AssertTrue(isDecoupled("MyTests"))

Dave Liddament

SHARE
YOUR
STORY

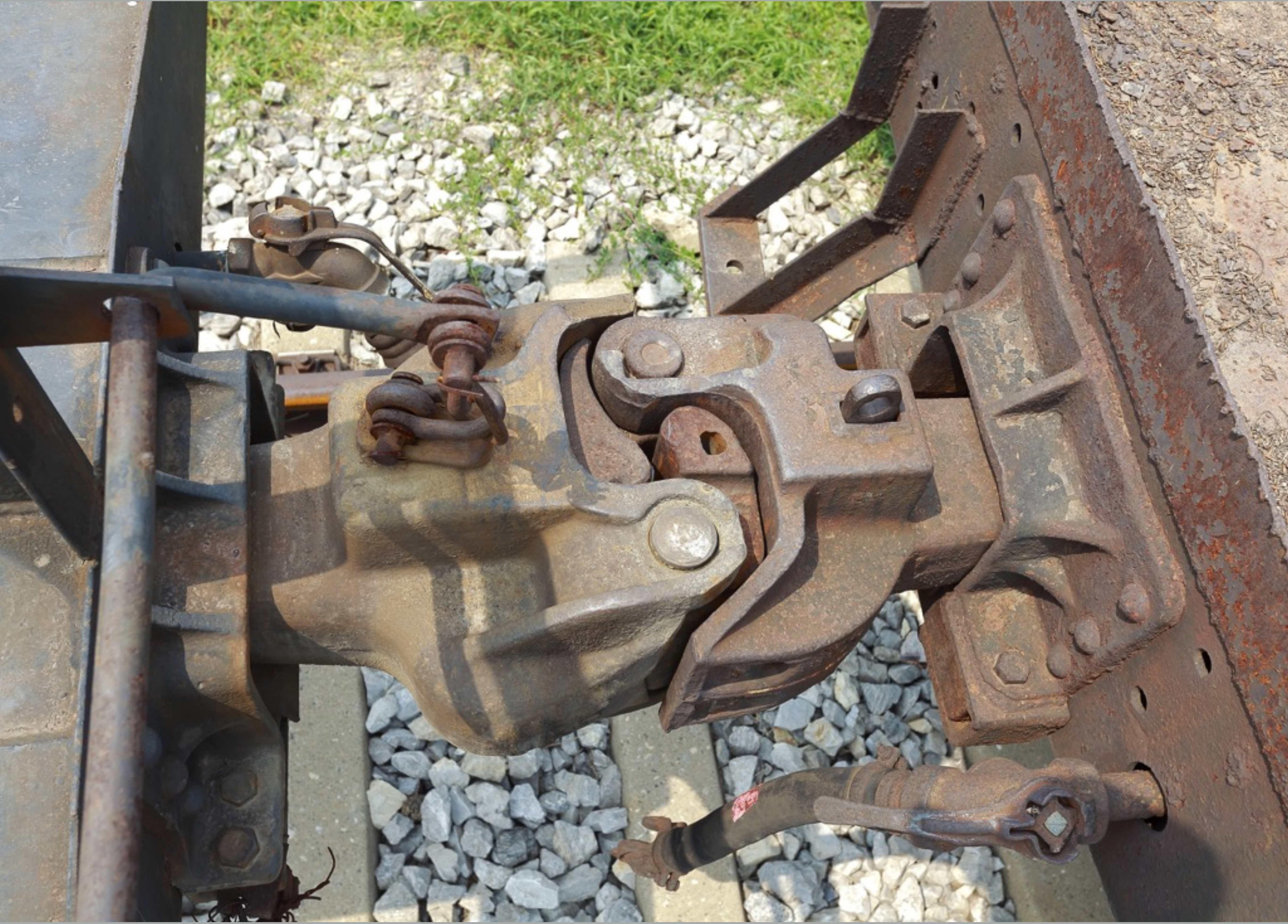
```
...............................................................  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
............................................................... 189 / 444 ( 42%)
............................................................... 252 / 444 ( 56%)
............................................................... 315 / 444 ( 70%)
............................................................... 378 / 444 ( 85%)
............................................................... 441 / 444 ( 99%)
...

Time: 1.99 seconds, Memory: 24.75MB

OK (444 tests, 1201 assertions)
```

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 1.55 seconds, Memory: 24.75MB
```

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 1.55 seconds, Memory: 24.75MB


 There were lots of failures:
```

```
................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.  63 / 444 ( 14%)
............................................................... 126 / 444 ( 28%)
.........................FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...... 189 / 444 ( 42%)
FFFFFFFFFFFF................................................... 252 / 444 ( 56%)
...................FF....FFFF.......FFFFFFFFFFFFFF............. 315 / 444 ( 70%)
..................................FFFFFFFFFFFFFFFFFFFFFFFFFFF.. 378 / 444 ( 85%)
FFFFFFFFFFFFFF........FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 441 / 444 ( 99%)
...

Time: 1.55 seconds, Memory: 24.75MB
```

There were lots of failures: 😞



DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

Dave Liddament

@daveliddament

Lamp Bristol

Organise PHP-SW and Bristol PHP Training

AGENDA

AGENDA

- Why

AGENDA

- Why
- Terminology

AGENDA

- Why
- Terminology
- Do the right kind of tests at the right level

AGENDA

- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests

AGENDA

- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests
- Building objects

AGENDA

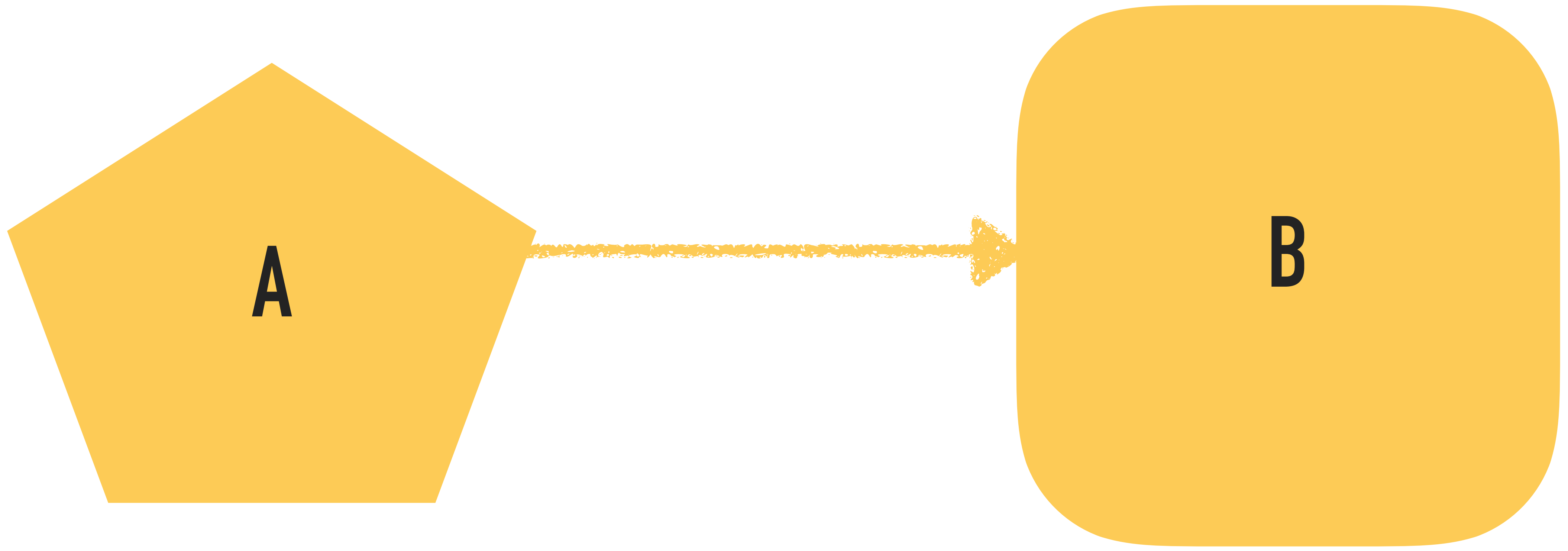
- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests
- Building objects
- Tips

