Date
17/02/2017

```
abstract class B extends A
{
    function sayHello()
    {
        echo '<br>I am inside sayHello function.';
    }
    private function sayHi()
    {
        echo '<br>I am inside sayHi function.';
    }
}
class C extends B
{
    function setUp()
    {
        echo '<br> I am inside child setUp function.';
        parent::setUp();
        /* $this->setUp(); // If it executes, trap in
            an infinite loop as it calls itself i.e.
            setUp() */
        /* But if this setUp() does not exist in
            current class C, then $this->setUp() can
            be called from below function sayHi() and
            it will present the same results as parent::
                                        setUp() */
    }
    function sayHi()
    {
        echo '<br> I am inside child sayHi function.';
    }
}
```

Date
17/02/2017

```
$abc = new C();
$abc->setUp();
$abc->sayHi();
```

Output:

```
I am inside child setUp function.
I am inside setUp function. A C
I am inside child sayHi function.
```

Imp fact about chain of interface, abstract and normal class

```
interface A
{
    public function infa();
}
interface B
{
    public function infb();
}
interface C extends A,B
{
    public function infc();
}
interface D
{
    public function infd();
}
```

Date
17/02/2017

```
abstract class E implements C, D
{
    public abstract function abstracte();
    public abstract function abstractee();
}
abstract class F extends E
{
    public abstract function abstractf();
}

class G extends F
{
    public function infa() { echo '<br>infa'; }
    public function infb() { echo '<br>infb'; }
    public function infc() { echo '<br>infc'; }
    public function infd() { echo '<br>infd'; }
    public function abstracte() { echo '<br>abstracte'; }
    public function abstractee() { echo '<br>abstractee'; }
    public function abstractf() { echo '<br>abstractf'; }
}
```

```
$abc = new G();
$abc->infa();
$abc->infb();
$abc->infc();
$abc->infd();
$abc->abstracte();
$abc->abstractee();
$abc->abstractf();
```

Output:

infa
infb
infc
infd
abstracte
abstractee
abstractf

Date
17/02/2017

Conclusion: When an abstract class implements an interface then it has flexibility to define or not define the all functions of interface.

But when a normal class extends an abstract class then it's responsibility/duty of normal class to define all methods of abstract class as well as the all methods of interface extended by the abstract class.

Date
20/02/2017

preg_split()

Split string by a regular expression. Return an array on success & FALSE on failure. split() is deprecated & removed in PHP 7.0.

Example:

```
$date = "04/30/1973";
list($month, $day, $year) = preg_split('/[\/]/'
                                , $date);
echo "Month: $month; Day: $day; Year: $year";
```

Output:

Month: 04; Day: 30; Year: 1973

Example:

```
$keywords = preg_split("/[\s,]+/", "hypertext lan
print_r($keywords);                guage, programming");
```

(1 space)

Output:

```
Array
(
    [0] => hypertext
    [1] => language
    [2] => programming
)
```