Object Oriented Programming with PHP

Object Oriented Concept

- **Classes**, which are the "blueprints" for an object and are the actual code that defines the properties and methods.
- **Objects**, which are running instances of a class and contain all the internal data and state information needed for your application to function.
- **Encapsulation**, which is the capability of an object to protect access to its internal data
- **Inheritance**, which is the ability to define a class of one kind as being a sub-type of a different kind of class (much the same way a square is a kind of rectangle).

Creating Class

- Let's start with a simple example. Save the following in a file called class.lat.php:

```
<?php
    class Demo
    {
    }
?>
```

Adding Method

- The Demo class isn't particularly useful if it isn't able to do anything, so let's look at how you can create a method.

```
<?php
        class Demo
        {
                function  SayHello($name)
                {
                    echo "Hello $name !";
                }
        }
?>
```

Adding Properties

- Adding a property to your class is as easy as adding a method.

```
<?php
        class Demo
        {
                public $name;

                function SayHello()
                {
                        echo "Hello $this->$name !";
                }

        }
?>
```

Object Instantiation

- You can instantiate an object of type Demo like this:

```
<?php
        require_once('class.lat.php');
        $objDemo = new Demo();
        $objDemo->name = "Bayu";
        $objDemo->SayHallo();
?>
```

Protecting Access to Member Variables (1)

- There are three different levels of visibility that a member variable or method can have :
  - Public
    - members are accessible to any and all code
  - Private
    - members are only accessible to the class itself
  - Protected
    - members are available to the class itself, and to classes that inherit from it

Public is the default visibility level for any member variables or functions that do not explicitly set one, but it is good practice to always explicitly state the visibility of all the members of the class.

Protecting Access to Member Variables (2)

- Try to change access level of property named "name" to private of previous code.
- What the possible solution of this problem?

- Make the getter and setter function…

Always use get and set functions for your properties. Changes to business logic and data validation requirements in the future will be much easier to implement.

Class Constants

- It is possible to define constant values on a per-class basis remaining the same and unchangeable.
- Constants differ from normal variables in that you don't use the $ symbol to declare or use them
- The value must be a constant expression, not (for example) a variable, a property, a result of a mathematical operation, or a function call

Class Constants (cont.)

```
<?php
   class MyClass
   {
      const constant = 'constant value';

      function showConstant() {
         echo  self::constant . "\n";
      }
   }

   echo MyClass::constant . "\n";
?>
```

Static Keyword

- Declaring class properties or methods as static makes them accessible without needing an instantiation of the class.
- A property declared as static can not be accessible with an instantiated class object

```php
<?php
class Foo
{
    public static $my_static = 'foo';

    public function staticValue() {
        return self::$my_static;
    }
}

class Bar extends Foo
{
    public function fooStatic() {
        return parent::$my_static;
    }
}

print Foo::$my_static . "\n";

$foo = new Foo();
print $foo->staticValue() . "\n";
print $foo-
>my_static . "\n";     // Undefined "Pro
perty" my_static

print $foo::$my_static . "\n";
$classname = 'Foo';
print $classname::$my_static . "\n"; //
 As of PHP 5.3.0

print Bar::$my_static . "\n";
$bar = new Bar();
print $bar->fooStatic() . "\n";
?>
```

Contructor

- Constructor is the method that will be implemented when object has been initiated
- Commonly, constructor is used to initialize the object
- Use function __construct to create constructor in PHP

```
<?php
        class Demo
        {
                function __construct
                {
                }
        }
?>
```