AGENDA

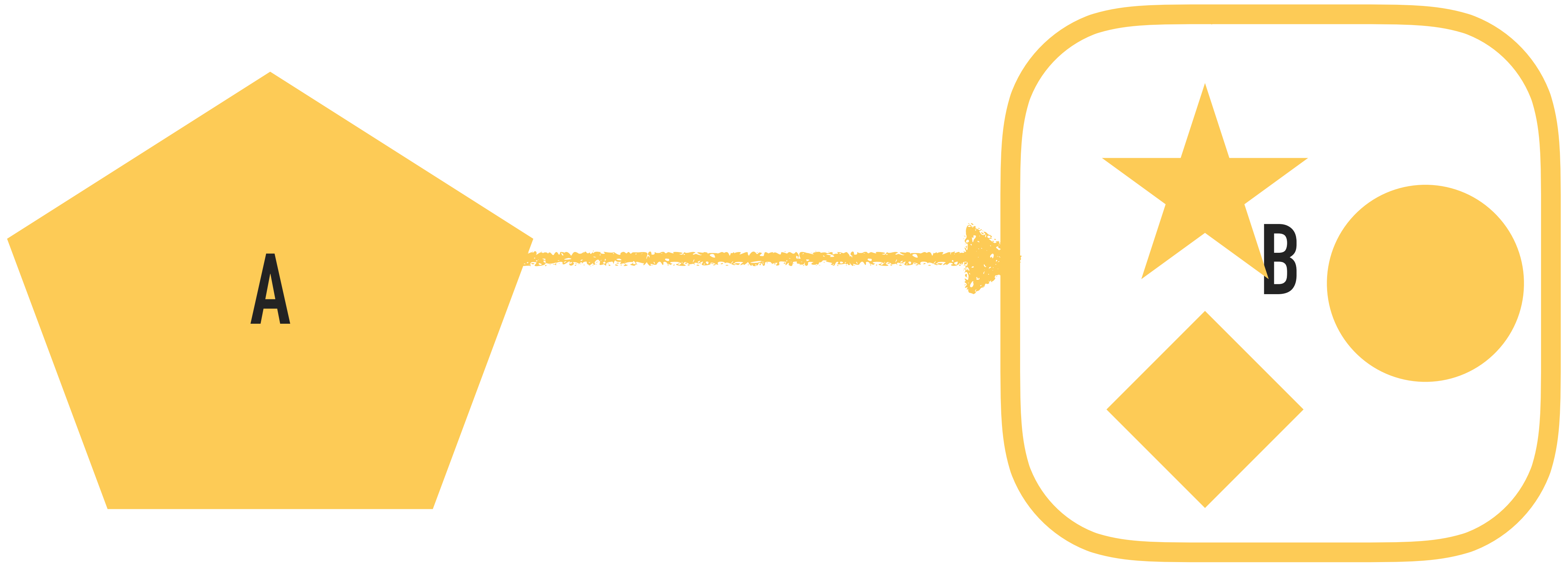
- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests
- Building objects
- Tips
- Summary

