CWX

CODEWORKS 2010

OOP THE PHP 5.3 WAY

SEATTLE • PORTLAND • AUSTIN • BALTIMORE • ORLANDO

@mtabini
Marco Tabini

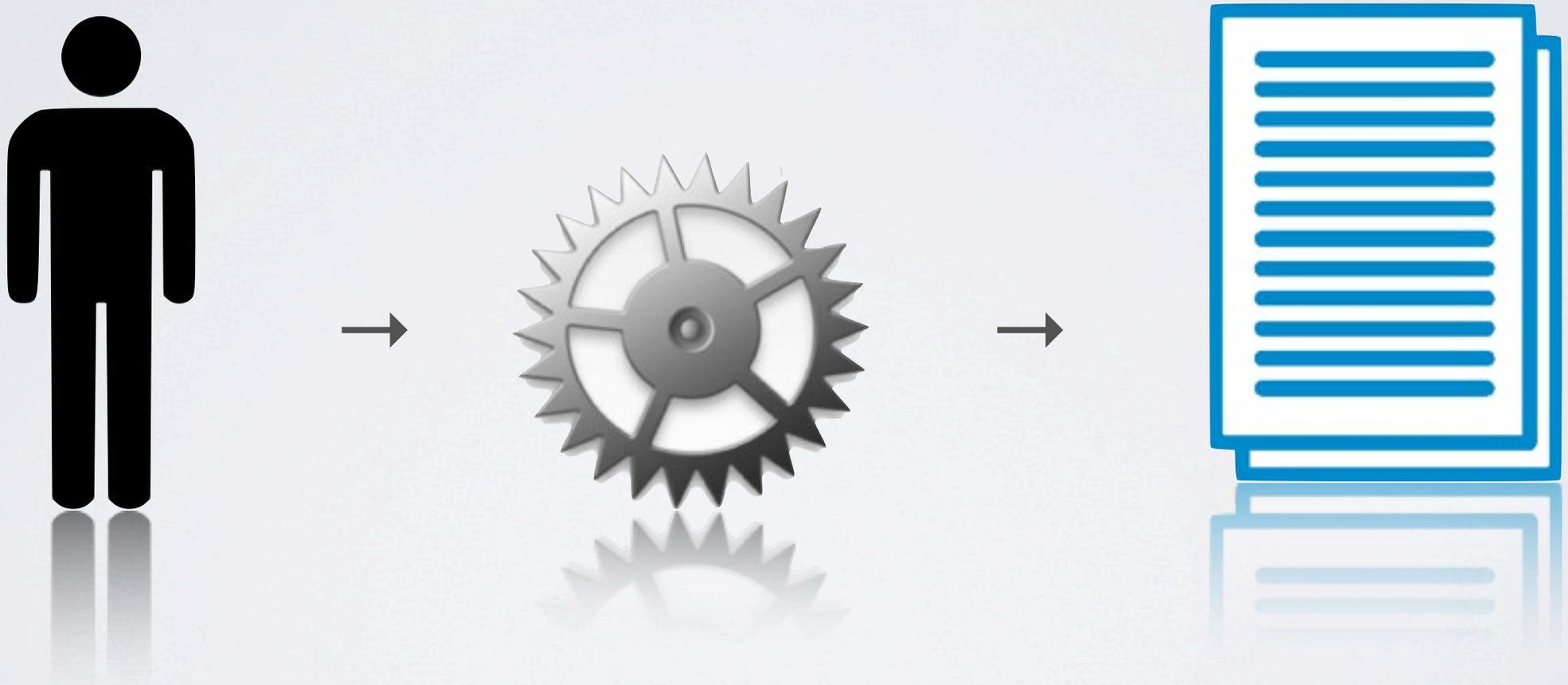
TWELVE compilation errors in one line of code. That *has* to be a new record.

