

# PHPUnitの MockObjectの紹介

Yoshio HANAWA(塙 与志夫)  
a.k.a. id:hnw

第4回Symfony2勉強会発表資料

# アジェンダ

- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ

- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ



# 自己紹介

- id:hnw
- round()の人
- 大好物：PHP本体のバグ

The screenshot shows a Hatena diary page. At the top, there is a navigation bar with 'Hatena::Diary' on the left, a search box, and buttons for '日記' and '検索'. On the right side of the navigation bar are links for 'ブログトップ', '記事一覧', '記事を書く', '管理', 'ログアウト', and 'ヘルプ'. Below the navigation bar is a dark red header area with the title 'hnwの日記' and icons for 'AI' and 'RSS'. Underneath the header is a light yellow area containing navigation links '<前日 | 次の日>' and '[プロフィール]'. The main content area features a dotted line separator above the entry text: '2011年4月7日 (木) PHPの新しいround関数にバグをみつけた'. To the right of the text are several icons: '編集', 'tB', '54 users', a comment icon, a share icon, and a star rating of seven stars.

# 自己紹介

- id:hnw
  - round()の人
    - 大好物：PHP本体のバグ
  - 最近Symfony2案件がありました
    - DIとテスト、楽しい！

- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ



モツクが  
何だか  
わかる方？

# 今日の目標

- モックが何だか、だいたいわかる
- Mockito試そう、って気になる



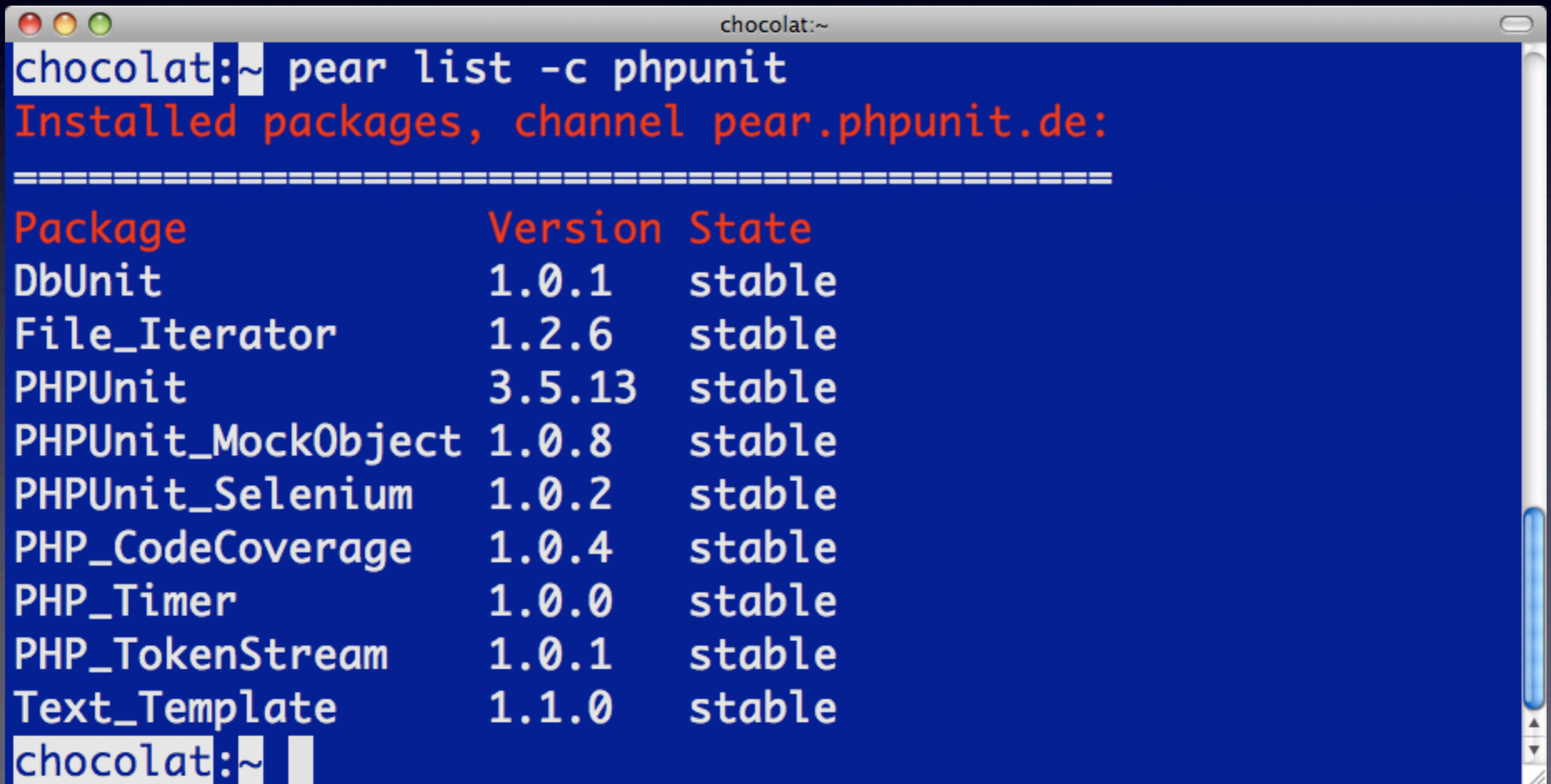
- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ

# PHPUnitの概要(1)

- テスティングフレームワーク
  - <http://www.phpunit.de/>
- Symfony2で採用
  - limeは捨てました！
- マニュアルの日本語版も充実