COUPLING

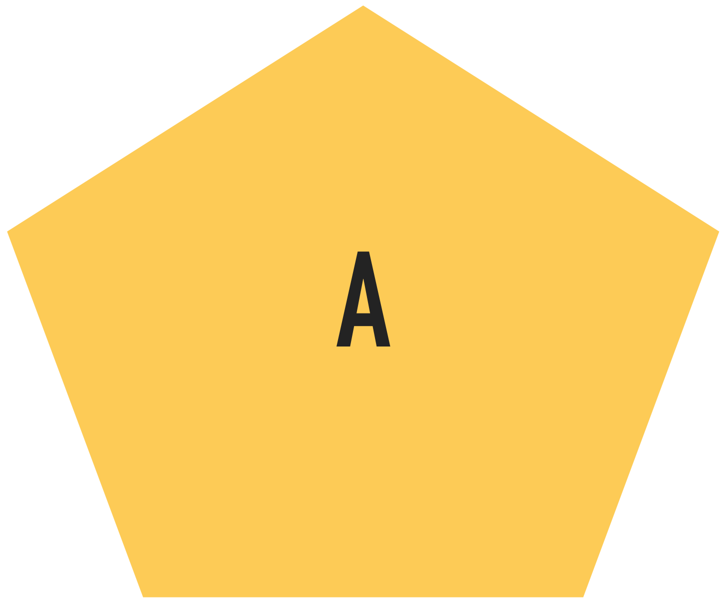
COUPLING



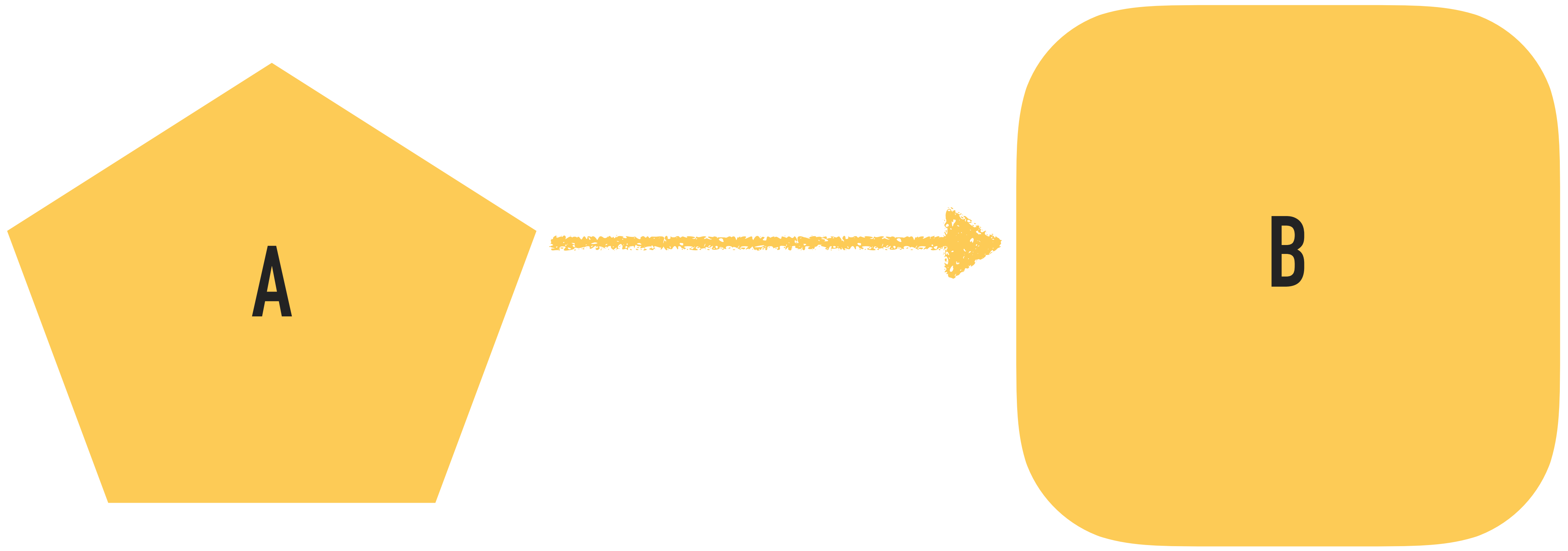
COUPLING



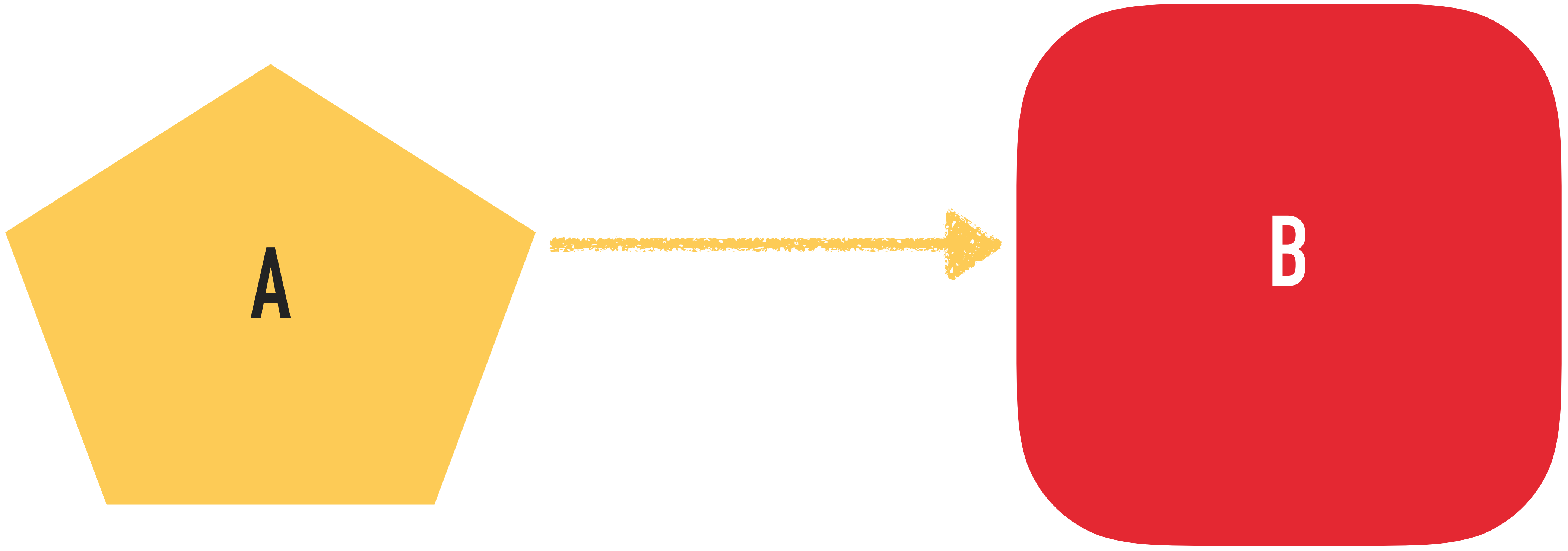
TEST DOUBLES



TEST DOUBLES



TEST DOUBLES



TEST DOUBLES

TEST DOUBLES

- Dummy

TEST DOUBLES

- Dummy
- Stub

TEST DOUBLES

- Dummy
- Stub
- Spy