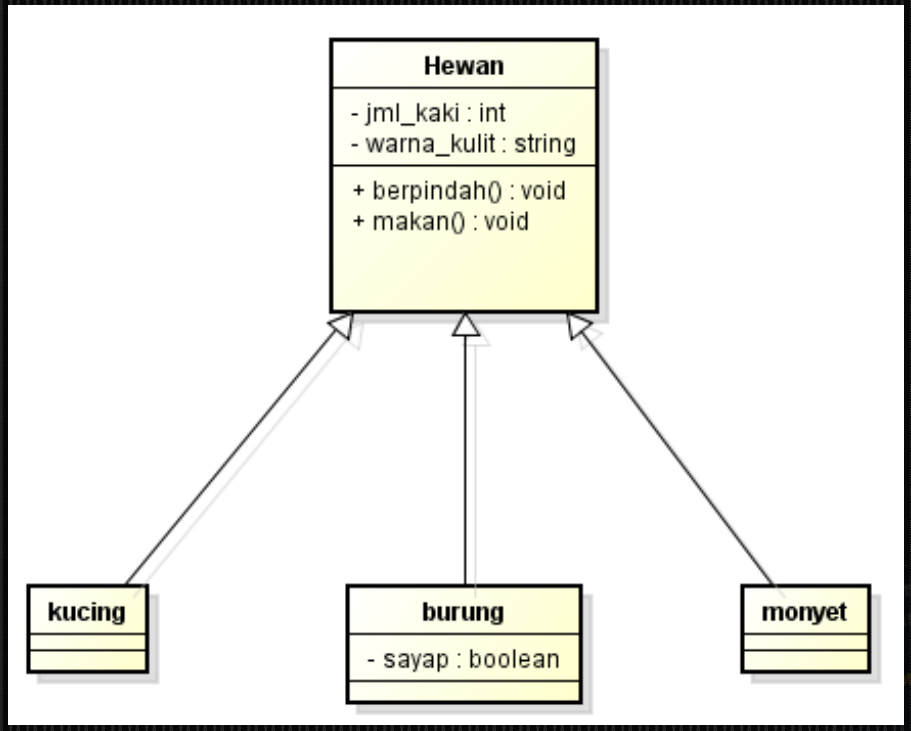
Destructor

- Destructor, is method that will be run when object is ended

```
<?php
        class Demo
        {
                function __destruct
                {
                }
        }
?>
```

Inheritance

- There are many benefits of inheritance with PHP, the most common is simplifying and reducing instances of redundant code.



