Introduction

ဒီစာအုပ်လေးမှာ OOP အကြောင်းပြောမှာဆိုတော့ OOP ဆိုတာကိုအရင် မိတ်ဆက်ပေးပါမယ်။ သူ့ရဲ့ long form ကတော့ object oriented programming ဖြစ်ပါတယ်။ Wiki ကတော့ ဒီလိုပြောထားပါတယ်။

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects"

OOP ဆိုတာ objects ဆိုတဲ့ သဘောတရားတွေပေါ်မှာ အခြေခံထားတဲ့ programming paradigm တစ်ခုဖြစ်ပါတယ်ဆိုပြီးတော့ ပြောထားပါတယ်။ ဒီနေရာမှာ programming paradigm ဆိုတဲ့ အသုံးအနှုန်းကို ဆက်ပြီးတော့ ကျနော်ရှင်းပြပါမယ်။ နားလည်ရလွယ်တဲ့ sequential နဲ့ procedural programming အကြောင်းကို ဥပမာပေးပြီးပြောပါမယ်။

Sequential programming ဆိုတာက code လိုင်းတွေကို statement တစ်ခုခြင်းဆီ execution လုပ်သွားတာမျိုးကိုပြောတာ၊ ဥပမာ နမူနာပေးရမယ်ဆို

```
1 $a = 1;
2 $b = 2;
3 $plus = $a + $b;
4 echo $plus;
```

ဒါဆိုလို့ရှိရင် အပေါ်က code block ထဲမှ variables တွေပါတယ်၊ ပြီးတော့ arithmetic + operator ကိုသုံးပြီးတော့ ပေါင်းခြင်းလုပ်သွားတယ်၊ နောက်တစ်ကြောင်းမှာ echo ဆိုပြီးရလဒ်ကို print ထုတ်ပြတယ်။ အခုပြထားတာ sequential programming ရဲ့ နမူနာတစ်ခု၊ ဒီနေရာမှာပြောစရာရှိတာက ပေါင်းရမယ့် number operation နောက်တစ်ကြိမ်ထပ်လုပ်ရမယ်ဆိုလဲ ဒီလိုမျိုး code ထပ်ရေးရဦးမှာပဲ။ ဒီနေရာမှာ procedural programming ဆိုတာက အသုံးဝင်လာတယ်။

Procedural programming ကတော့ sequential ရဲ့အားနည်းချက်တွေက cover လုပ်ပေးသွားနိုင်တယ်။ sequential နဲ့ရေးလာမယ်ဆို နောက်ပိုင်းမှာ ကိုယ့် project code တွေကတစ်အားရှုပ်လာမယ်။ code line တွေမလိုအပ်ဘဲတစ်အားများလာလိမ့်မယ်။ အဲ့လို ပြဿနာတွေကို procedural နဲ့ဖြေရှင်းလို့ရမယ်၊ ဘယ်လိုလဲဆိုတော့ procedural မှာ function ဆိုတာပါလာတယ်၊ function ကို သုံးပြီးတော့ sequence code တွေကိုစုစုစည်းစည်းထားလိုက်မယ်ဆို နောက်တစ်ကြိမ် အလားတူ logic တွေလာပြီဆို functionလေးကို reuse လုပ်လိုက်ရုံပဲ၊ မလိုအပ်တဲ့ code တွေလျော့ချနိုင်သလို complexity လည်းနည်းသွားမယ်။ ဥပမာ အပေါ်က sequential ကို procedural အဖြစ်ပြောင်းမယ်ဆို

```php
function addNum($a,$b)
{
  $plus = $a+$b;
  echo $plus;
}

//addNum(1,2);
```