# PHPUnitの概要(2)

- 機能充実！

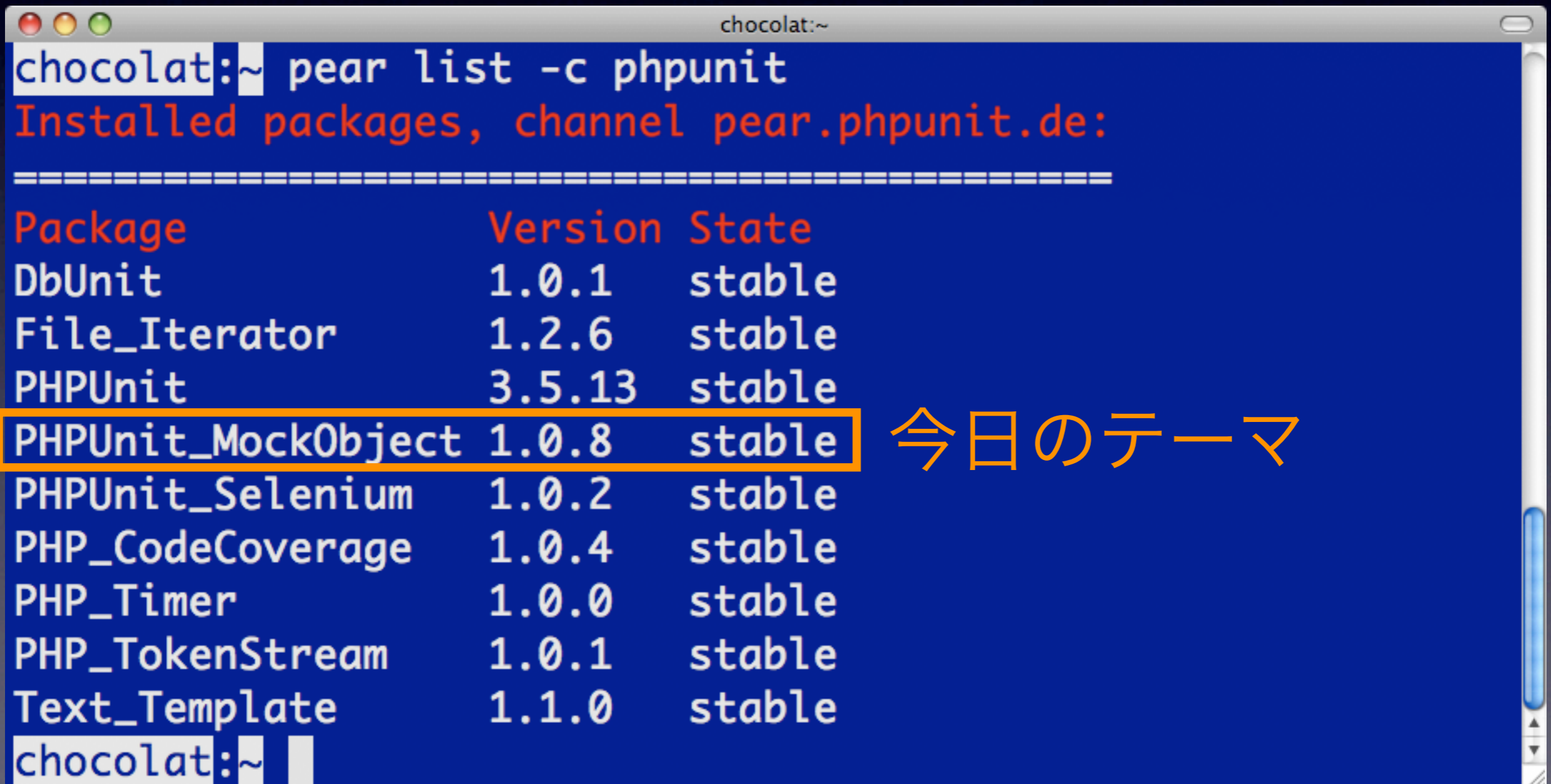


```
chocolat:~ pear list -c phpunit
Installed packages, channel pear.phpunit.de:
=====
Package           Version  State
DbUnit            1.0.1    stable
File_Iterator     1.2.6    stable
PHPUnit           3.5.13   stable
PHPUnit_MockObject 1.0.8    stable
PHPUnit_Selenium  1.0.2    stable
PHP_CodeCoverage  1.0.4    stable
PHP_Timer         1.0.0    stable
PHP_TokenStream   1.0.1    stable
Text_Template     1.1.0    stable
chocolat:~
```



# PHPUnitの概要(2)

- 機能充実！

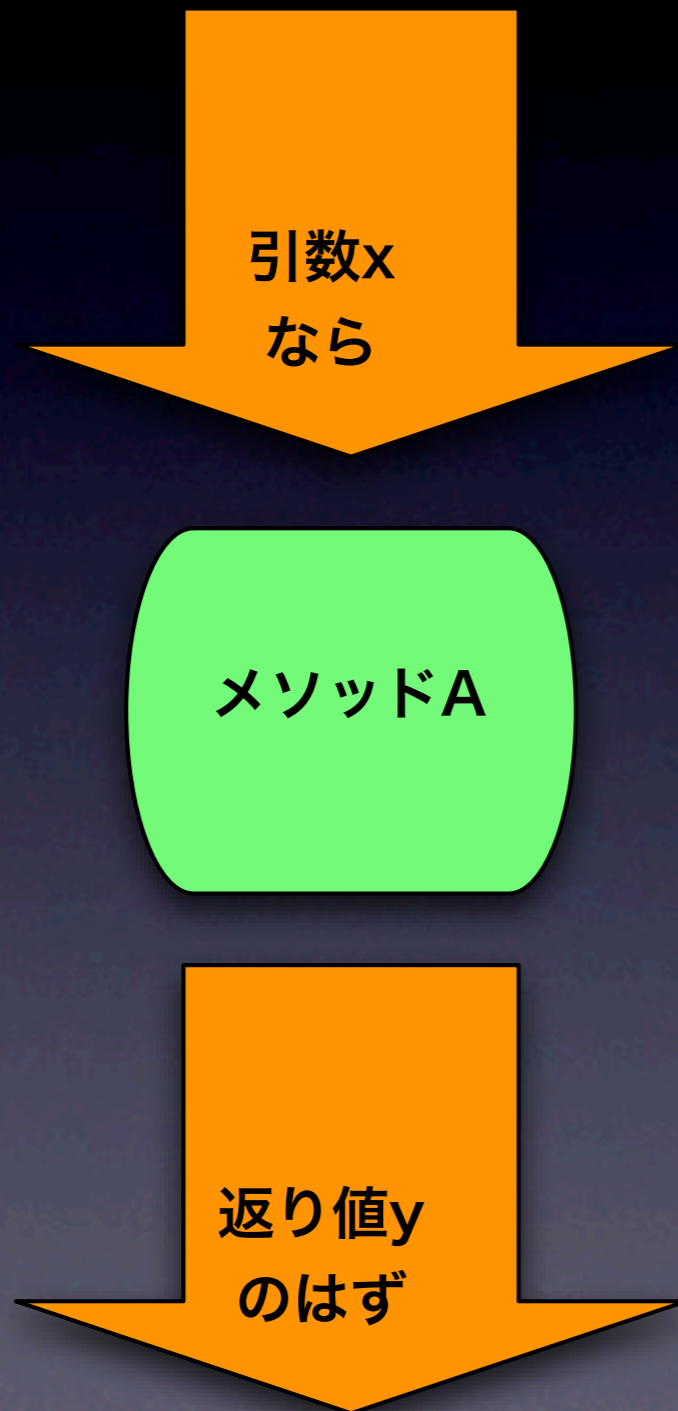


```
chocolat:~ pear list -c phpunit
Installed packages, channel pear.phpunit.de:
=====
Package          Version State
DbUnit           1.0.1  stable
File_Iterator    1.2.6  stable
PHPUnit          3.5.13 stable
PHPUnit_MockObject 1.0.8  stable
PHPUnit_Selenium 1.0.2  stable
PHP_CodeCoverage 1.0.4  stable
PHP_Timer        1.0.0  stable
PHP_TokenStream  1.0.1  stable
Text_Template    1.1.0  stable
chocolat:~
```