1 Oct via web ☆ Favorite ↩ Reply 🗑 Delete

CWX
CODEWORKS 2010

OOP THE PHP 5.3 WAY

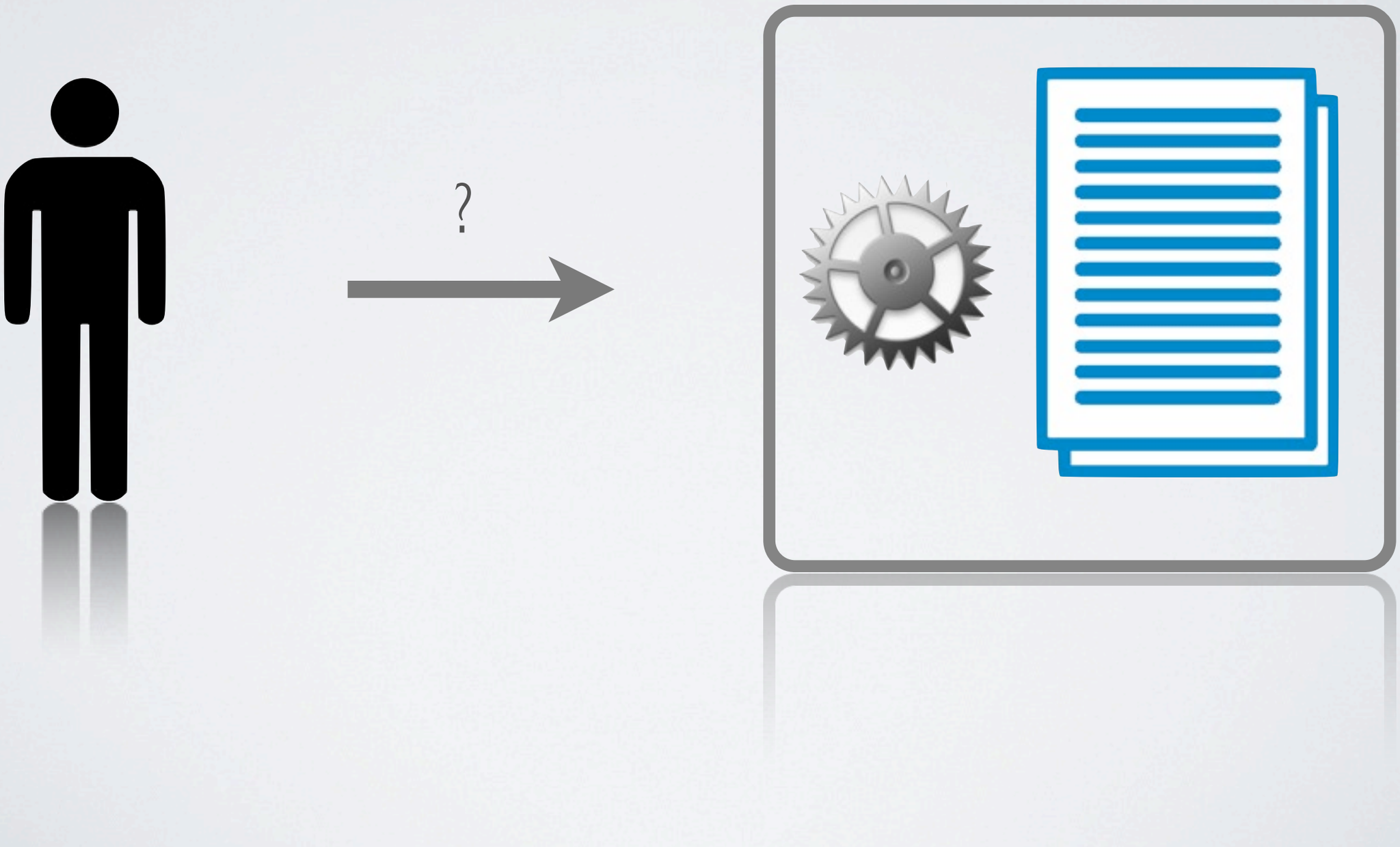
Marco Tabini



CWX