ခေါ်သုံးချင်ပြီဆို addNum(1,2) ဆိုပြီးခေါ်လိုက်ရုံပဲ၊ ဒါဆိုလို့ရှိရင် နောက် plus operation တွေရှိတိုင်းမှာ sequence မှာတုန်းကလို လိုင်းငကြောင်းရေးစရာမလိုတော့ဘဲ တစ်ကြောင်းတည်းနဲ့ ပြတ်သွားလိုက်မယ်။ နောက်တစ်ခုတွေ့မှာက function အနောက်မှာပါလာတဲ့ param လေးတွေပဲ၊ နောက်ပိုင်း calculation လုပ်တဲ့အချိန် မတူညီတဲ့ digit တွေအတွက် addNum လို့ခေါ်တဲ့အချိန်မှာတည်းက တစ်ခါတည်းထည့်ပေးလိုက်လို့ရအောင်လုပ်ပေးထားတယ်။ ဒါကြောင့်မို့လို့ procedural နဲ့ဆိုလို့ရှိရင် sequence မှာကြုံလာရမယ့် code ထပ်မယ့် ပြဿနာတွေကိုရှောင်လို့ရသွားလိုက်မယ်။

အပေါ်ကပြောခဲ့တဲ့ sequential နဲ့ procedural ဆိုတာ programming paradigm တွေပဲ၊ paradigm ကိုမြန်မာလိုဘာသာပြန်ရရင် ရေးထုံးပုံစံတွေပေါ့။ OOP ဆိုတာကလည်း object သဘောတရားကို အခြေခံထားတဲ့ paradigm တစ်ခုပဲ၊ သူမှာပါတဲ့ ရေးထုံးတွေကိုတော့ ကျနော်တို့ တစ်ဖြည်းဖြည်းခြင်းလေ့လာသွားကြတာပေါ့။ အဓိကကတော့ OOP မှာရှိတဲ့ ရေးထုံးပုံစံတွေကြောင့် code ရေးရတဲ့နေရာမှာ ပိုပြီးတော့ သပ်သပ်ရပ်ရပ်ဖြစ်သွားမယ်၊ ထိန်းသိမ်းရတာပိုလွယ်ပြီးတော့ ပြန်လည်အသုံးပြုရတဲ့နေရာမှာ ကောင်းမွန်သွားမယ့်အပြင် တစ်ခြားသော ကောင်းကျိုးတွေလည်း

အများကြီးရှိပါတယ်။ ဒါကြောင့်လည်း programming language အားလုံးတိုင်းလိုလိုမှာ OOP ဆိုတဲ့ ရေးထုံးကို support လုပ်လာခဲ့ကြတာဖြစ်ပါတယ်။

Class and Object

အပေါ်မှာ ကျနော်ပြောခဲ့သလိုပဲ OOP ဆိုတာ object သဘောတရားတွေနဲ့ ဖွဲ့စည်းထားတယ်ဆိုတဲ့အတွက် class နဲ့ object ဆိုတာ OOP ရဲ့အဓိက ပင်မထောက်တိုင်တွေလို့ သတ်မှတ်လို့ရပါတယ်။ Class ဆိုတာက object ကို ခြုံငုံထားတဲ့ template တစ်ခုလို့ဆိုလို့ရပါတယ်။ မြင်သာအောင်ရှင်းပြရရင် နိုင်ငံ ဆိုတဲ့ ဝေါဟာရက class ဖြစ်ပြီးတော့ မြန်မာ၊ ဂျပန်၊ အမေရိကန် စတဲ့ အရာတွေက object အဖြစ်သတ်မှတ်နိုင်ပါတယ်။ တစ်နည်းအားဖြင့် ဆိုရရင် class ဆိုတာက logicial entity ဖြစ်ပြီးတော့ object ကတော့ physical entity အဖြစ်တည်ရှိပါတယ်။ နိုင်ငံ ဆိုတဲ့ဝေါဟာရဟာ အပြင်မှာ လက်တွေ့မရှိပါဘူး၊ စကားလုံးဝေါဟာရအဖြစ်သာ logical entity အဖြစ်သတ်မှတ်ထားတာဖြစ်ပြီးတော့ မြန်မာ ဆိုတဲ့ အရာကတော့ လက်တွေ့မှာ တစ်ကယ်တည်ရှိတဲ့ physical entity ဖြစ်ပါတယ်။ နောက်တစ်ခု ဥပမာ ထပ်ပေးရရင် Animal ဆိုတဲ့ class မှာ dog, cat, bird အစရှိတဲ့ object တွေရှိသလိုမျိုးပေါ့။ အောက်က code example လေးကိုဆက်ကြည့်ရအောင်။

```php
<?php

class Country {
  // property
  public $name = 'Myanmar';

  // method
  public function getName() {
    return $this->name;
  }
}
?>
```

