PHP

DATE: / / 20
PAGE No.

↳ Destructor → Dy destructor are called when u explicitly destroy an object or when all the references to the object go out of scope

↳ destructor have special name

PHP __ destruct

↳ destructor don't have any arguments

↳ Syntax → function __ destruct ()
{
echo "Destroyed"; }

Inheritance →

(1) The mechanism of deriving a new class from old one is called inheritance or derivation

Old class
↓
New class

Old class → Super class, Base class,
Parent class

Child class → new class, Sub class,
Extended class

Parent Class
↓
Child class

Father
- name
- money
- business

Son
- Job
- Access property of father

Class A
{
public $ id
public $ name
};
class B extends A
{
public $ salary
};

Declaration of Child Class

class child class name extends parent class name
{
members of child class
};

DATE : / 20
PAGE No

Type of Inheritance

1. Single Inheritance → Parent class → Child class
2. Multiple ×
3. Multi-level → Father → son → Grandson
4. Hierarchical → Father → son, Daughter, son

Syntax

```
class Parent
{
}
class Child extends Parent
{
}
```

Multi-level

```
class Father
{ }
class son extends Father
{ }
class Grandson extends son
{ }
```

Hierarchical

Class Father
{ };
Class Son extends Father
{ };
Class Daughter extends Father
{ };

~~Static~~ Abstract class ⇒ is declared with abstract keyword.
↳ It can have abstract or non abstract methods.
↳ Object of abstract class can't be created

Abstract method ⇒
A method that is declared as abstract & doesn't have implementation

abstract function disp(); // no body.

abstract class father → Object of abstract class can't be created

{

abstract function disp();

}

Class Son extends father

{

public function disp()

{ echo "Abstract defined";

}

}

```
<?php
class Father {
function disp() {
echo "Method";
}}
$obj = new Father;
$obj -> disp();
?>
```

If we use abstract class then shows error

∴ we can't create the obj of abstract class

```
<?php
abstract class Father {
function disp() {
echo "Method"; }
abstract function ab();
}
class Son extends Father {
public function ab() {
echo "Abstract"; }}
$obj = new Obj;
$obj -> ab();
?>
```

OP → Abstract

abstract method → mandatory
class to make class abstract

final →

final keyword is used to create final method or class

↳ final method can't be overridden in child class

↳ final class can't be inherited it means we can't create subclass of final class

```
class A {
final function disp() {
  echo "Super class";
}
}
class B extends A {
  function disp() {          → error → can't override
  echo "Child class";          final method.
}}
```

final class →

```
final class A {
  function disp();
}
class B extends A → error → ∴ we can't create (∴ class is final
```

→ class B may not inherit from final class A