

Abstract Class dan Abstract Method

1. Apa itu *Abstract Class*

Abstract class adalah sebuah *class* dalam OOP yang tidak dapat diinstansiasi atau dibuat *object*-nya. *Abstract class* biasanya berisi fitur-fitur dari sebuah *class* yang belum implementasikan. Seperti pada pembahasan sebelumnya tentang *class* *Connection* dimana kita harus membuat implementasi dari *class* tersebut dengan meng-*extends*-nya menjadi *MySQLConnection* dan

PostgreSQLConnection. Karena *abstract class* harus diimplementasikan melalui proses pewarisan, maka dalam *abstract class* berlaku aturan-aturan yang ada pada konsep pewarisan yang telah kita bahas sebelumnya.

Didalam sebuah *abstract class* kita dapat membuat *property* dan *method* yang nantinya dapat digunakan oleh *child class*. Tentu saja *property* dan *method* yang dapat digunakan oleh *child class* adalah *property* dan *method* yang memiliki visibilitas *protected* dan *public*.

Apakah kita dapat mendefinisikan *property* dan *method* dengan visibilitas *private* pada *abstract class*? Tentu saja bisa, dan tidak ada yang melarangnya. Biasanya *property* atau *method* pada *abstract class* dibuat *private* karena kita tidak ingin *child class* dapat mengakses *property* atau *method* tersebut atau kita membuat cara lain untuk mengakses *private property*.

2. Apa itu *Abstract Method*

Sama seperti *abstract class*, *abstract method* adalah sebuah *method* yang harus diimplementasikan oleh *child class*. *Abstract method* hanya ada pada *abstract class* dan *interface* (akan dibahas secara terpisah).

Bila biasanya setiap *method* yang kita buat pasti mempunyai kurawal {}, pada *abstract method* hal tersebut tidak dapat ditemui karena *abstract method* adalah sebuah *method* yang tidak memiliki *body* atau badan *method*.

Pada *child class*, *abstract method* harus didefinisikan ulang dan kita tidak dapat menggunakan *keyword* *parent* untuk memanggil *abstract method* pada *parent class*. Bila kita melakukan hal tersebut maka akan terjadi *error*.

3. Kegunaan *Abstract Class* dan *Abstract Method*

Secara mudah abstract class dan abstract method berguna untuk memastikan child class memiliki fitur-fitur yang telah ditentukan sebelumnya. Abstract class akan sangat berguna pada saat kita membahas tentang type hinting atau parameter hinting.

Dengan abstract class dan abstract method kita bisa lebih percaya diri ketika memanggil buah method karena dapat dipastikan method tersebut dimiliki child class.

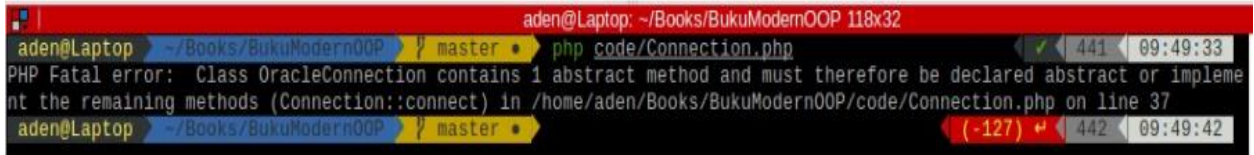
4. Contoh Penggunaan *Abstract Class* dan *Abstract Method*

```
<?php
abstract class Connection
{
    abstract public function connect($database, $username, $password, $host =
'localhost');
}
class MySQLConnection extends Connection
{
    public function connect($database, $username, $password, $host = 'l
ocalhost')
    {
        /**
         * Ceritanya ini logic koneksi ke database MySQL
         *
         * Anda tidak bisa menggunakan _keyword_ `parent` un
         tuk memanggil _method_ `connect()` milik _parent class_
         * karena akan mengakibatkan error.
         */
    }
}
class PostgreSQLConnection extends Connection
{
    public function connect($database, $username, $password, $host =
'localhost')
    {
        /**
         * Ceritanya ini logic koneksi ke database PostgreSQL
         *
         * Anda tidak bisa menggunakan _keyword_ `parent`
         untuk memanggil _method_ `connect()` milik _parent class_
         * karena akan mengakibatkan error.
         */
    }
}
```

Dengan cara demikian, maka *class* MySQLConnection dan PostgreSQLConnection dipaksa untuk mendefinisikan ulang *method* connect() sama seperti tujuan utama kita membuat *class* Connection pada bab sebelumnya. Bila kita tidak mengimplementasikan *method* connect() maka akan terjadi *error*. Perhatikan contoh berikut:

```
<?php
abstract class Connection
{
    abstract public function connect($database, $username,$password,$host=
'localhost');
}
class MySQLConnection extends Connection
{
    public function connect($database, $username, $password,$host = 'l
ocalhost')
    {
        /**
         * Ceritanya ini logic koneksi ke database MySQL
         *
         * Anda tidak bisa menggunakan _keyword_ `parent` un
tuk memanggil _method_ `connect()` milik _parent class_
         * karena akan mengakibatkan error.
        **/
    }
}
class PostgreSQLConnection extends Connection
{
    public function connect($database, $username, $password, $host =
'localhost')
    {
        /**
         * Ceritanya ini logic koneksi ke database PostgreSQL
         *
         * Anda tidak bisa menggunakan _keyword_ `parent`
untuk memanggil _method_ `connect()` milik _parent class_
         * karena akan mengakibatkan error.
        **/
    }
}
class OracleConnection extends Connection
{
}
```

Pada contoh diatas, kita membuat sebuah *child class* Oracle Connection yang *extends* dari *class* Connection tanpa mengimplementasikan *method* connect(). Bila program diatas dijalankan maka akan terjadi *error* PHP Fatal error: Class OracleConnection contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (Connection::connect) seperti gambar dibawah ini.



Perlu Anda ketahui bahwa didalam *abstract class* kita juga dapat membuat *non abstract method* sebagai berikut:

```
<?php
abstract class AbstractTaxCalculator
{
    private $previous;
    abstract public function maxPtkp();
    abstract public function minPtkp();
    abstract public function taxPercentage();
    public function getPrevious()
    {
        return $this->previous;
    }
    public function setPrevious($taxCalculator)
    {
        $this->previous = $taxCalculator;
    }
    public function calculate($ptkp)
    {
        $previousValue = 0;
        if ($previous = $this->getPrevious()) {
            $previousValue = $this->getPrevious()->calculate($previous-
>maxPtkp());
            $ptkp -= $previous->maxPtkp();
        }
        return ($this->taxPercentage() *$ptkp) +$previousValue;
    }
    public function isSupportPtkp($ptkp)
    {
        if ($ptkp < $this->maxPtkp() && $ptkp >= $this->minPtkp()) {
            return true;
        }
    }
}
```

```
}  
    return false;  
}  
}
```

Bagaimana apakah sudah lebih bisa memahami apa itu *abstract class* dan *abstract method*? Contoh diatas hanyalah sebuah ilustrasi dari sebuah *abstract class* yang memiliki *abstract method* dan *non abstract method*. Pada pembahasan-pembahasan selanjutnya, kita akan semakin memahami kegunaan *abstract class* dan *abstract method* secara lebih mendalam dengan melihat langsung penggunaan *abstract class* dan *abstract method* dalam *real project*.