[EXPERIMENT - 8]

[CLASS AND OBJECT]

— [CLASSES]

① CLASSES ARE A BLUEPRINT OR A TEMPLATE FOR CREATING OBJECTS

② CLASSES DEFINE PROPERTIES AND METHODS THAT OBJECTS OF CLASS WILL HAVE.

③ CLASSES ARE DEFINED VIA KEYWORD "class" FOLLOWED BY ITS CLASS NAME.

SYN

```
class class_name {
    // class body
}
```

Eg

```
class game {
    public $name = "Red Dead";
    public $dev = "Rockstar";
    public $year = 2018;

    public function show() {
        echo $this->name . $this->dev .
        $this->year;
    }
}
```

Sundaram®
FOR EDUCATIONAL USE

OBJECTS

① AN OBJECT IS AN INSTANCE OF A CLASS

② THEY CREATED VIA THE "new" KEYWORD FOLLOWED BY CORRESPONDING CLASS NAME.

③ OBJECTS HAVE PROPERTIES AND METHODS DEFINED IN THEIR CLASS

SYN `$obj_name = new class_name();`

Eg
```
class A {
    public function show() {
        echo "Hi";
    }
}

$obj = new A();
$obj -> show();   // Hi
```

CONSTRUCTOR AND DESTRUCTOR

CONSTRUCTOR

① IS A SPECIAL METHOD THAT IS AUTOMATICALLY EXECUTED WHEN AN OBJECT IS CREATED OF A CLASS

Sundaram
FOR EDUCATIONAL USE

(2) IT IS USED TO INITIALIZE PROPERTIES.

(3) IN [PHP], CONSTRUCTORS ARE NAMED "__construct()"

(4) CONSTRUCTORS HAVE NO RETURN TYPE.

(5) CONSTRUCTORS CAN ACCEPT PARAMETERS

SYN

```
class class_name {
    function __construct() {
        // CONSTRUCTOR BODY
    }
    // CLASS BODY
}
```

Eg

```
class A {
    function __construct() {
        echo "Object Created";
    }
}

$obj = new A(); // Object Created
```

Sundaram®
FOR EDUCATIONAL USE

DESTRUCTOR

(1) A SPECIAL METHOD THAT IS AUTO-MATICALLY CALLED WHEN AN OBJECT IS DESTROYED OR THE EXECUTION OF SCRIPT ENDS OR WHEN OBJECT GOES OUT OF SCOPE.

(2) USED TO PERFORM CLEANUP TASKS RELATED TO OBJECTS.

(3) THEY DONT ACCEPT PARAMETERS.

(4) IN [PHP] THEY ARE NAMED "__destruct()"

SYN

```
class   class_name {
    function __destruct() {
        // body
    }
    // class body
}
```

Eg.

```
class DB {
    private $connection;
    function __construct() {
```

Sundaram®
FOR EDUCATIONAL USE

```
$this->connection = mysqli_connect(
                    "localhost", "root",
                    "123", "eco_eco_databe");
    }

    function __destruct(){

        mysql_close($this->connection);
        echo "DB close";
    }
}

$con = new DB();
```

INHERITANCE

(1) WHEN A CLASS INHERITS PROPERTIES AND METHODS OF ANOTHER CLASS IS CALLED AS INHERITANCE.

(2) INHERITANCE IS COMMONLY USED FOR REUSABILITY.

(3) THE CHILD CLASS CAN ACCESS METHODS AND PROPERTIES ACCORDING TO THEIR RESPECTING ACCESS SPECIFIER.

(4) IT CAN ALSO HAVE ITS OWN METHODS AND PROPERTIES

Sundaram
FOR EDUCATIONAL USE

Eg.
```
class A {
    // body
}

class B extends A {
    // body
}
```

TYPES

— A CLASS INHERITS FROM ONLY ONE CLASS IS CALLED SINGLE INHERITANCE.

Eg.
```
class Parent {
    // body
}

class Child extends Parent {
    // body
}
```

— IF A CHILD CLASS INHERITS FROM ANOTHER CHILD CLASS, CREATING A HEIRARCHY ITS CALLED AS MULTILEVEL INHERITANCE

Sundaram
FOR EDUCATIONAL USE

Eg.

```
Class A {
    //body
}

class B extends A {
    // body
}

class C extends B {
    //body
}
```

- IF A CLASS INHERITS FROM MULTIPLE CLASSES IT'S CALLED AS MULTIPLE INHERITANCE HOWEVER, THIS IS NOT POSSIBLE IN (PHP) IT CAN BE ACHEIVED VIA "traits".

- TRAITS ARE DECLARED VIA "trait" KEYWORD

- TO USE A TRAIT IN A CHILD CLASS WE USE THE "use" KEYWORD FOLLOWED BY THE TRAIT NAME.

Eg.

