Never trust a test that never fails

/thoughtworks

What is it?

/thoughtworks

It's always green

Why is it bad?

/thoughtworks

It hurts!

You lose confidence!

Why should I worry about?

/thoughtworks

It's a tool!

Allow safe changes

How to fail?

/thoughtworks

Code coverage

```php
public static function getFactorialOf(int $number): int {
  if ($number !== 1) {
    return $number * self::getFactorialOf($number - 1);
  }

  return 1;
}

public function testFactorialAlgorithmn(): void {
  Assert::assertEquals(720, self::getFactorialOf(6));
}
```

```php
public static function getFactorialOf(int $number): int {
  return 720;
}

public function testFactorialAlgorithmn(): void {
  Assert::assertEquals(720, self::getFactorialOf(6));
}
```

```php
public static function getFactorialOf(int $number): int {
  if ($number !== 1) {
    return $number * self::getFactorialOf($number - 1);
  }

  return 1;
}

public function testFactorialAlgorithmn(): void {
  Assert::assertEquals(720, self::getFactorialOf(6));
  Assert::assertEquals(120, self::getFactorialOf(5));
}
```

Should I drop it?

/thoughtworks

No!

Mutation tests

/thoughtworks

```php
public function greatherThan100(int $number): string {
  if ($number > 100) {
    return 'Yes, it is!';
  }

  return 'No, it is not!'
}
```

```php
public function greatherThan100(int $number): string {
  if ($number < 100) {
    return 'Yes, it is!';
  }

  return 'No, it is not!'
}
```

```php
public function greatherThan100(int $number): string {
  if ($number > 120) {
    return 'Yes, it is!';
  }

  return 'No, it is not!'
}
```

How to write?

/thoughtworks

It only fails for one reason

Courage

Use TDD

Run all the time

Analyze your code

Throw code away

A few tips

/thoughtworks

```php
public function testRefund(): void {
  $payment = createPayment();

  $payment->refund(self::DEFAULT_REFUND_AMOUNT);

  Assert::assertEquals(100, $payment->getOriginalAmount());
  Assert::assertEquals(70, $payment->getBalance());
}
```

```php
public function testWhenRefunding_ShouldDecreaseFromPaymentAmount(): void {
  $payment = createPayment();

  $payment->refund(self::DEFAULT_REFUND_AMOUNT);

  Assert::assertEquals(100, $payment->getOriginalAmount());
  Assert::assertEquals(70, $payment->getBalance());
}
```

```php
public function testWhenRefunding_ShouldDecreaseFromPaymentAmount(): void {
  $payment = new Payment(
    $customer_name = 'Ivo Batistela',
    $customer_email = 'ivo.batistela@thoughtworks.com',
    $amount = 100
  );

  $payment->refund(self::DEFAULT_REFUND_AMOUNT);

  Assert::assertEquals(100, $payment->getOriginalAmount());
  Assert::assertEquals(70, $payment->getBalance());
}
```

```php
public function testWhenRefunding_ShouldDecreaseFromPaymentAmount(): void {
  $payment = self::createPayment($amount = 100);

  $payment->refund($refund_amount = 30);

  Assert::assertEquals(100, $payment->getOriginalAmount());
  Assert::assertEquals(70, $payment->getBalance());
}
```

```php
public function testWhenRefunding_ShouldDecreaseFromPaymentAmount(): void {
  $payment = self::createPayment($amount = 100);

  $payment->refund($refund_amount = 30);

  Assert::assertEquals(70, $payment->getBalance());
}
```

Unexpected test failures

A good test is documentation

Thank you!

Questions?

Ivo Batistela
Consultant
ivo.batistela@thoughtworks.com

/thoughtworks