CODEWORKS 2010

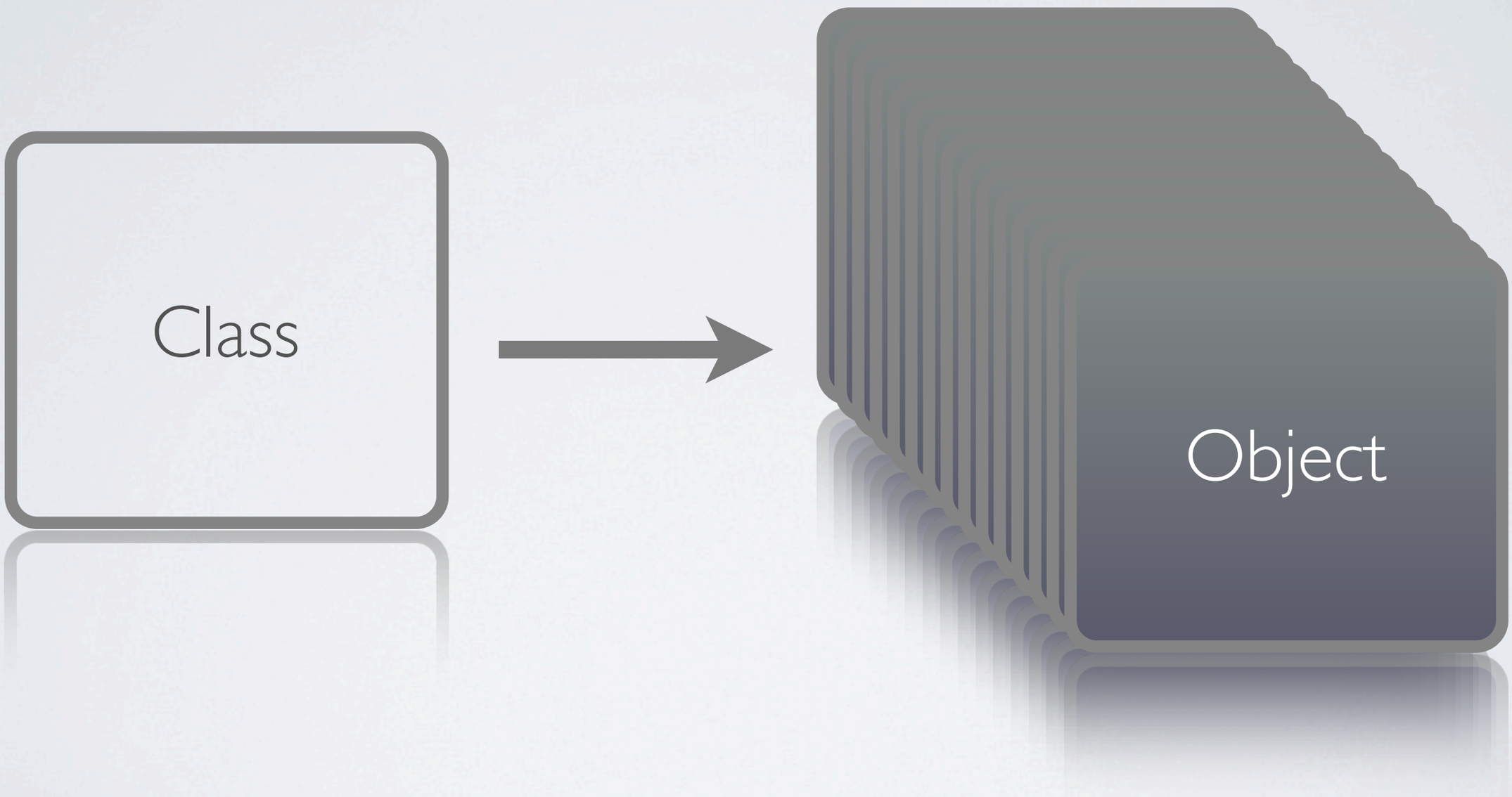
OOP THE PHP 5.3 WAY

Marco Tabini



OOP THE PHP 5.3 WAY

Marco Tabini



$x = new Class()