SYN

```
trait trait_name {
    //body
}
```

Sundaram®
FOR EDUCATIONAL USE

Eg.

```
class A {
    // body
}

trait B {
    // body
}

class C : extends A {
    use B ;
    // body
}
```

[STATIC]

— PROPERTIES

- CLASS LEVEL VARIABLES THAT ARE SHARED AMONG ALL INSTANCES OF THE CLASS

- THEY ARE DECLARED VIA "static" KEYWORD

- THEY CAN ACCESSED WITHOUT CREATING INSTANCES OF THE CLASS

- WITHIN ANOTHER METHOD OF SAME, WE CAN ACCESS THEM VIA "self :: prop_name"

Sundaram®
FOR EDUCATIONAL USE

- IF WITHIN CHILD CLASS "parent :: prop_name"

Eg. class A {

public static $name = "GO";

}

echo A:: $name; // GO

— METHODS

- THESE METHODS CAN BE CALLED WITHOUT CREATING INSTANCES OF THE CLASS

- DECLARED VIA "static" KEYWORD

~~AN ABSTRACT CLASS MUST HAVE~~

- TO ACCESS WITHIN SAME CLASS DIFF. METHOD WE USE "self :: method"

- WITHIN CHILD CLASS WE USE "parent :: method"

Eg. class A { function

public static show(){ echo "Hi"; }

}

echo A:: show(); // Hi

Sundaram®
FOR EDUCATIONAL USE

METHOD OVERLOADING

- WE CAN PERFORM METHOD OVERLOADING WITH HELP OF MAGIC FUNCTION "__call()"

- IT ACCEPTS TWO ARGUMENTS - "func-name" AND AN "array".

Eg.

```
class shape {

    function __call ($func_name, $arg) {
        if ($func_name == "area") {

            switch ($arg[0]) {

                case "circle":
                    return 3.14 * $arg[1];

                case "rect":
                    return $arg[1] * $arg[2];

                case "tri":
                    return 0.5 * $arg[1] * $arg[2];
            }
        }
    }
```

Sundaram®
FOR EDUCATIONAL USE

```
$s = new Shape;
echo "Circle = " . $s->area("circle", 5);
echo "Rect = " . $s->area("rect", 7, 8);
echo "Tri=" . $s->area("tri", 12, 8);
```

METHOD OVERRIDING

- IF A FUNCTION FROM PARENT CLASS IS IMPLEMENTED IN CHILD CLASS WITH SAME NAME AND SIGNATURE AS OF PARENT CLASS, ITS CALLED AS METHOD OVERRIDDING.

Eg.

```
class A {
    public function show() {
        echo "Parent";
    }
}
class B {
    public function show() {
        echo "Child";
    }
}
```

Sundaram®
FOR EDUCATIONAL USE

```
$x = new B();
$x -> show();          // Child
```

FINAL

- FINAL KEYWORD IS USED BY EITHER OR METHODS.

- IF A CLASS DEFINED VIA FINAL KEYWORD THEN SUCH A CLASS CANT BE INHERITED.

- IF A METHOD IS DEFINED VIA FINAL KEYWORD THEN SUCH A METHOD CANT BE OVERRIDDEN

SYN

```
final class class_name {

    final public function func_name() {
        // body
    }
}

// body
}
```

Sundaram®
FOR EDUCATIONAL USE

Eg.

```
final class A {
    final public fonction show(){
        //echo "I cant be overridden"
        //echo " Cant ben inherited"
}

class B extends A { // throws error
    public fonction show() { // throws error
        //body
}
    // body
}
```

[OBJECT CLONING]

[SHALLOW COPY]

— HERE OBJECT "A" WILL COPY ALL FIELDS OF OBJECT "B" VALUES.

— IF ITS MEMORY ADDRESS THEN ITS MEMORY ADDRESS IS COPIED, IF VALUE IS PRIMITIVE TYPE THEN IT COPIES VALUE OF PRIMITIVE TYPE

Sundaram
FOR EDUCATIONAL USE

- IN SHALLOW COPY IF YOU MODIFY A'S FIELD YOU ALSO MODIFY WHAT B'S FIELD CONTAINS

[DEEP COPY]

- HERE THE DATA IS COPIED COMPLETELY.

- HERE THE INSTANCES DONT DEPEND ON EACH OTHER

- ALL VALUES IN A's MEMORY ARE COPIED TO B's MEMORY

[CLONING]

SHALLOW COPY → ASSIGNMENT (=)

DEEP COPY → KEYWORD "clone"

Eg.

```
class myClass {

    private string $name;
    private string $weapon;

    public function __construct($name, $weapon){

        $this->name = $name;
        $this->weapon = $weapon;
```

Sundaram
FOR EDUCATIONAL USE

```
    }

    private function show(){
        echo $this->name;
        echo $this->weapon;
    }
}

$a = new AC("Spectre", "Ripper");
$a->show();
    // SpectreRipper
$b = $a;                          //Shallow Clone
$b->name = "Seraph";
$a->show();                       // Seraph Ripper
$b->show();                       // Seraph Ripper
$c = clone $a;
$c->name = "Reaper";
$a->show();                       // Seraph Ripper
$e->show();                       // ReaperRipper
```

	PRODUCT	PROCESS	TOTAL	SIGN
	09	15	24.	

Sundaram
FOR EDUCATIONAL USE