TEST DOUBLES

- Dummy
- Stub
- Spy
- Mock

TEST DOUBLES

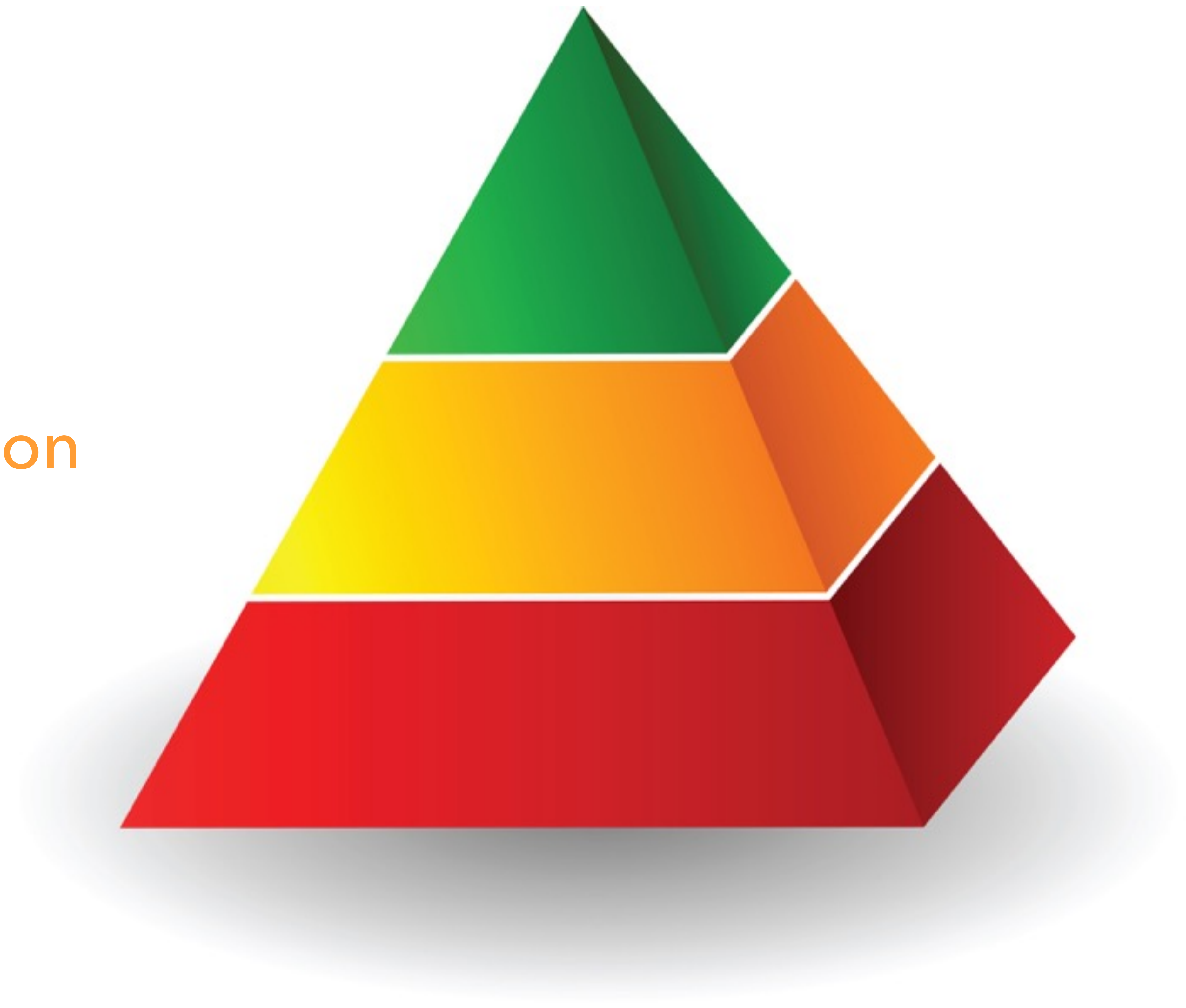
- Dummy
- Stub
- Spy
- Mock
- Fake

TEST PYRAMID

UI

Integration

Unit



AGENDA

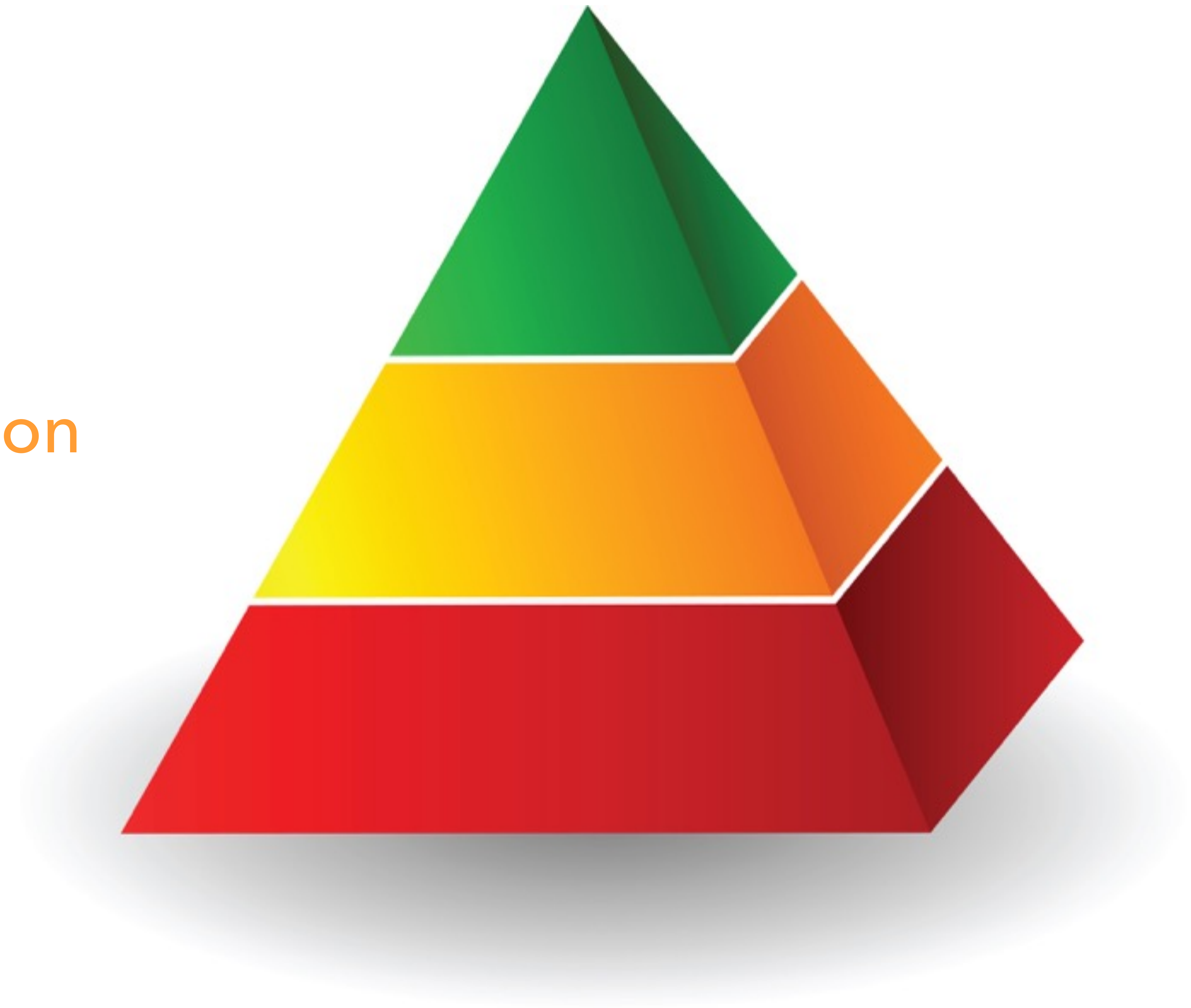
- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests
- Building objects
- Tips
- Summary

