PHP – Abstract Classes

The list of reserved words in PHP includes the "abstract" keyword. When a class is defined with the "abstract" keyword, it cannot be instantiated, i.e., you cannot declare a new object of such a class. An abstract class can be extended by another class.

```
abstract class myclass {
   // class body
}
```

As mentioned above, you **cannot declare an object of this class**. Hence, the following statement –

```
$obj = new myclass;
```

will result in an **error** message as shown below –

```
PHP Fatal error:  Uncaught Error: Cannot instantiate abstract class myclass
```

An abstract class may include properties, constants or methods. The class members may be of public, private or protected type. One or more methods in a class may also be defined as abstract.

If any method in a class is abstract, the class itself must be an abstract class. In other words, a normal class cannot have an abstract method defined in it.

This will raise an **error** –

```
class myclass {
   abstract function myabsmethod($arg1, $arg2);
   function mymethod() #this is a normal method {
      echo "Hello";
   }
}
```

The **error message** will be shown as –

```
PHP Fatal error:  Class myclass contains 1 abstract method
and must therefore be declared abstract
```

You can use an abstract class as a parent and extend it with a child class. However, the child class must provide concrete implementation of each of the abstract methods in the parent class, otherwise an error will be encountered.

Example

In the following code, **myclass** is an **abstract class** with **myabsmethod()** as an **abstract method**. Its derived class is **mynewclass**, but it doesn't have the implementation of the abstract method in its parent.

Open Compiler

```php
<?php
   abstract class myclass {
      abstract function myabsmethod($arg1, $arg2);
      function mymethod() {
         echo "Hello";
      }
   }
   class newclass extends myclass {
      function newmethod() {
         echo "World";
      }
   }
   $m1 = new newclass;
   $m1->mymethod();
?>
```

The **error message** in such a situation is –

```
PHP Fatal error:  Class newclass contains 1 abstract method and must
therefore be declared abstract or implement the remaining
methods (myclass::myabsmethod)
```

It indicates that newclass should either implement the abstract method or it should be declared as an abstract class.

Example

In the following PHP script, we have **marks** as an abstract class with percent() being an abstract method in it. Another **student** class extends the **marks** class and implements its percent() method.

Open Compiler

```php
<?php
   abstract class marks {
      protected int $m1, $m2, $m3;
      abstract public function percent(): float;
   }

   class student extends marks {
      public function __construct($x, $y, $z) {
         $this->m1 = $x;
         $this->m2 = $y;
         $this->m3 = $z;
      }
      public function percent(): float {
         return ($this->m1+$this->m2+$this->m3)*100/300;
      }
   }

   $s1 = new student(50, 60, 70);
   echo "Percentage of marks: ". $s1->percent() . PHP_EOL;
?>
```

It will produce the following **output** –

```
Percentage of marks: 60
```

Difference between Interface and Abstract Class in PHP

The concept of abstract class in PHP is very similar to interface. However, there are a couple of differences between an interface and an abstract class.

Abstract class	**Interface**

Use abstract keyword to define abstract class	Use interface keyword to define interface
Abstract class cannot be instantiated	Interface cannot be instantiated.
Abstract class may have normal and abstract methods	Interface must declare the methods with arguments and return types only and not with any body.
Abstract class is extended by child class which must implement all abstract methods	Interface must be implemented by another class, which must provide functionality of all methods in the interface.
Can have public, private or protected properties	Properties cannot be declared in interface