT INTO (name, autor) VALUES ('Рога и копыта', 'Вася Пупкин')");  
     echo "Последний ID: ".$db->sql\_nextid()."<BR/>";

?>