DO THE RIGHT KIND OF TESTS AT THE RIGHT LEVEL

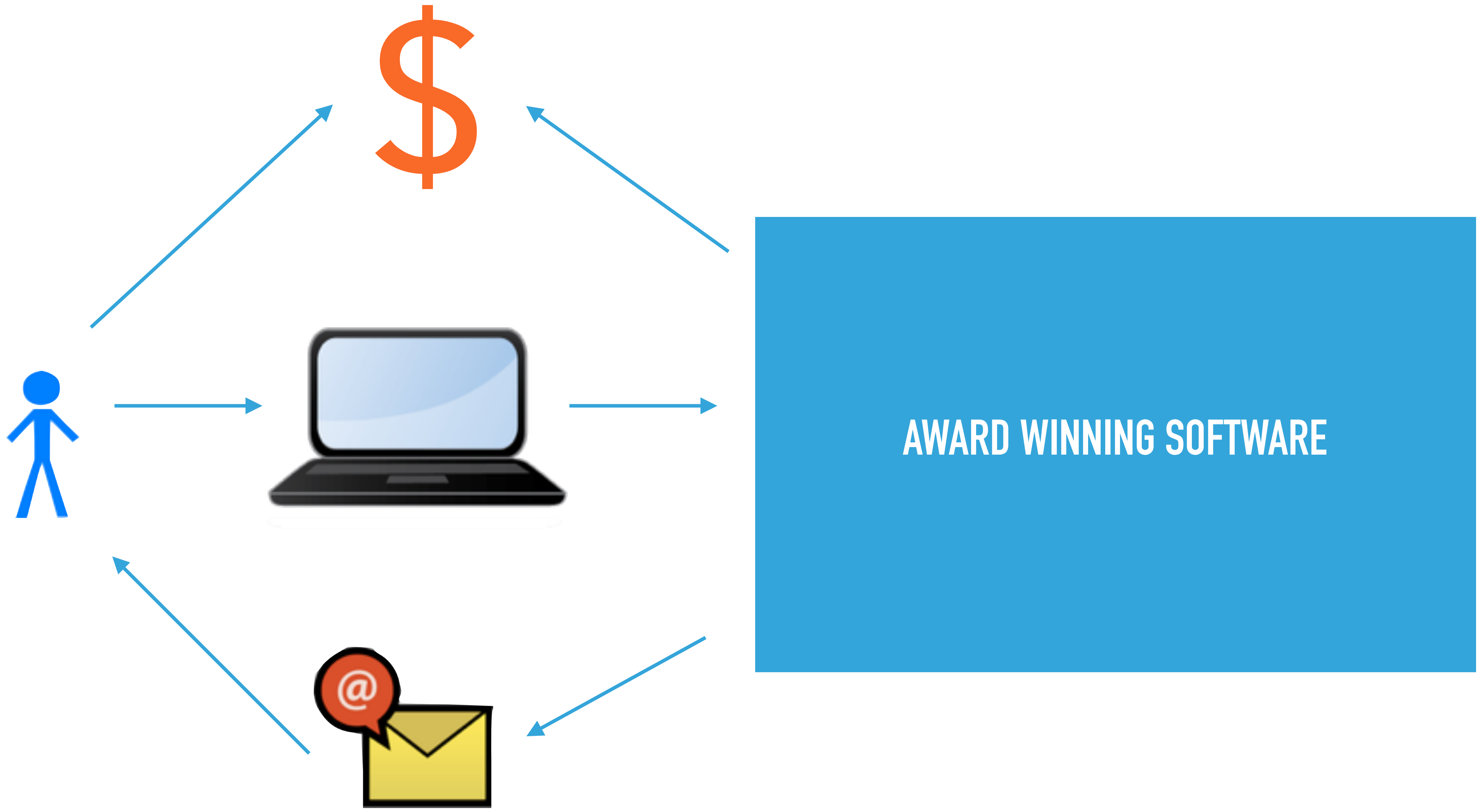
UI

Integration

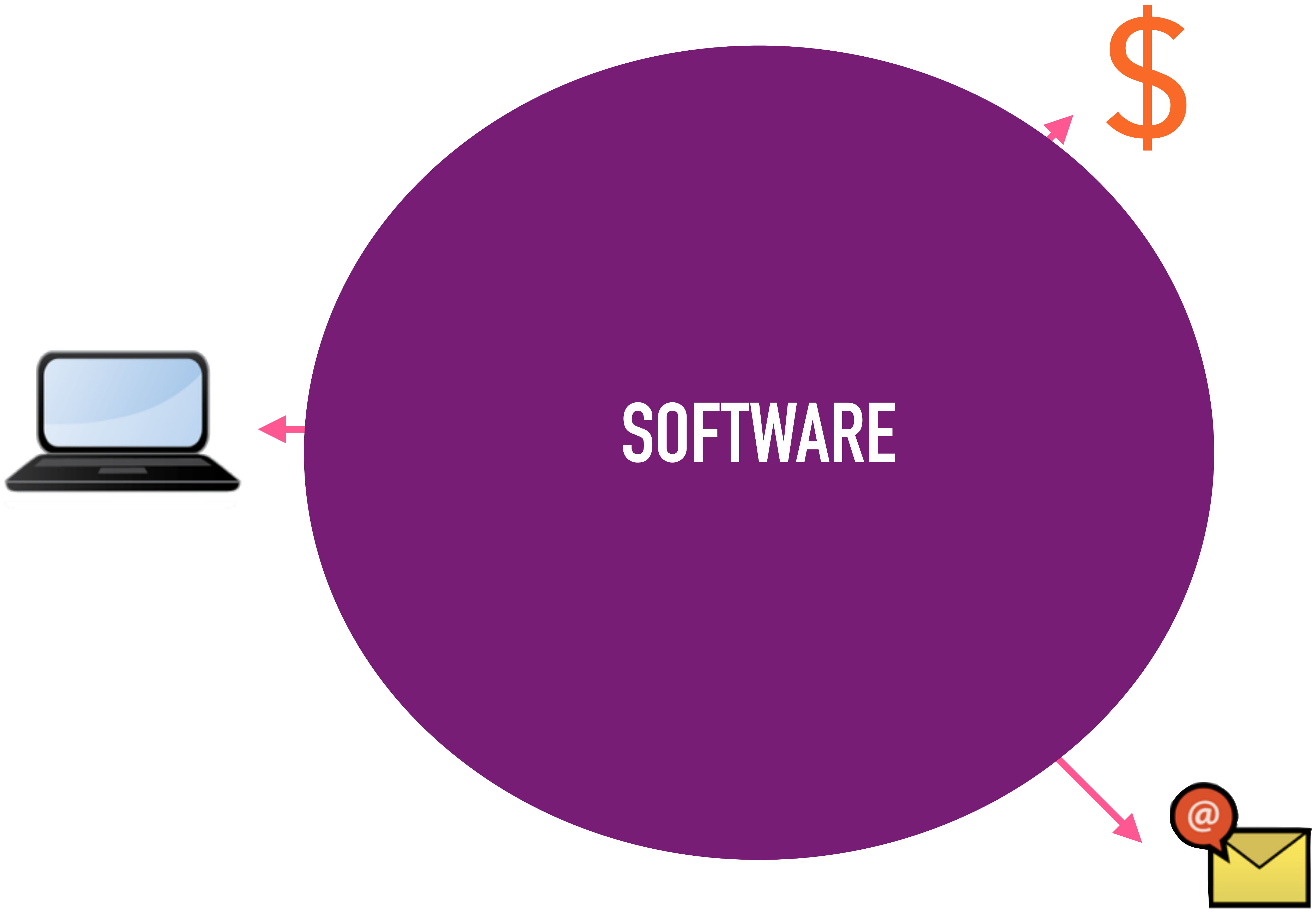
Unit



DECOUPLING



DECOUPLING



DECOUPLING

DECOUPLING

DECOUPLING

DECOUPLING

DECOUPLING

DECOUPLING

Online Banking
Login:
Password:
Security Code:
12-34-56
SUBMIT
BANK

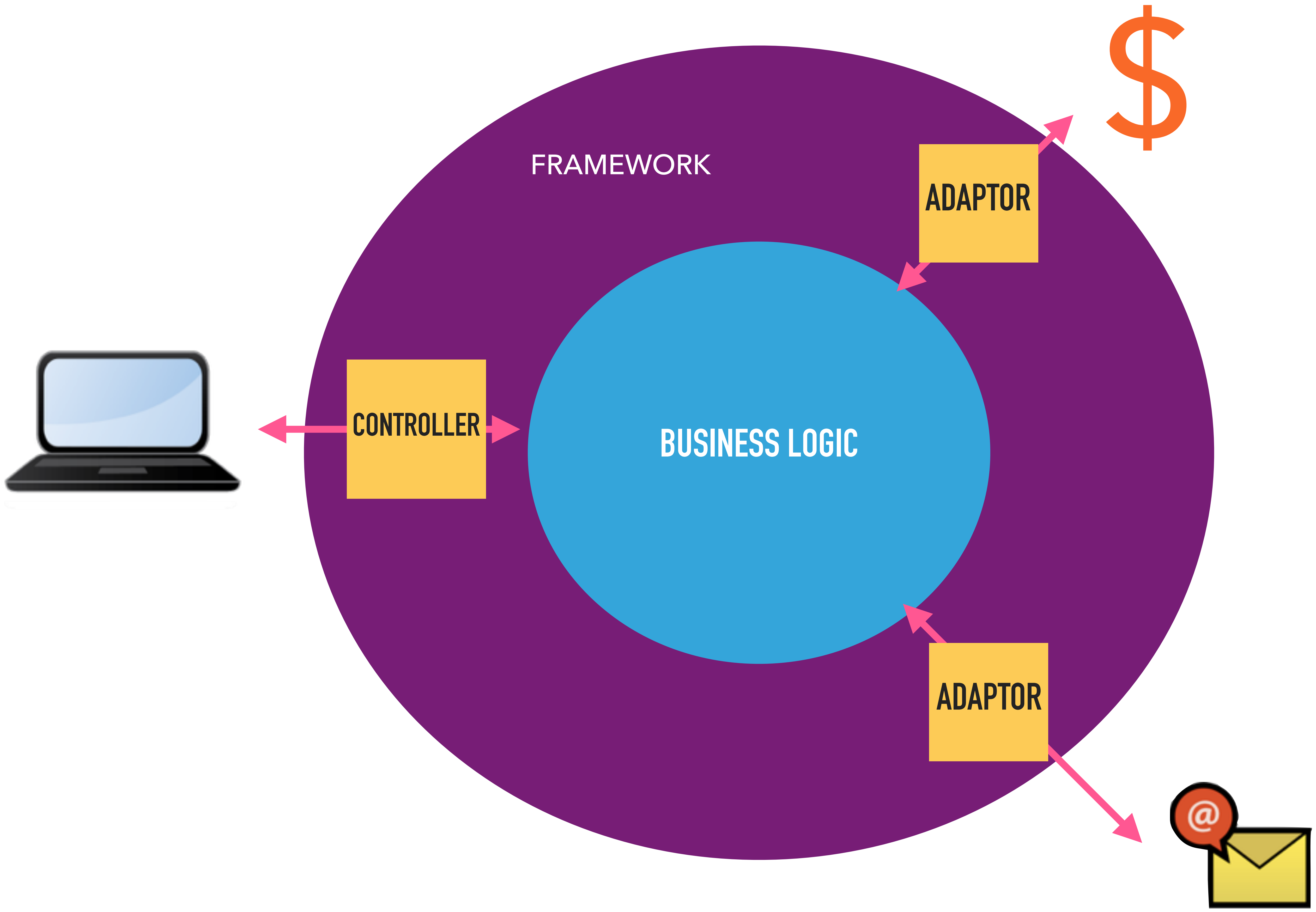
DECOUPLING

DECOUPLING

✓

ONE SMALL CHANGE TO IN THE USER INTERFACE

DECOUPLING



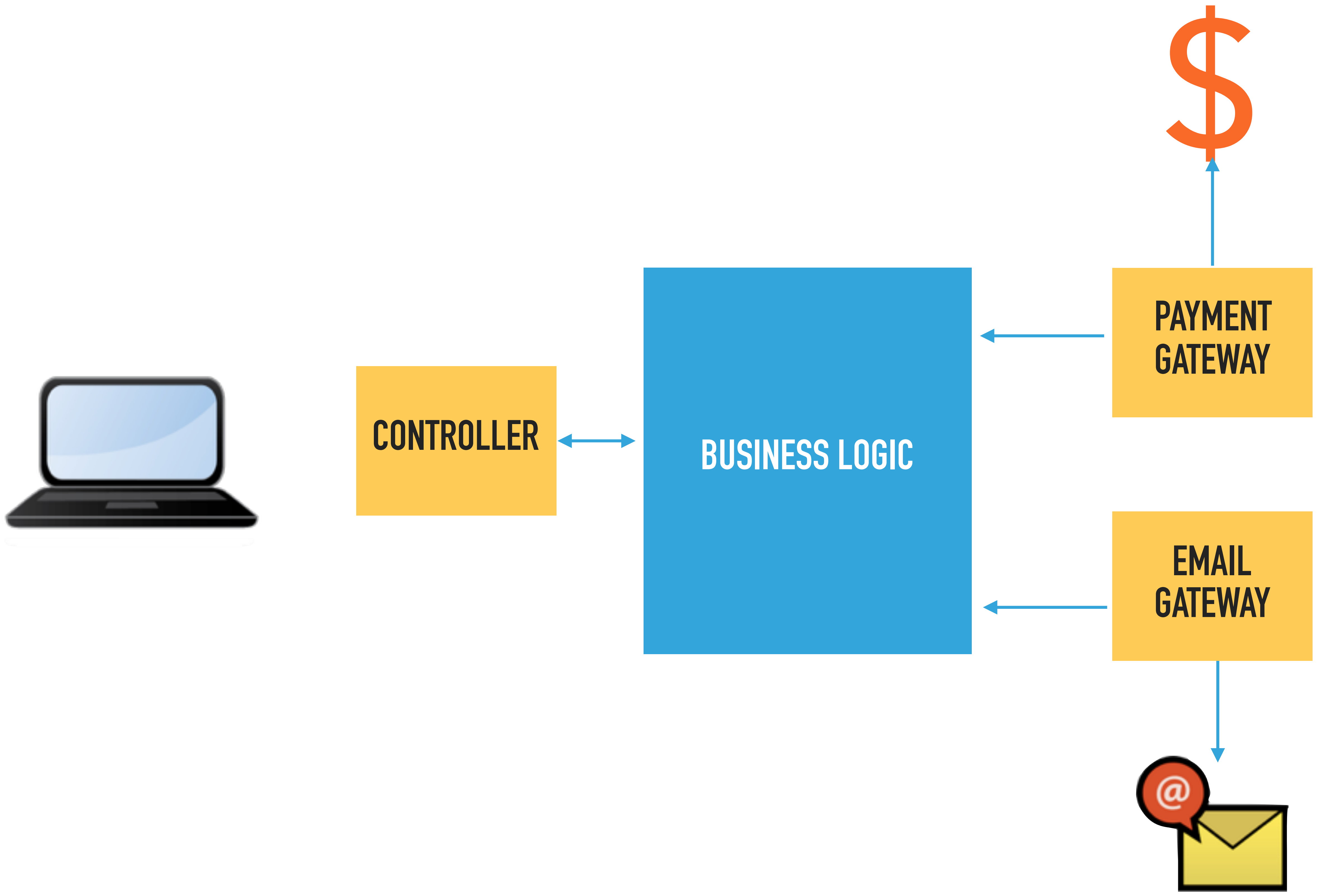
SERVICE LAYER

```
interface PasswordService
{
  /**