今日のテーマ

- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ

# 通常のテスト

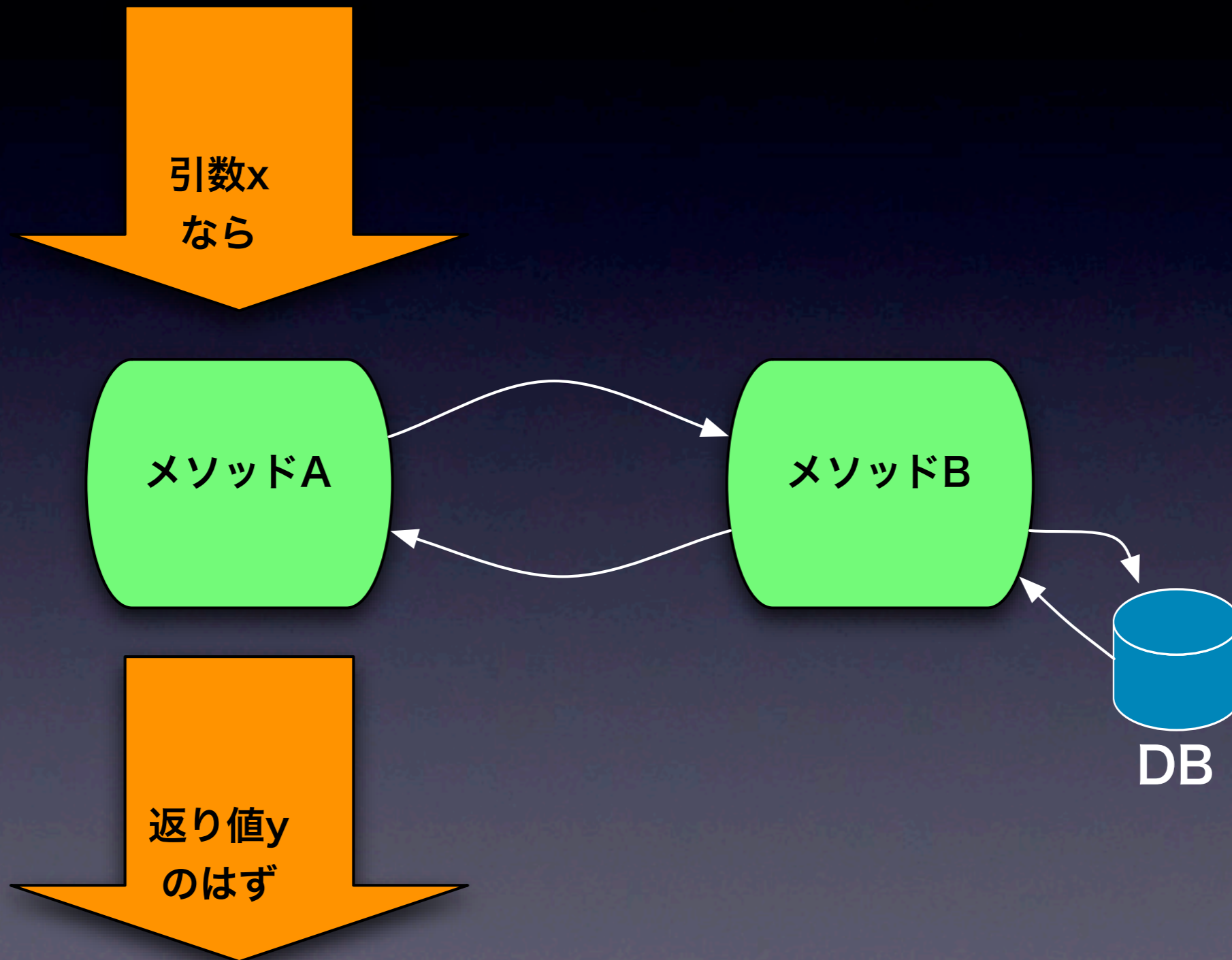




# どうテストする？(1)

- 外部システムに依存するテスト
  - 課金とか
  - DBアクセスとか

# どうテストする？(1)

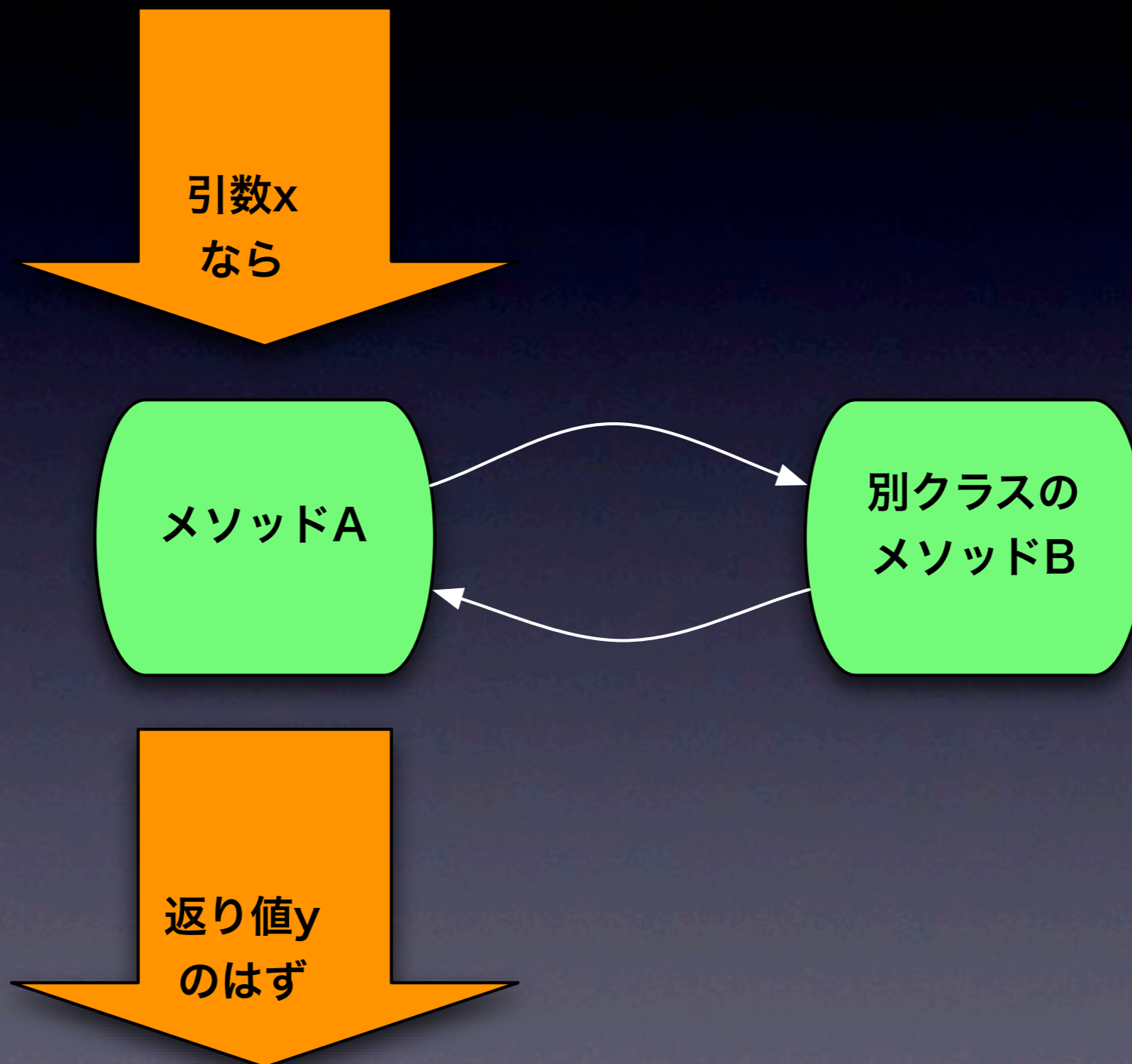


# どうテストする？(2)

- 他に依存しているクラスの単体テスト
  - 他のクラスのメソッドを呼び出している場合、誰のバグかわからない