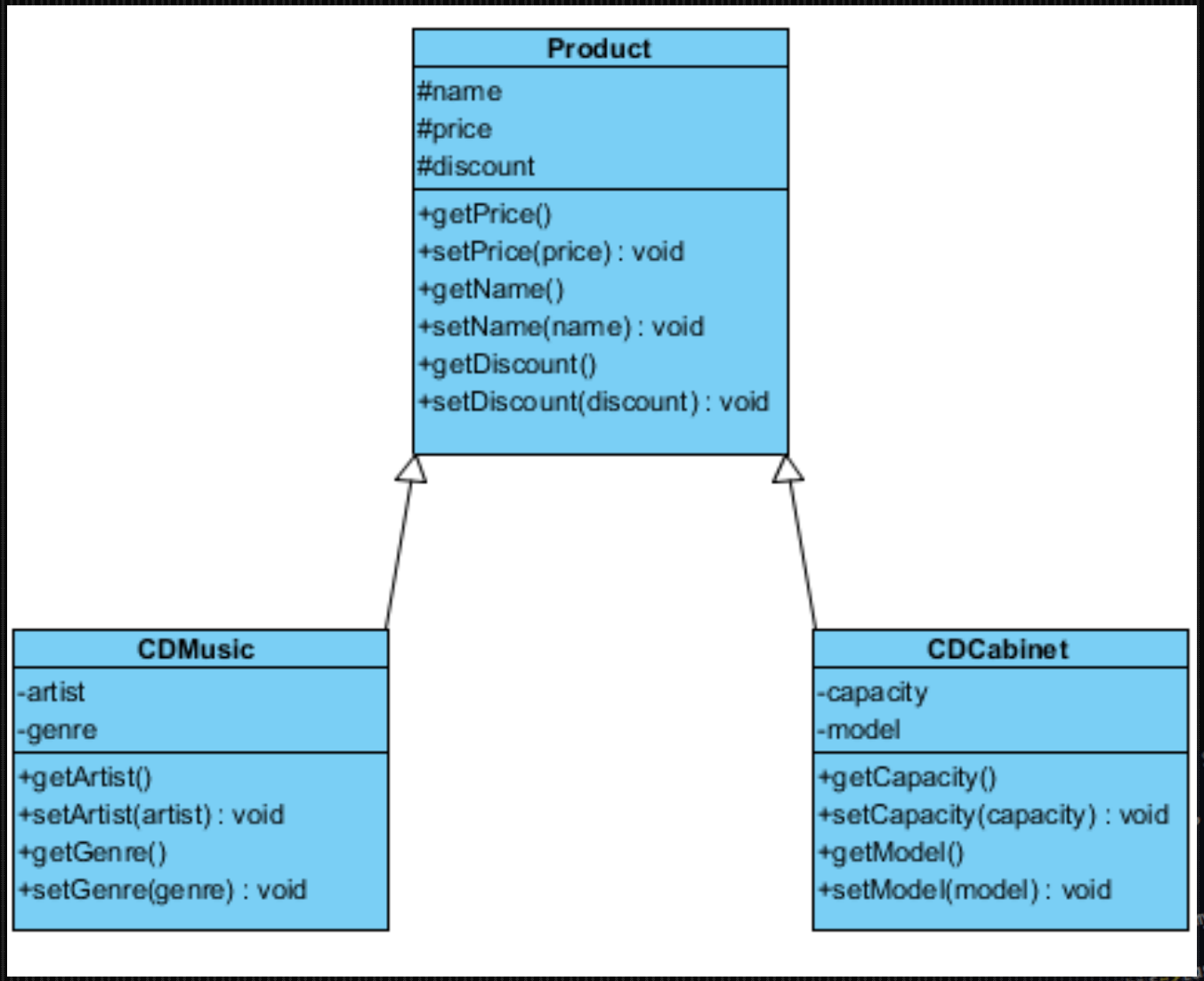
Inheritance (2)

```
class hewan
    {
        protected $jml_kaki;
        protected $warna_kulit;

        function __construct()
        {
        }

        function berpindah()
        {
                echo "Saya berpindah";
        }

        function makan()
        {
                echo "Saya makan";
        }
    }
```

Inherintace (3)

```
class kucing extends hewan
    {
        function berpindah()
        {
            echo "Saya merangkak dengan 4 kaki";
        }
    }

class burung extends hewan
    {
        protected $sayap;

        function berpindah()
        {
            echo "Saya terbang";
        }

        function makan()
        {
            echo "Saya makan dengan mematuk";
        }

    }

class monyet extends hewan
    {

    }
```

Tugas


Product
#name
#price
#discount
+getPrice()
+setPrice(price) : void
+getName()
+setName(name) : void
+getDiscount()
+setDiscount(discount) : void
CDMusic
-artist
-genre
+getArtist()
+setArtist(artist) : void
+getGenre()
+setGenre(genre) : void
CDCabinet
-capacity
-model
+getCapacity()
+setCapacity(capacity) : void
+getModel()
+setModel(model) : void

Tugas (cont.)

- Class product :
  - name
  - price
  - discount
- Class CDMusic :
  - artist
  - Genre
- Class CDRack
  - capacity
  - model

Tugas (cont.)

- CDMusic
  - Menuruni name, price dan discount dari Product
  - Price = price + 10%
  - Ada penambahan 5% pada discount
- CDRack
  - Menuruni name, price dan discount dari Product
  - Price = price + 15%
  - Tidak ada penambahan discount
- Buatlah code dalam PHP, serta simulasi untuk kasus tersebut!


```
<?php
        echo “Terima Kasih….!!!”
?>
```