   * Send user link to reset their password
   */
  public function requestPasswordReset($emailAddress): void;

  /**
   * Reset password from link
   */
  public function resetPassword($token, $newPassword): bool;

  /**
   * Normal password reset
   */
  public function updatePassword($user, $newPassword): bool;

}
```

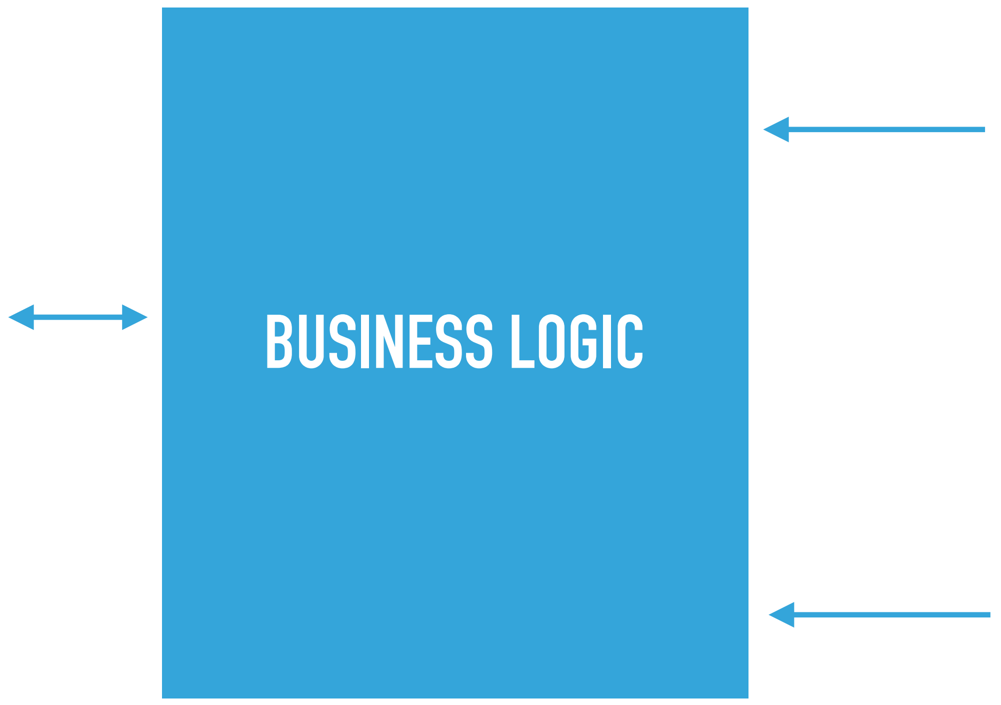
EMAIL GATEWAY

```
interface EmailGateway
{

  /**
   * Send an email to a user
   */
  public function sendEmail(
      $to,
      $from.
      $subject,
      $message
  ): void;

}
```