ကျနော် country ဆိုပြီး class လေးတစ်ခုဆောက်ထားပါတယ်။ အဲ့ဒီ class ထဲမှာ name ဆိုတဲ့ property လေးထည့်ထားတယ်၊ value ကို Myanmar ဆိုပြီးတစ်ခါတည်းပေးထားပါတယ်။ နောက်တစ်ခု getName ဆိုတဲ့ method လေးရေးထားတယ်၊ သူကတော့ class ထဲမှာရှိတဲ့ name property ကို return ပြန်ပေးပါမယ်။ ဒါကရိုးရိုးရှင်းရှင်း class လေးတစ်ခုဖြစ်ပါတယ်။ အဲ့ဒီ class ကနေ object ဆိုတာဘယ်လိုဖြစ်လာလဲဆက်ကြည့်ရအောင်။

```php
<?php

class Country {
  // property
  private $name = 'Myanmar';

  // method
  public function getName() {
    return $this->name;
  }
}

$myanmar = new Country();
var_dump($myanmar);
//output = object(Country)#1 (1) { ["name"]=> string(7) "Myanmar" }
var_dump($myanmar->getName());
//output = string(7) "Myanmar"

?>
```

Country class ရဲ့ အောက်မှာ Myanmar ဆိုတဲ့ variable လေးတစ်ခုဆောက်ထားပြီး new ဆိုတဲ့ keyword နဲ့ country class ကိုလှမ်းခေါ်ထားပါတယ်။ ဒီလိုမျိုးလုပ်တဲ့ process ကို instantiate လုပ်တယ်လို့ခေါ်ပါတယ်။ country class ကို instantiate လုပ်ပြီးလို့ ရလာတဲ့ myanmar ဆိုတဲ့ variable သည် object ဖြစ်ပါတယ်။ var_dump နဲ့ ထုတ်ကြည့်မယ်ဆို country object တစ်ခုရလာမှာဖြစ်ပါတယ်။ နောက်တစ်ဆင့်အနေနဲ့ class ထဲမှာရှိနေတဲ့ getName ဆိုတဲ့ method ကို myanmar object ကနေတစ်ဆင့်ခေါ် ကြည့်မယ်ဆို return value အနေနဲ့ myanmar ဆိုတဲ့ string output ပြန်ရလာမှာဖြစ်ပါတယ်။ ဒါကတော့ ရိုးရိုးရှင်းရှင်းရှင်း class and object ပဲဖြစ်ပါတယ်။

အပေါ်က code လေးထဲမှာ property ကို private လို့ပေးထားပြီး method ကို public function လို့ပေးထားတာမြင်ပါလိမ့်မယ်။ လက်ရှိအခြေအနေမှာ myanmar object ထဲကနေ function ကိုလှမ်းခေါ်သလို property ကို myanmar->name လို့လှမ်းခေါ်မယ်ဆိုရင် error တတ်ပါလိမ့်မယ်။ ဘာလို့လဲဆိုတာ နောက်တစ်ပိုင်းမှာရှင်းပြပေးပါမယ်။

Visibility & Access Modifiers

ကျနော်တို့ အပေါ်က code example တွေကနေတစ်ဆင့် တွေ့ခဲ့ရတာက private နဲ့ public ပဲတွေ့ခဲ့ရပါသေးတယ်။ ဒါလေးတွေကို access modifiers တွေလို့ခေါ် ပြီးတော့ public, protected & private ဆိုပြီးသုံးမျိုးရှိပါတယ်။

Public ကိုသုံးထားတယ်ဆို သူ့ရဲ့ property တွေ method တွေကို သူ့ကို create လုပ်ထားတဲ့ class ရဲ့ ပြင်ပကနေလည်း လှမ်းပြီးတော့ ခေါ်ယူအသုံးပြုလို့ရတယ်။

