```php
<?php
class PayPal {

    // environment
    private $_environment = 'sandbox';

    // urls
    private $_url_production = 'https://www.paypal.com/cgi-bin/webscr';
    private $_url_sandbox = 'https://www.sandbox.paypal.com/cgi-bin/webscr';

    // url to be used
    private $_url;

    // transaction type :
    // _xclick = buy now buttons
    // _cart = basket
    private $_cmd;

    // all products array
    private $_products = array();

    // all input fields array
    private $_fields = array();

    // your paypal id
    private $_business = 'sebast_1291569317_biz@gmail.com';

    // page style
    private $_page_style = 'Test';

    // return url
    private $_return;

    // cancel url
    private $_cancel_payment;

    // notify url (IPN)
    private $_notify_url;

    // currency code
    private $_currency_code = 'GBP';

    // tax / vat amount for _cart
    public $_tax_cart = 0;

    // tax / vat amount for _xclick
    public $_tax = 0;

    // pre-populating checkout pages
    // address1 *, address2, city *, state *, zip *
    // country *, email *, first_name *, last_name *
    // you have to have all required fields filled in - otherwise it won't work
    public $_populate = array();

    // data received from paypal
    private $_ipn_data = array();
    // path to the log file for ipn response
    private $_log_file = null;

    // result of sending data back to paypal after ipn
    private $_ipn_result;



    public function __construct($cmd = '_cart') {

        $this->_url = $this->_environment == 'sandbox' ?
                        $this->_url_sandbox :
                        $this->_url_production;

        $this->_cmd = $cmd;

        $this->_return = SITE_URL."/?page=return";
        $this->_cancel_payment = SITE_URL."/?page=cancel";
        $this->_notify_url = SITE_URL."/?page=ipn";
        $this->_log_file = ROOT_PATH.DS."log".DS."ipn.log";
    }



    public function addProduct($number, $name, $price = 0, $qty = 1) {

        switch($this->_cmd) {

            case '_cart':
            $id = count($this->_products) + 1;
            $this->_products[$id]['item_number_'.$id] = $number;
            $this->_products[$id]['item_name_'.$id] = $name;
            $this->_products[$id]['amount_'.$id] = $price;
            $this->_products[$id]['quantity_'.$id] = $qty;
            break;
            case '_xclick':
            if (empty($this->_products)) {
                $this->_products[0]['item_number'] = $number;
                $this->_products[0]['item_name'] = $name;
                $this->_products[0]['amount'] = $price;
                $this->_products[0]['quantity'] = $qty;
            }
            break;
        }
    }




    private function addField($name = null, $value = null) {
        if (!empty($name) && !empty($value)) {
            $field  = '<input type="hidden" name="'.$name.'" ';
            $field .= 'value="'.$value.'" />';
            $this->_fields[] = $field;
        }
    }
```

```php
    private function standardFields() {
        $this->addField('cmd', $this->_cmd);
        $this->addField('business', $this->_business);
        if ($this->_page_style != null) {
            $this->addField('page_style', $this->_page_style);
        }
        $this->addField('return', $this->_return);
        $this->addField('notify_url', $this->_notify_url);
        $this->addField('cancel_payment', $this->_cancel_payment);
        $this->addField('currency_code', $this->_currency_code);
        $this->addField('rm', 2);

        switch($this->_cmd) {
            case '_cart':
            if ($this->_tax_cart != 0) {
                $this->addField('tax_cart', $this->_tax_cart);
            }
            $this->addField('upload', 1);
            break;
            case '_xclick':
            if ($this->_tax != 0) {
                $this->addField('tax', $this->_tax);
            }
            break;
        }
    }



    private function prePopulate() {
        if (!empty($this->_populate)) {
            foreach($this->_populate as $key => $value) {
                $this->addField($key, $value);
            }
        }
    }



    private function processFields() {
        $this->standardFields();
        if (!empty($this->_products)) {
            foreach($this->_products as $product) {
                foreach($product as $key => $value) {
                    $this->addField($key, $value);
                }
            }
        }
        $this->prePopulate();
    }
    private function getFields() {
        $this->processFields();
        if (!empty($this->_fields)) {
            return implode("", $this->_fields);
        }
    }




    private function render() {
        $out  = '<form action="'.$this->_url.'" method="post" id="frm_paypal">';
        $out .= $this->getFields();
        $out .= '<input type="submit" value="Submit" />';
        $out .= '</form>';
        return $out;
    }




    public function run($transaction_id = null) {
        if (!empty($transaction_id)) {
            $this->addField('custom', $transaction_id);
        }
        return $this->render();
    }




    private function validateIpn() {

        $hostname = gethostbyaddr($_SERVER['REMOTE_ADDR']);

        // check if post has been received from paypal.com
        if (!preg_match('/paypal\.com$/', $hostname)) {
            return false;
        }

        // get all posted variables and put them to array
        $objForm = new Form();
        $this->_ipn_data = $objForm->getPostArray();

        // check if email of the business matches the email recived
        // in post from IPN
        if (
            !empty($this->_ipn_data) &&
            array_key_exists('receiver_email', $this->_ipn_data) &&
            strtolower($this->_ipn_data['receiver_email']) !=
            strtolower($this->_business)
        ) {
            return false;
        }
        return true;
    }


    private function getReturnParams() {

        $out = array('cmd=_notify-validate');
        if (!empty($this->_ipn_data)) {
            foreach($this->_ipn_data as $key => $value) {
                $value = function_exists('get_magic_quotes_gpc') ?
                            urlencode(stripslashes($value)) :
                            urlencode($value);
                $out[] = "{$key}={$value}";
            }
        }
        return implode("&", $out);
    }




    private function sendCurl() {

        $response = $this->getReturnParams();

        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $this->_url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $response);
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_HTTPHEADER, array(
            "Content-Type: application/x-www-form-urlencoded",
            "Content-Length: " . strlen($response)
        ));
        curl_setopt($ch, CURLOPT_VERBOSE, 1);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
        curl_setopt($ch, CURLOPT_TIMEOUT, 30);

        $this->_ipn_result = curl_exec($ch);
        curl_close($ch);
    }





    public function ipn() {

        if ($this->validateIpn()) {

            $this->sendCurl();

            if (strcmp($this->_ipn_result, "VERIFIED") == 0) {

                $objOrder = new Order();
                // update order
                if (!empty($this->_ipn_data)) {

                    $objOrder->approve(
                        $this->_ipn_data,
                        $this->_ipn_result
                    );

                }

            }

        }

    }

}
```