# どうテストする？(2)

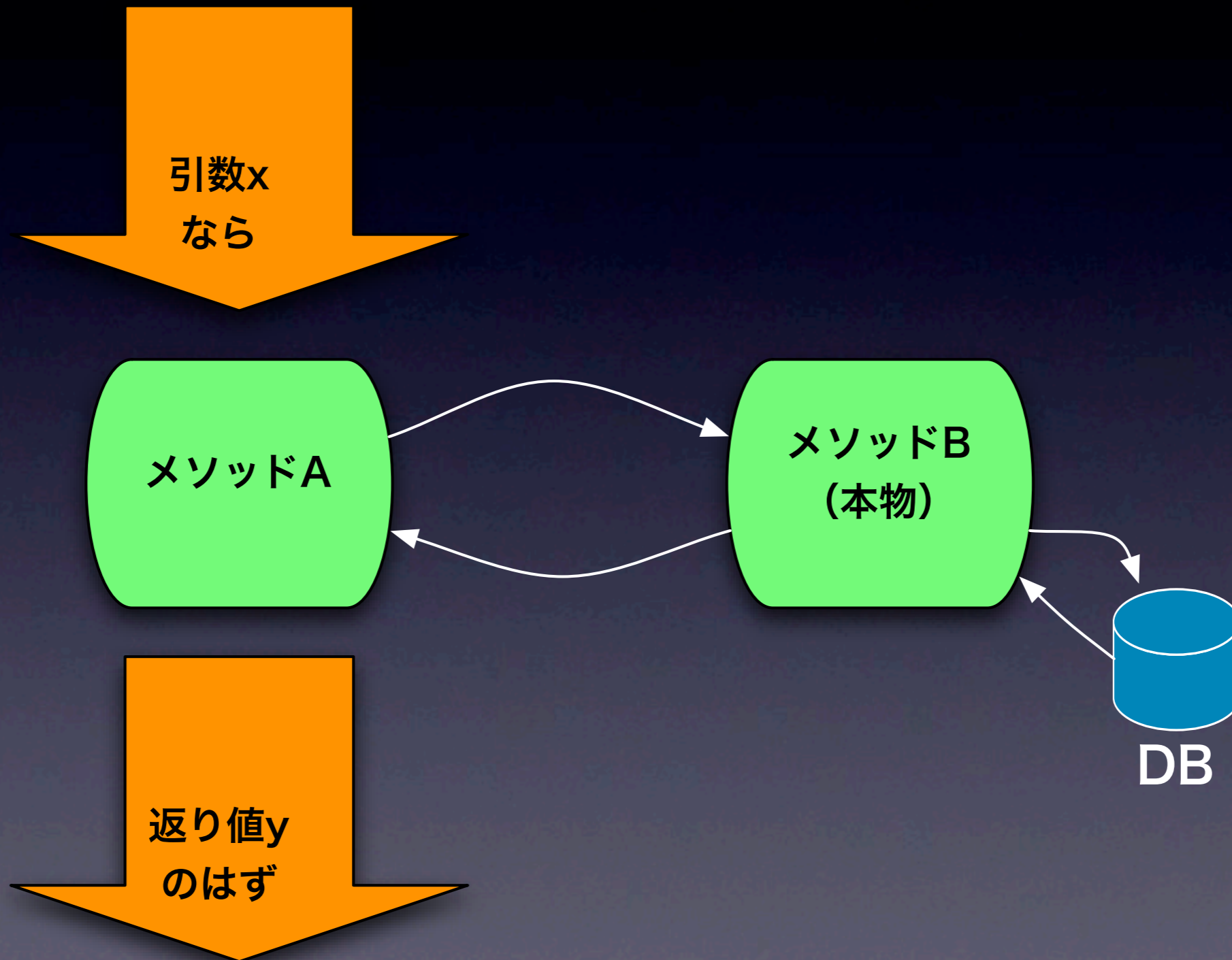


どこの  
ご家庭でも  
共通の悩み

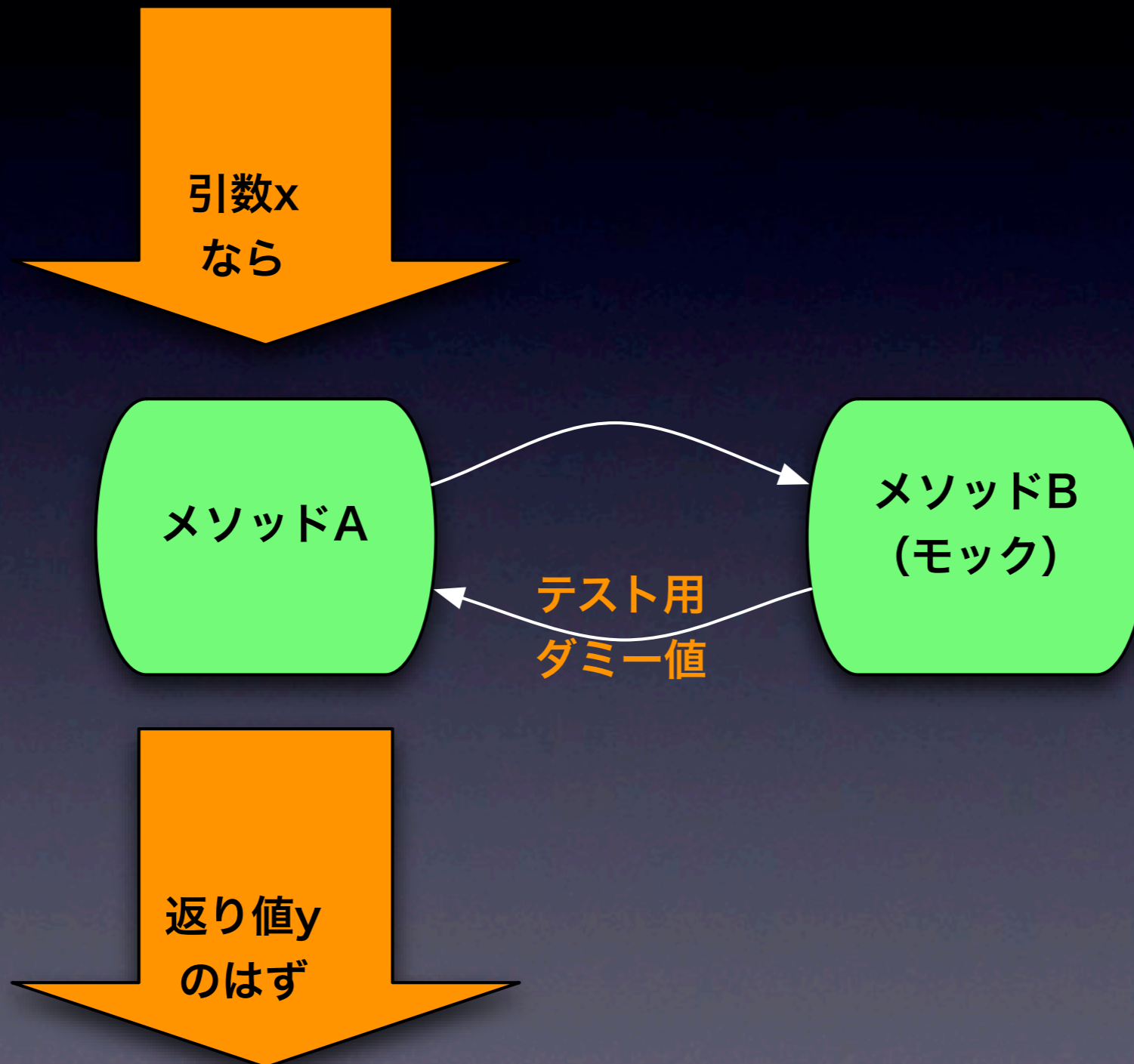
モツクが  
解決します



# モックとは



# モックとは



# モックとは

- 本物のクラスオブジェクトにそっくりな、テスト用のオブジェクト
- 本物と差し替えて使う
- PHPUnit\_MockObject  
→大抵のことが実現可能



# MockObject記述例

```
$mock = $this->getMock('Payment');  
$mock->expects($this->once()  
->method('authorize')  
->with('1111-2222-3333-4444')  
->will($this->returnValue(false));
```

# MockObject記述例

```
$mock = $this->getMock('Payment')  
$mock->expects($this->once())  
->method('authorize')  
->with('1111-2222-3333-4444')  
->will($this->returnValue(false));
```

