Name : Mayur Purushvani

Throwing an Exception :

Exceptions are used to change the normal flow of a script if a specified error occurs.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

Example :

```php
<?php

function devide ($num1, $num2)
{
    if($num2 == 0) {
        throw new Exception('Cannot devide by zero');
    }
    else {
        return $num1/$num2;
    }
}

devide(10,0);
?>
```

Try catch :

1. try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
2. throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
3. catch - A "catch" block retrieves an exception and creates an object containing the exception information

Example :

```php
<?php

$age = 16;

try {
    if($age > 18) {
        echo 'Old enought';
    }
    else {
        throw new Exception('Not old enought.');
    }
}
catch(exception $e) {
    echo 'Error : '.$e->getMessage();
}
?>
```

<u>Custome Exceptions Example :</u>

```php
<?php


$mysql_host = 'localhost';
$mysql_user = 'root';
$mysql_pass = '';
$mysql_db = 'practice';

class ServerException extends Exception
{
    public function showSpecific()
    {
        return 'Error thrown on line ' . $this->getLine() . ' in ' . $this->getFile();
    }
}

class DatabaseException extends Exception
{
    public function showSpecific()
    {
        return 'Error thrown on line ' . $this->getLine() . ' in ' . $this->getFile();
    }
}

try {
    $con = @mysqli_connect($mysql_host, $mysql_user, $mysql_pass);
    if (!$con) {
        throw new ServerException();
    } else if (!@mysqli_select_db($con, $mysql_db)) {
        throw new DatabaseException();
    } else {
        echo 'Connected!';
    }
} catch (ServerException $se) {
    echo $se->showSpecific();
} catch (DatabaseException $de) {
    echo $de->showSpecific();
}
?>
```

Access Modifier :

- public - the property or method can be accessed from everywhere. This is default
- protected - the property or method can be accessed within the class and by classes derived from that class
- private - the property or method can ONLY be accessed within the class

Constants :

```php
<?php

class circle {
    const pi = 3.14;

    public function Area($radius) {
        return self::pi * ($radius*$radius);
    }
}

$circle = new circle;
echo $circle->Area(100);
?>
```

Constructor :

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a `__construct()` function, PHP will automatically call this function when you create an object from a class.

Example :

```php
<?php

class Example {
    public function __construct($something) {
        $this->SaySomething($something);
    }
    public function SaySomething($something) {
        echo $something;
    }
}

$example = new Example('Some text here');
?>
```

Multiple Instances :

We have created below 2 instances named ‘mayur’ and ‘max’.

We can access the methods of same class with two different instances.

```php
<?php


class BankAccount
{
    public $balance = 0;

    public function DisplayBalance() {
        return 'Balance : '.$this->balance;
    }
    public function Withdraw($amount) {
        if(($this->balance) < $amount) {
            echo 'Not enought Money!';
        } else {
            $this->balance = $this->balance - $amount;
        }
    }
    public function deposit($amount) {
        $this->balance = $this->balance + $amount;
    }

}

$mayur = new BankAccount;
echo 'Mayur<br>';
$mayur->deposit(1000);
echo $mayur->DisplayBalance()."<br>";
$mayur->Withdraw(16);
echo $mayur->DisplayBalance()."<br>";


$max = new BankAccount;
echo 'Max<br>';
$max->deposit(100);
echo $max->DisplayBalance()."<br>";
$max->Withdraw(16);
echo $max->DisplayBalance();

?>
```

Extends Example :

```php
<?php


class BankAccount
{
    public $balance = 0;

    public function DisplayBalance()
    {
        return 'Balance : ' . $this->balance;
    }
    public function Withdraw($amount)
    {
        if (($this->balance) < $amount) {
            echo 'Not enought Money!';
        } else {
            $this->balance = $this->balance - $amount;
        }
    }
    public function deposit($amount)
    {
        $this->balance = $this->balance + $amount;
    }
    public function settype($input)
    {
        $this->type = $input;
    }
}

class SavingAccount extends BankAccount
{
}

$mayur = new BankAccount;
$mayur->setType('18-25 Current');
$mayur->deposit(100);
$mayur->Withdraw(20);
echo $mayur->DisplayBalance() . "<br>";

$mayur_saving = new SavingAccount;
$mayur_saving->settype('Super Saving');
$mayur_saving->deposit(3000);

echo $mayur->type . ' has ' . $mayur->DisplayBalance() . "<br>";
echo $mayur_saving->type . ' has ' . $mayur_saving->DisplayBalance() . "<br>";
?>
```

Database Connection in OOP Example :

```php
<?php

class DatabaseConnect
{
    public function __construct($db_host, $db_username, $db_password)
    {
        echo 'Attempting connection...';
        if (!@$this->Connect($db_host,  $db_username, $db_password)) {
            echo 'Connection failed!';
        }
        else if($this->Connect($db_host, $db_username, $db_password)) {
            echo 'Connected to '.$db_host;
        }
    }
    public function Connect($db_host,$db_username,$db_password){
        if(!mysqli_connect($db_host, $db_username, $db_password)) {
            return false;
        }
        else {
            return true;
        }
    }
}
$connection = new DatabaseConnect('localhost','root','');
?>
```