Private ကိုသုံးထားတယ်ဆိုရင်တော့ဖြင့် သူ့ကို create လုပ်ထားတဲ့ class အတွင်းမှာပဲ ခေါ်ယူအသုံးပြုနိုင်မယ်။ class ရဲ့ပြင်ပကနေ လှမ်းခေါ်သုံးလို့မရဘူး။

Public & private ကိုပဲအရင် ဥပမာ ကြည့်ရအောင်။

```php
<?php
class Person {
  public $name = 'Hlaing';
  private $age = '23';
}

$man = new Person();
echo $man->name; // output = Hlaing
echo $man->age; // can't access private property error
?>
```

Protected ကလည်း private နဲ့သဘောတရား အတူတူပဲ။ ဒါပေမယ့် protected ကတော့ မူလသူ့ကို create လုပ်ခဲ့တဲ့ class ကို ပြန် inheritance လုပ်ထား class ထဲမှာတော့ လှမ်းခေါ်သုံးခွင့်ရှိတယ်။ မလုပ်ထားရင်တော့ ခေါ်သုံးခွင့်မရှိဘူး။ ဒီနေရာမှာ ကျနော် inheritance ဆိုတာကိုပြောသွားတယ်။ လောလောဆယ်မှာ inheritance ဆိုတာ အမွေဆက်ခံယူထားတဲ့ class အဖြစ်ဘဲ ကျနော်တို့ သတ်မှတ်ထားရအောင်၊ ဥပမာ Parent Class ကို Child Class ကနေ လှမ်းပြီး inheritance လုပ်သလိုမျိုးပေါ့။

```php
<?php
class ParentClass {
  public $name = 'Hlaing';
}

// Inheritance (From Parent to child)
class ChildClass extends ParentClass {

}

$child = new ChildClass();
echo $child->name; //output Hlaing
?>
```

အပေါ်က ဥပမာ ကိုကြည့်မယ်ဆို childclass မှာ property name ကမရှိပေမယ့် ParentClass ဆီကနေတစ်ဆင့် အမွေဆက်ခံ(Inheritance) ထားတဲ့အတွက် Parent ထဲမှာရှိတဲ့ property ကိုပါ လှမ်းပြီးတော့ အသုံးပြုနိုင်နေတာပဲဖြစ်ပါတယ်။ နောက်ပိုင်းမှာ ဒီအတွက် အပိုင်းတစ်ပိုင်း သက်သက်လာမှာဆိုတော့ ဒီလောက်ပဲသိထားရင် အဆင်ပြေပါတယ်။ protected ကို inheritance နဲ့ ဘယ်လိုလှမ်းခေါ်လို့ရနိုင်မလဲဆိုတာ code example လေးနဲ့ တစ်ချက်ဆက်ကြည့်ရအောင်။

```php
<?php
class Person {
  public $name = 'Hlaing';
  protected $age = '23';

}

// Inheritance
class Hlaing extends Person {
  public function output() {
    //calling protected property `age` using `this` keyword
    echo $this->age;
  }
}

$man = new Hlaing();
$man->output(); // output = 23
$man->age; // protected property can't access outside of the class
?>
```

Hlaing ဆိုတဲ့ class ကနေပြီးတော့ Person class ဆီကနေ inheritance လှမ်းလုပ်လိုက်ပါတယ်။ Hlaing class ထဲမှာ public function တစ်ခုဆောက်ပြီးတော့ Person class က protected

property ဖြစ်တဲ့ age ကို `this` ဆိုတဲ့ keyword လေးသုံးပြီးတော့ echo ထုတ်လိုက်ပါတယ်။ inheritance လုပ်ထားတဲ့အတွက်ကြောင့်သာ ဒီလို လှမ်းခေါ်သုံးနိုင်တာဖြစ်ပါတယ်။ အောက်က client code မှာ Hlaing class ကို instantiate လုပ်ပြီးတော့ public function ဖြစ်တဲ့ output ကိုလှမ်းခေါ်လိုက်မယ်ဆို Person class မှာ protected ဖြစ်နေတဲ့ age value ကိုရလာမှာဖြစ်ပြီးတော့ person class က protected property age ကိုတိုက်ရိုက်လှမ်းခေါ်မယ်ဆိုရင်တော့ error တတ်မှာဖြစ်ပါတယ်။