クラス  
のモックは

# MockObject記述例

```
$mock = $this->getMock('Payment')  
$mock->expects($this->once())
```

クラス  
のモックは

```
->method('authorize')  
->with('1111-2222-3333-4444')  
->will($this->returnValue(false));
```



# MockObject記述例

```
$mock = $this->getMock('Payment')  
$mock->expects($this->once())
```

クラス  
のモックは

```
->method('authorize')
```

メソッドが  
引数

```
->with('1111-2222-3333-4444')
```

で

```
->will($this->returnValue(false));
```

# MockObject記述例

```
$mock = $this->getMock('Payment')  
$mock->expects($this->once())
```

クラス  
のモックは

```
->method('authorize')
```

メソッドが  
引数

```
->with('1111-2222-3333-4444')
```

で

```
->will($this->returnValue(false))
```

返り値  
で

# MockObject記述例

```
$mock = $this->getMock('Payment')  
$mock->expects($this->once())  
->method('authorize')  
->with('1111-2222-3333-4444')  
->will($this->returnValue(false));
```

クラス  
のモックは  
1回呼ばれる  
メソッドが  
引数  
で  
返り値  
で



# getMock()メソッド

- 指定されたクラスのニセモノを作る
- 元のメソッド名・メンバ名は全て維持
  - メソッドの中身は空っぽ

# 何ができるの？

- getMock()で作ったモックに対して
  - メソッドの置き換え
  - 返り値の指定
  - メソッドの呼び出し回数のテスト
  - メソッド呼び出しの引数のテスト

いつでも

使える

わけじゃない



# モックと疎結合(1)

- 密結合だとモックへ差し替えられない

```
function authorize() {  
    $p = new Payment();  
    $p->authorize();  
}
```

# モックと疎結合(2)

- DI設計なら差し替えられる

```
function authorize(Payment $p) {  
    $p->authorize();  
}
```