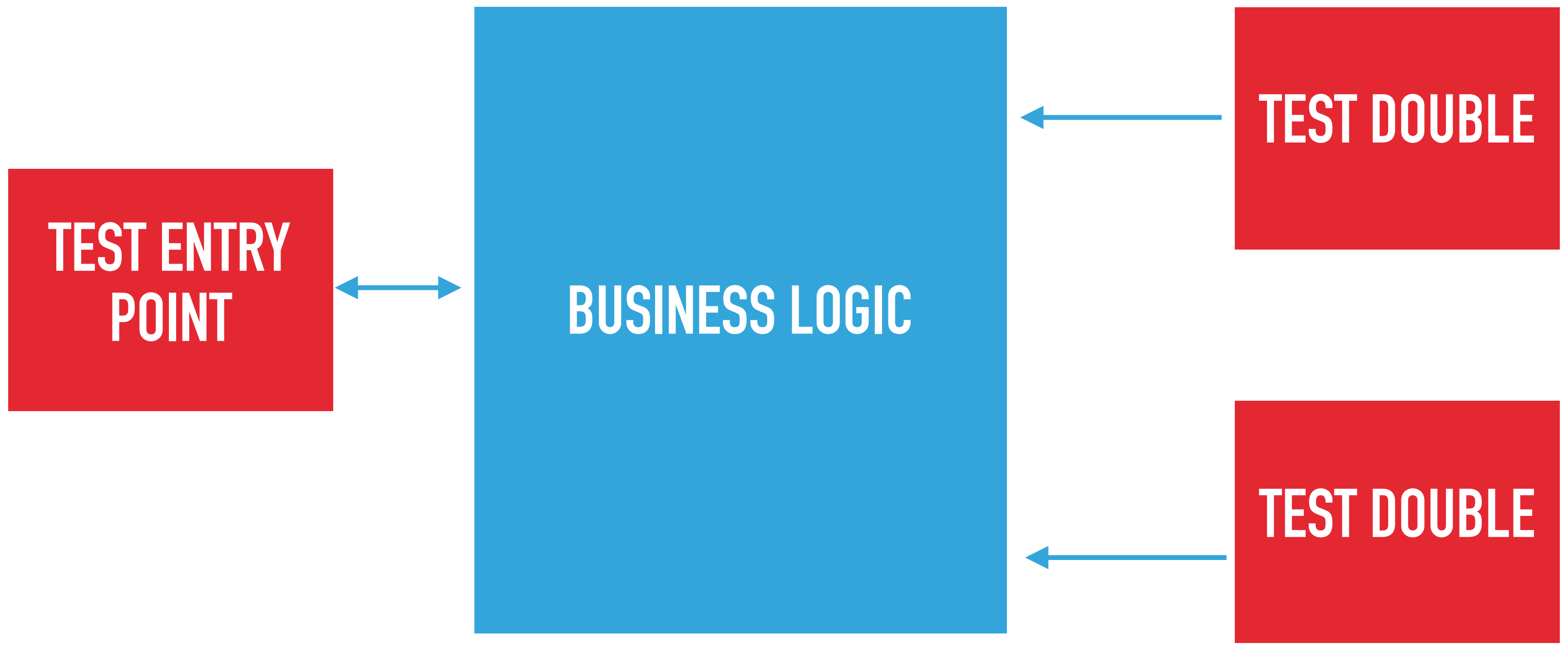
ARCHITECTURE



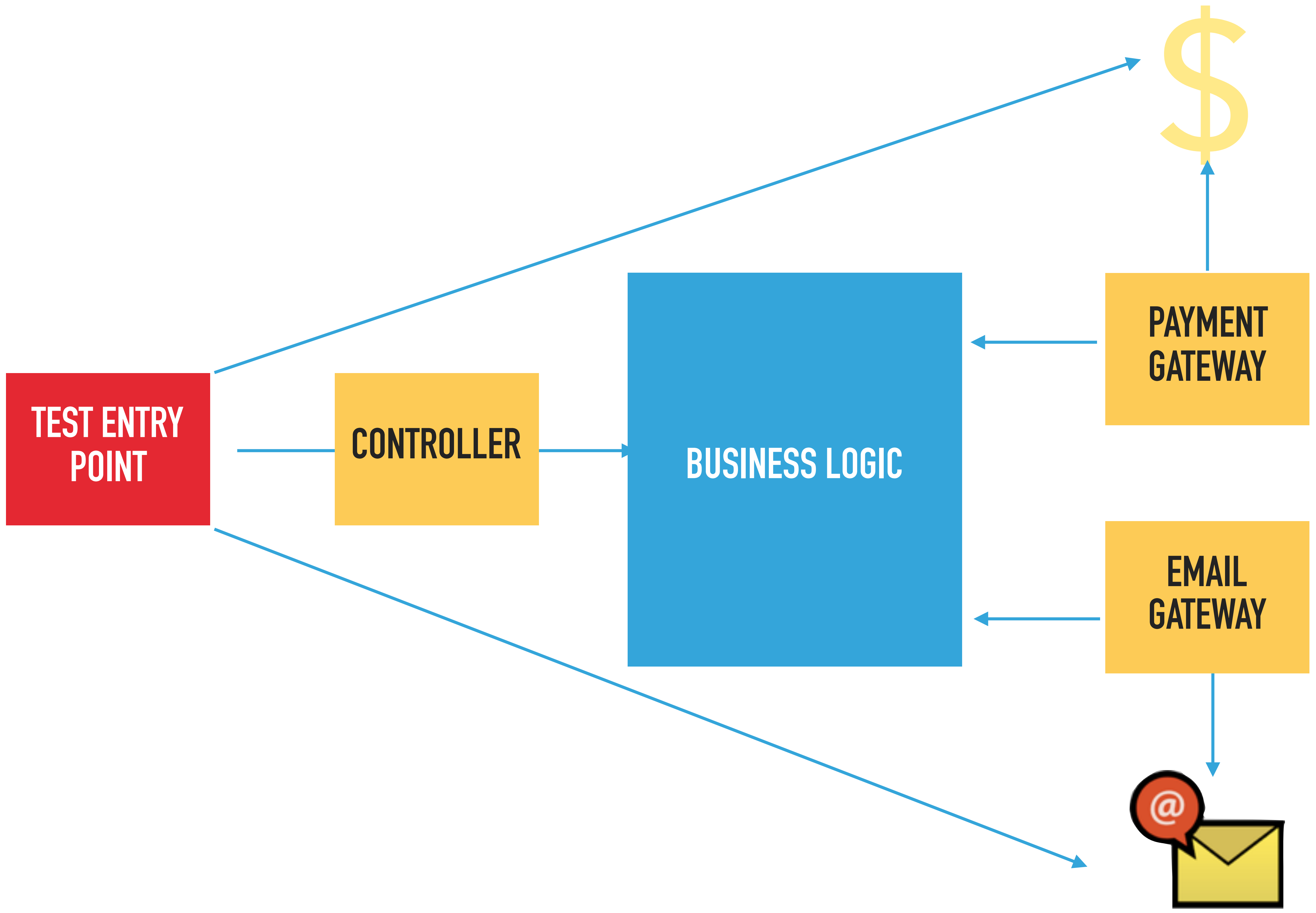
ARCHITECTURE



ARCHITECTURE



DECOUPLING


TEST ENTRY POINT
CONTROLLER
BUSINESS LOGIC
PAYMENT GATEWAY
EMAIL GATEWAY
$
@

TAKE AWAY

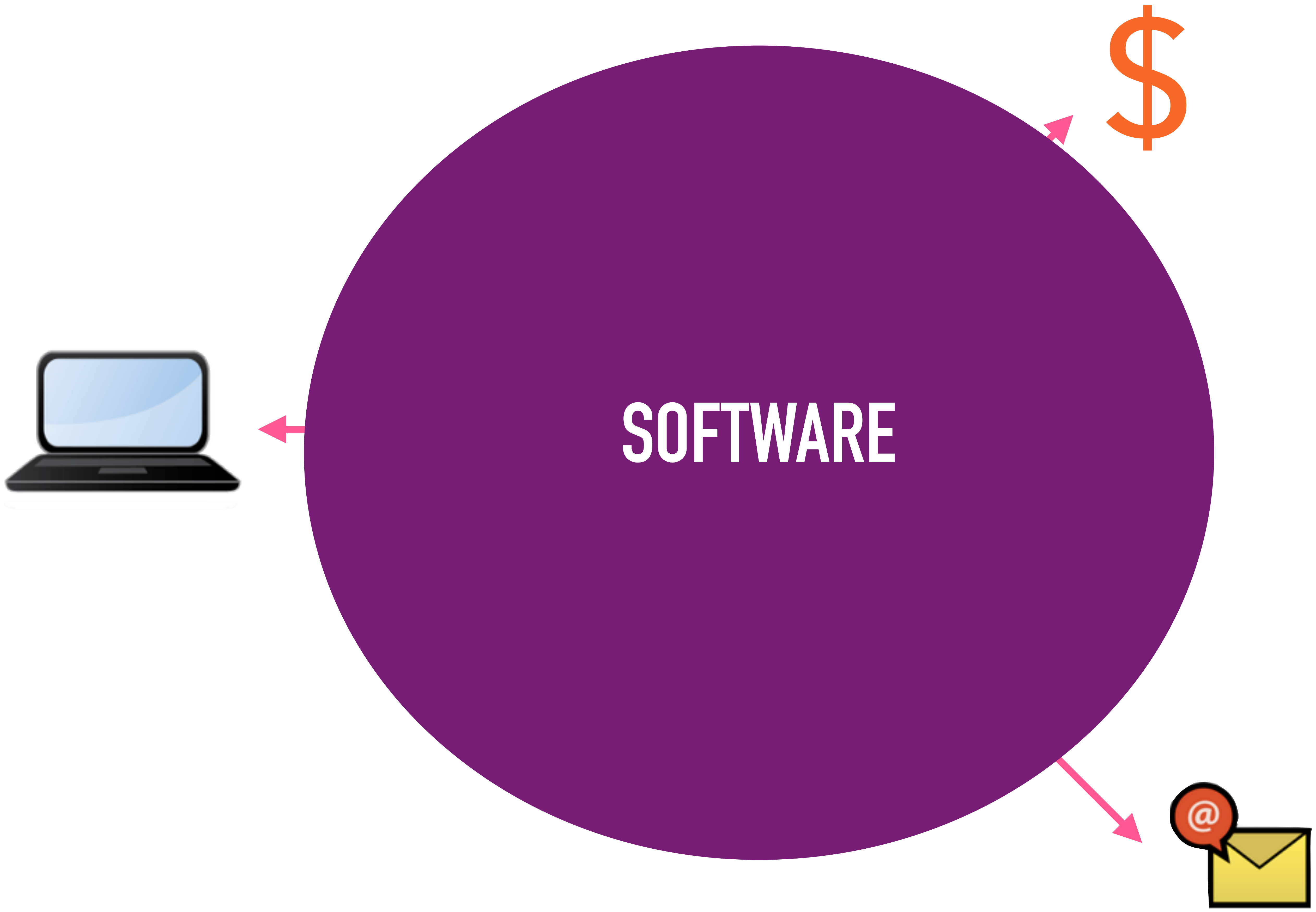
Business Logic

DECOUPLING

DECOUPLING

DECOUPLING

DECOUPLING



DECOUPLING

COUPLED TEST

- Go to web home page
- Login as user Bob
- Enter “Clean Code” into search box
- Iterate through results to find book
- Click add to basket
- Click checkout
- Enter payment details
- Click confirm
- Enter delivery address
