oops

we break our code into objects

class →

classes are used to define a new data type like int, float → and class is also a data type

class gives a facility to hide the data.

Rules

① class name can be any valid label

② It can't be php reserved word

③ A valid class name starts with letter or underscore

```
class Classname → start with first letter
                  capital
{
  var $variable-name;  ] → data member / properties
  var $variable-name;  ]
function method_name()  → w/o parameter
{ body of method; }
                          → parameter-list
function method-name (parameter)
{ body of method;
}.
                              methods /
                              member function
```

Attribute/ → camera, RAM,
Property.   screen

DATE: / 20
PAGE No

```
class Mobile {
  public
   var  $model // global variable
  function showModel ($number)
   {
     global $model;
       $model = $number;   → $this→model = $number
    echo "model number : $model";
                  ↓
   }          echo "model no": $this→model
}.
```

※. You can't assign computed value inside a class

Ex →

```
   public $ price = 10+20;
  public $ name = "Geeky"." shocus";
```

```
class Name                     simply write
{                                 ↓  "Geeksforgeeks";
 public $ name = "Geeks.for.geeks";
   function setName ($ name)
     {
      $this→name = $ name;
     }}
```

*-You can't begin the name of method with double underscore

Ex → function __setName()

Object →

new operator is used to create an object

Syntax → $object_name = new class_name

Creating object →

```
class Mobile
{
public $model; // properties / Class Members
                                  Data Member
function showModel($number)
{ $this->model = $number;
  echo "model no: $this->model";
}}
$samsung = new Mobile;
```

DATE : / 20
PAGE No

accessing class member using object

-> operator is used to access class member using object

```
object_name -> variable_name;
$samsung -> model;

object_name -> method-name()
$samsung -> showModel();

Object_name -> method_name(parameter_list)
$samsung -> showModel('A8');
```

Scanned by CamScanner

```
<?php
Class Mobile {
var $model; // properties / Class Member / data members
function show Model ($number) {
  global $model;       ]
  $model = $number;    ] → $this → model = $number
echo "Model number is : $model";
                           ↳ $this → model <br>
}}
         ↳ object name
$samsung = new Mobile ;
$samsung → show model('A8'); (Accessing m/m function using object name)
$ LG = new Mobile;
$ lg → show model ('G5);   ?>
```

O/P → Model number is 'A8';
model number is 'G5';

```
<?php
Class Mobile {
var $model;
function show model() {
echo "Model number is $ this→model <br>";   → keyword points to current object
}}
$samsung = new Mobile ;
$samsung → model = "A8+"
$samsung → new model ();
?>
```

O/P → ~~M~~Model no is A8

Scanned by CamScanner

DATE : / 20
PAGE No

$lg = new Mobile;

$lg → model = "4G5";

$lg → showModel();

$this keyword →

$this keyword points to current object You can use $this followed by the → aperator

Constructor →

↳ They are called directly when an object is created

↳ Contsructor should have the same name as the class name

↳ Constructor have a special name in PHP __construct

Scanned by CamScanner

declaration of constructor

```
Class Student          default          class Student
{                      const            {
 function __construct()                 function Student()
 {                                      {
 echo "Const called";                   echo "Const called";
 }                                      }}
}
```

```
<?php
Class Student {
function __construct() {
  echo "constructor called";
}}
 $stu = new Student;
?>
```

O/P → Construct called

default constructor → which has no parameters

```
Class Student
{ function __construct() // default constructor
{ echo "default const";
}}
$stu = new Student;
```

Parametrized const → which can take the arguments

```
Class Student
{
 public $roll;
function __Construct ($enroll)
{
 this → roll = $enroll;
}}
 $stu = new Student (10);
```

→ one parameter

```
Class Student
{
function __Construct ($a, $b, $c)
{
}
$stu = new Student ("Hello", "Everyone", "Bye");
```

→ more than one parameter

```
<?php
class Student {
 public $roll;
function __construct($enroll){
 echo "para Constructor";
  $this->roll = $enroll;
  echo $this->roll;
 }
}
 $stu = new Student (10);
?>
```