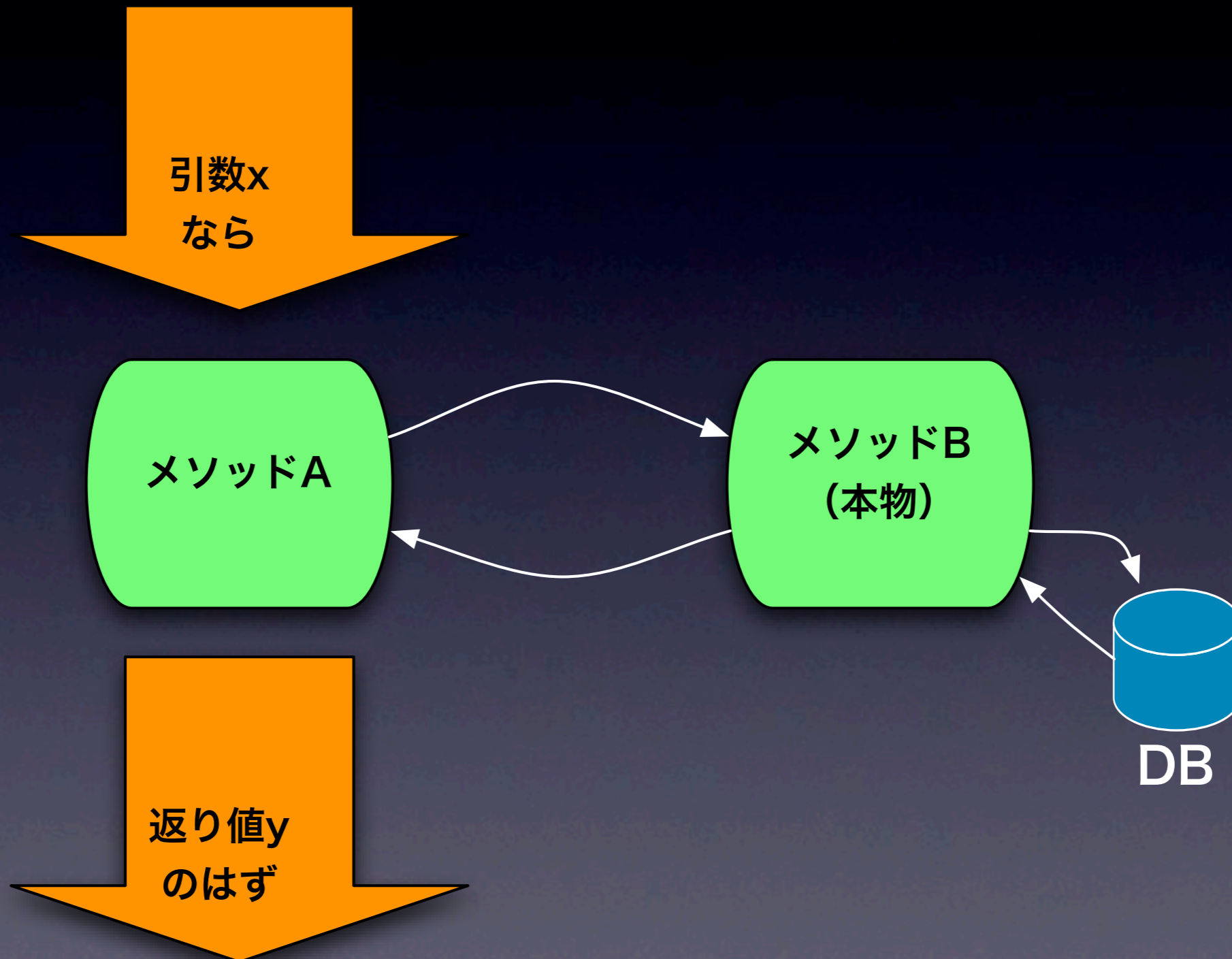
DI

いいいよね

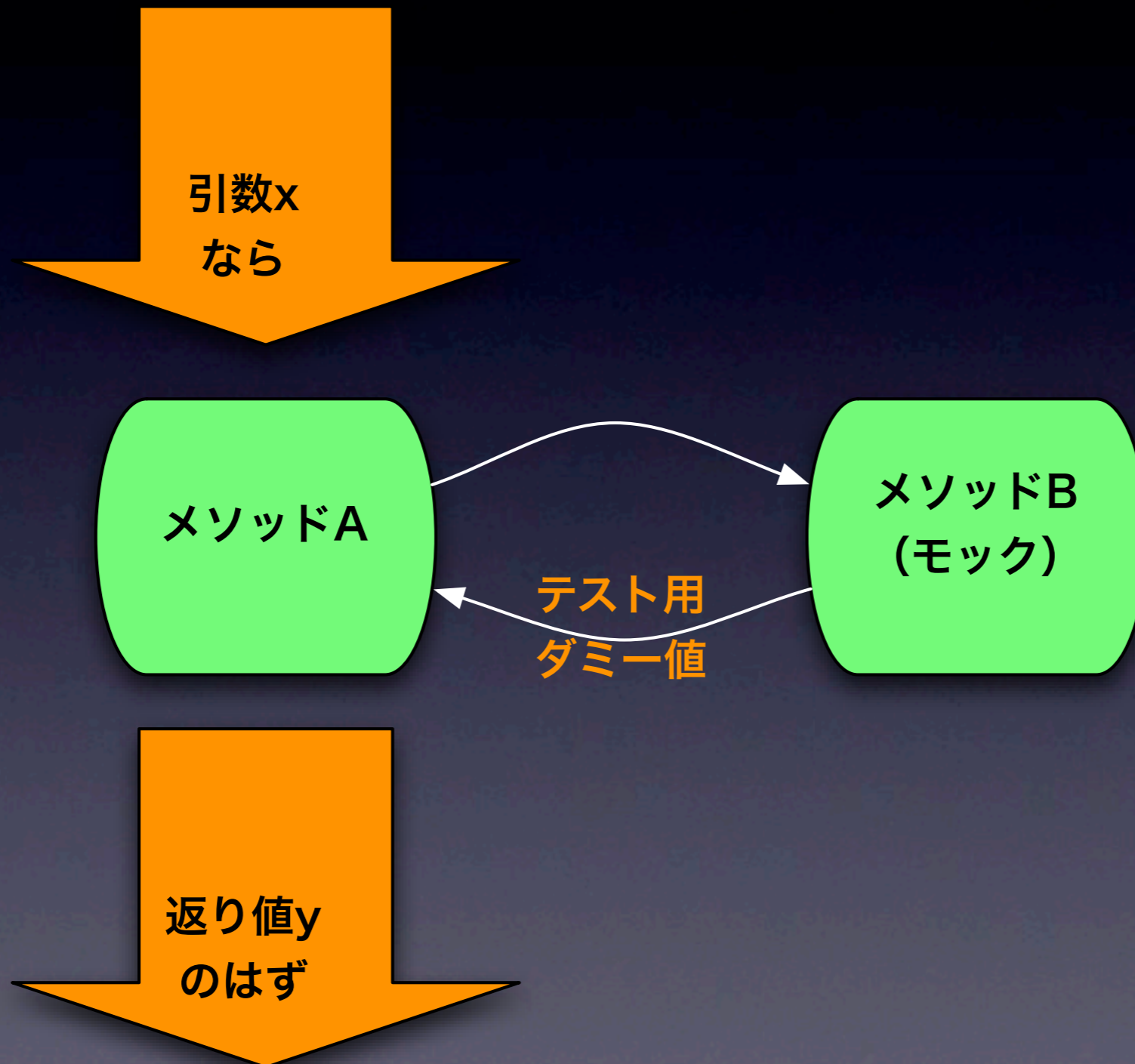


- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ

# テストしにくい！

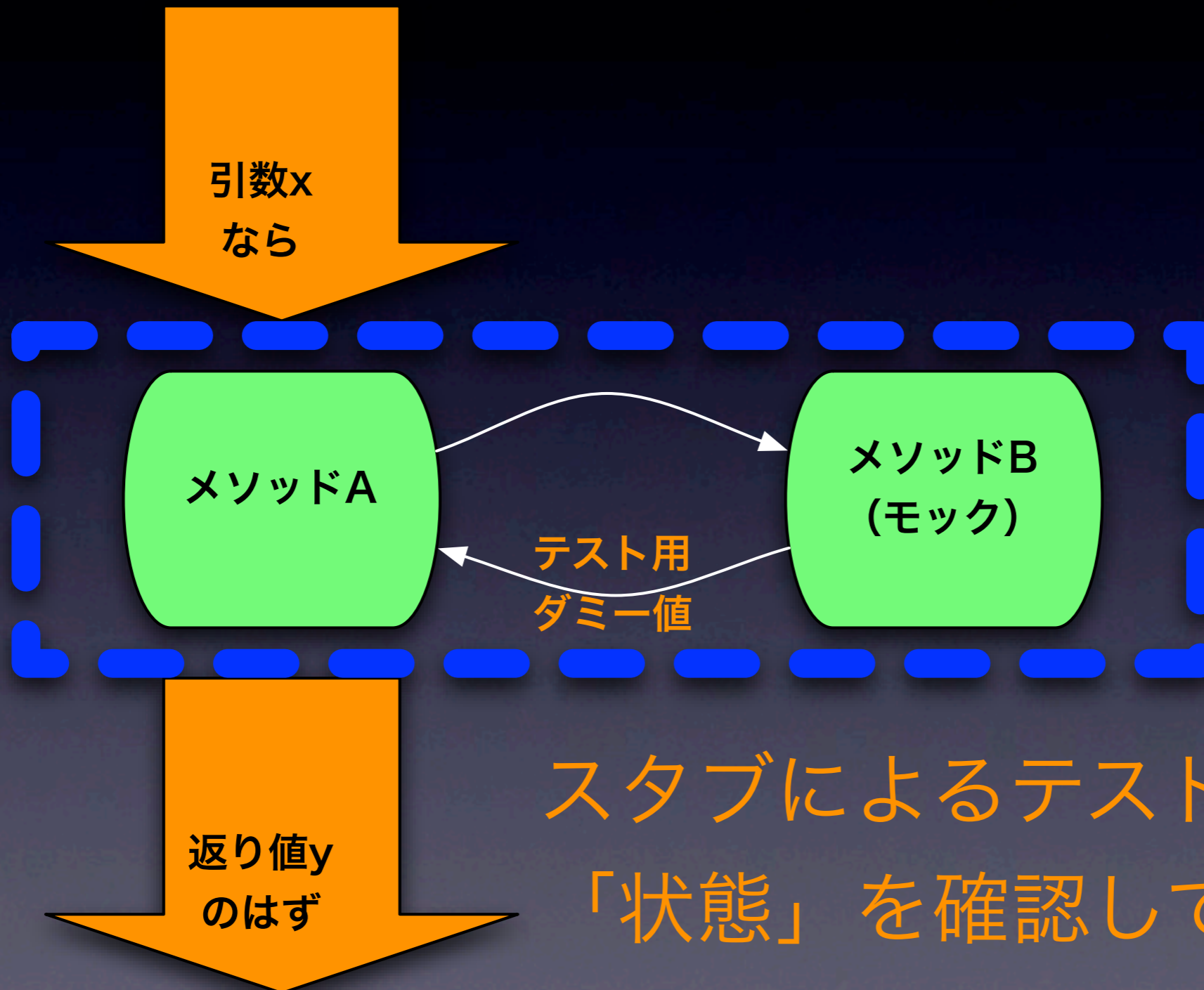


# スタブ化



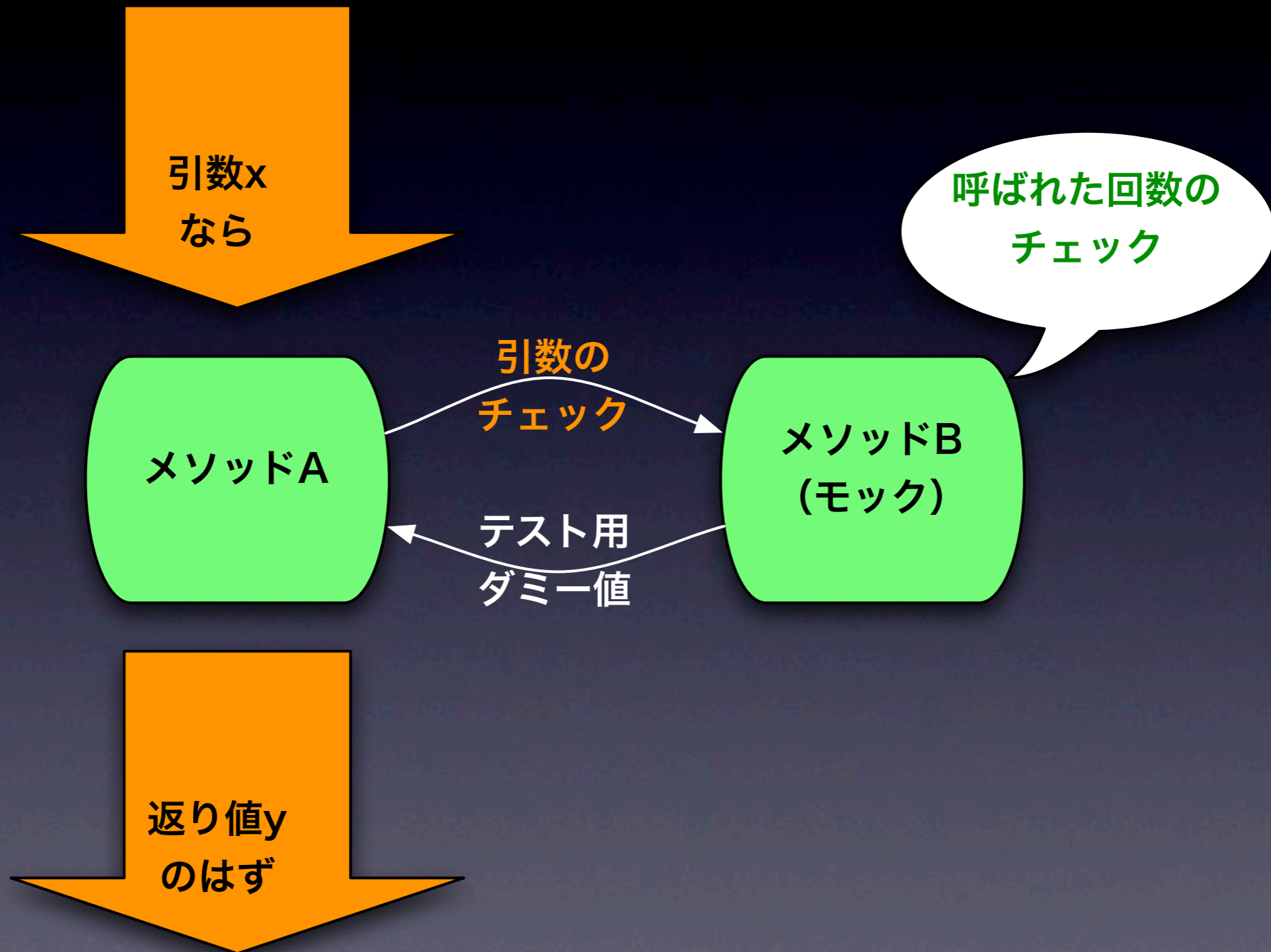


# スタブ化



スタブによるテスト。  
「状態」を確認している

# モック化



# モック化





今ままでとは  
違うテストが  
できるよ！

# Martin Fowlerさん曰く

- モックによるテストに自分は惹かれな  
い。古典的テストで十分。
- 惹かれた人は試せば？肌合う人もい  
る。活躍する場所は確かにある。

—— 「Mocks Aren't Stubs」 より

# 僕個人の意見

- テスト書くのが大変！
- 使いどころ
  - 重要かつ複雑なビジネスロジック



- 自己紹介
- 今日の目標
- PHPUnitの概要
- MockObjectで出来ること
- モックとスタブの違い
- まとめ

# まとめ

- PHPUnit\_MockObjectを使って
  - スタブで古典的テスト
  - モックによる「ふるまい」のテスト
- DI設計と親和性が高い
  - 「差し替えるクラス」はモックも含む

ご清聴  
ありがとうございます  
ございました