Class

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

Class

```
class a {

}
```

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

Class

```
class a {
    const C = 10;

    var $aVariable;

    function __construct() {
    }

    function __destruct() {
    }

    function doSomething() {
    }
}
```

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

Class

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

```
class a {
    const C = 10;

    protected $aVar;
    public $bVar;
    private $cVar;

    protected function __construct() {
    }

    private function __destruct() {
    }

    public function doSomething() {
    }
}
```

Class

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

```
class a {
    const C = 10;

    protected $aVar;
    public $bVar;
    private $cVar;

    protected function __construct() {
    }

    private function __destruct() {
    }

    public function doSomething() {
    }
}

class b extends a {

    public function doSomething() {
        parent::doSomething();
    }

}
```

Class

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

```
class a {
    const C = 10;

    protected $aVar;
    public $bVar;
    private $cVar;

    protected function __construct() {
    }

    private function __destruct() {
    }

    public function doSomething() {
    }
}

class b extends a {

    public function doSomething() {
        parent::doSomething();
    }

}
```

Class

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

```php
interface IMyClass {
    function a($a, $b);
}

abstract class a {

    abstract function something($data);

    function somethingElse($dataToo);

}

class b extends a {

    function something($data) {
        // Actual implementation
    }

}
```

Class

Encapsulation
Visibility
Inheritance
Polymorphism
Abstraction

```
class a {

    protected $_mySecretVariable;

    protected _mySecretFunction($data) {
        echo $data;
    }

    function __get($varName) {
        return $varName == "secret" ? $this->_mySecretVariable;
    }

    function __set($varName, $varValue) {
        if ($varName == "secret") {
            $this->mySecretValue = (int) $varValue;
        }
    }

    function __call($functionName, $arguments) {
        if ($functionName == "aFunctionThatDoesntExist") {
            call_user_func_array(array($this, $argumnets))
        }
    }

}
```

Static Classes

```php
<?php
class A {
    public static function who() {
        echo __CLASS__;
    }
    public static function test() {
        self::who();
    }
}

class B extends A {
    public static function who() {
        echo __CLASS__;
    }
}

B::test();
?>
```

```php
<?php
class A {
    public static function who() {
        echo __CLASS__;
    }
    public static function test() {
        static::who();
    }
}

class B extends A {
    public static function who() {
        echo __CLASS__;
    }
}

B::test();
?>
```

Finding Classes

```php
<?php
function __autoload($class_name) {
    include $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
?>
```

Error Reporting

```php
class AKindOfException extends Exception {}
class AnotherKindOfException extends Exception {}

try {

    try {

        if (rand(0, 1) < 0.5) {
            throw new AKindOfException();
        } else {
            throw new AnotherKindOfException();
        }

    } catch (AnotherKindOfException $e) {

        // Handle another kind of exception

    }

} catch (AKindOfException $e) {

    // Catch a kind of exception

}
```

Namespaces

example.php

```php
<?php

namespace combluepara bolamarco;

class Example {
    // Class implementation goes here
}
```

use.php

```php
<?php

require "example.php";

use combluepara bolamarcoExample as MarcoExample

class Example {
    // Define another Example here
}

$a = new combluepara bolamarcoExample;
$b = new MarcoExample;
$c = new Example;
```

Standard PHP Library

Standard PHP Library

Data Structures
Iterators
Interfaces
Exceptions

Standard PHP Library

Data Structures

- Linked List
- Stack
- Heap
- Priority Queue

Iterators

Interfaces

Exceptions

Standard PHP Library

Data Structures
Iterators
Interfaces
Exceptions

- Iterators
- Directory Access
- Array Access
- SimpleXML Iterator

Standard PHP Library

Data Structures
Iterators
Interfaces
Exceptions

- Countable
- Iterator
- Array Access
- Recursive Iterator

Standard PHP Library

Data Structures
Iterators
Interfaces
Exceptions

- Function call
- Range
- Logic
- Domain
- Under/Overflow
- Unexpected value

Thank you

marcot@tabini.ca
@mtabini