- Click confirm
- Enter next day delivery option
- Check price includes additional delivery charge

DECOUPLED TEST

- Given I have a shopping basket containing “Clean Code”

  (NOTE: this book costs £10)
- When I check out with “Free delivery”
- Then I should be charged £10

DECOUPLED TEST (2)

- Given I have a shopping basket containing “Clean Code”

  (NOTE: this book costs £10)

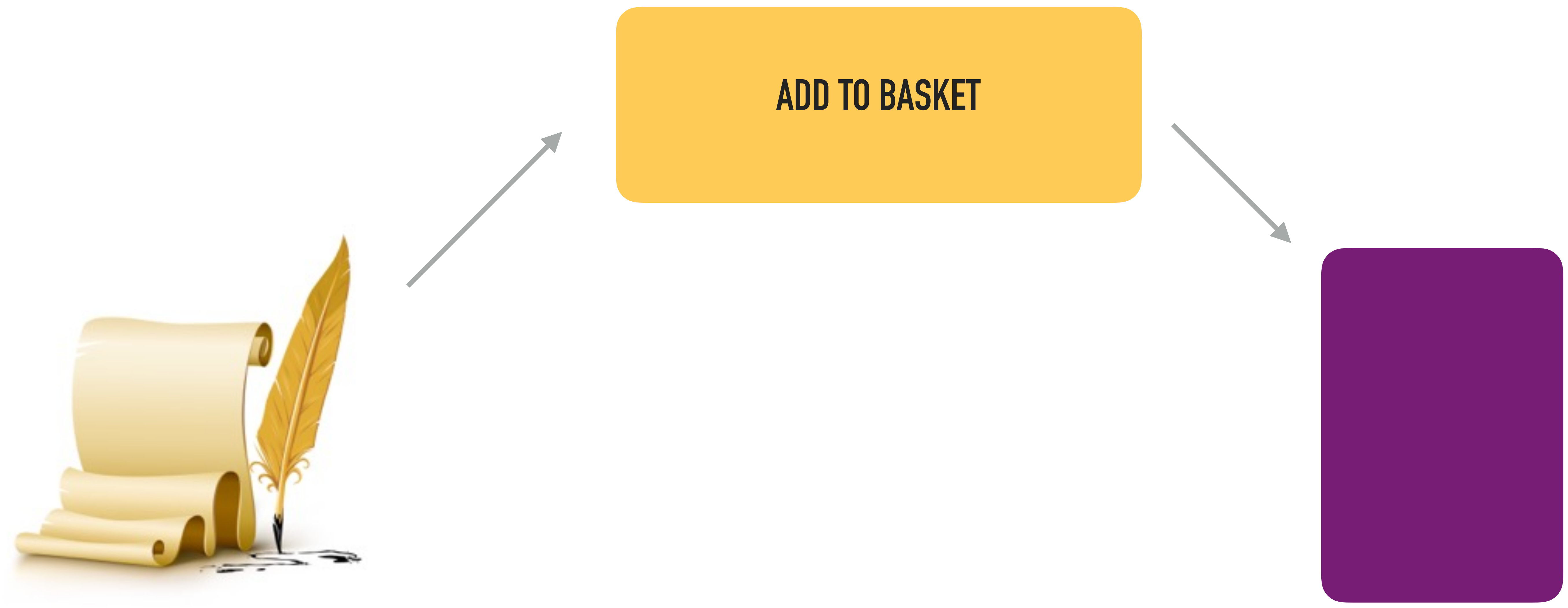
- When I check out with “Next day delivery”

  (NOTE: this costs £5)

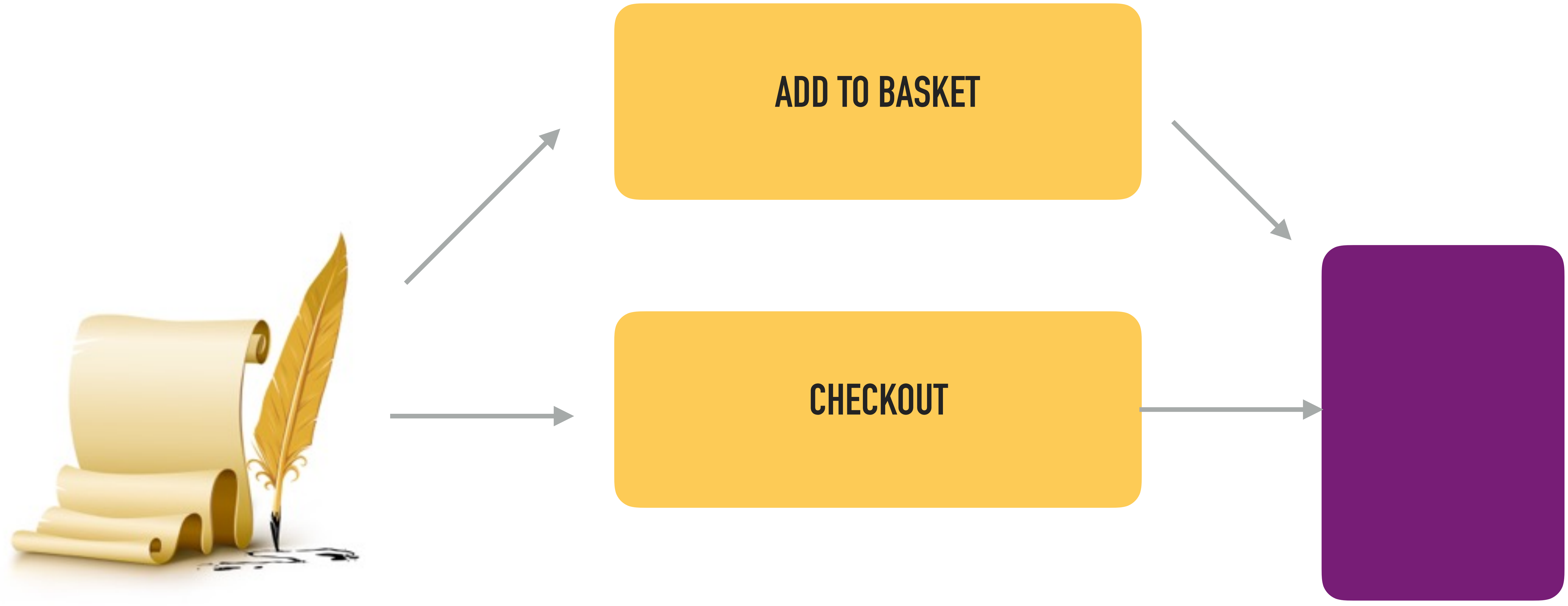
- Then I should be charged £15

DECOUPLING

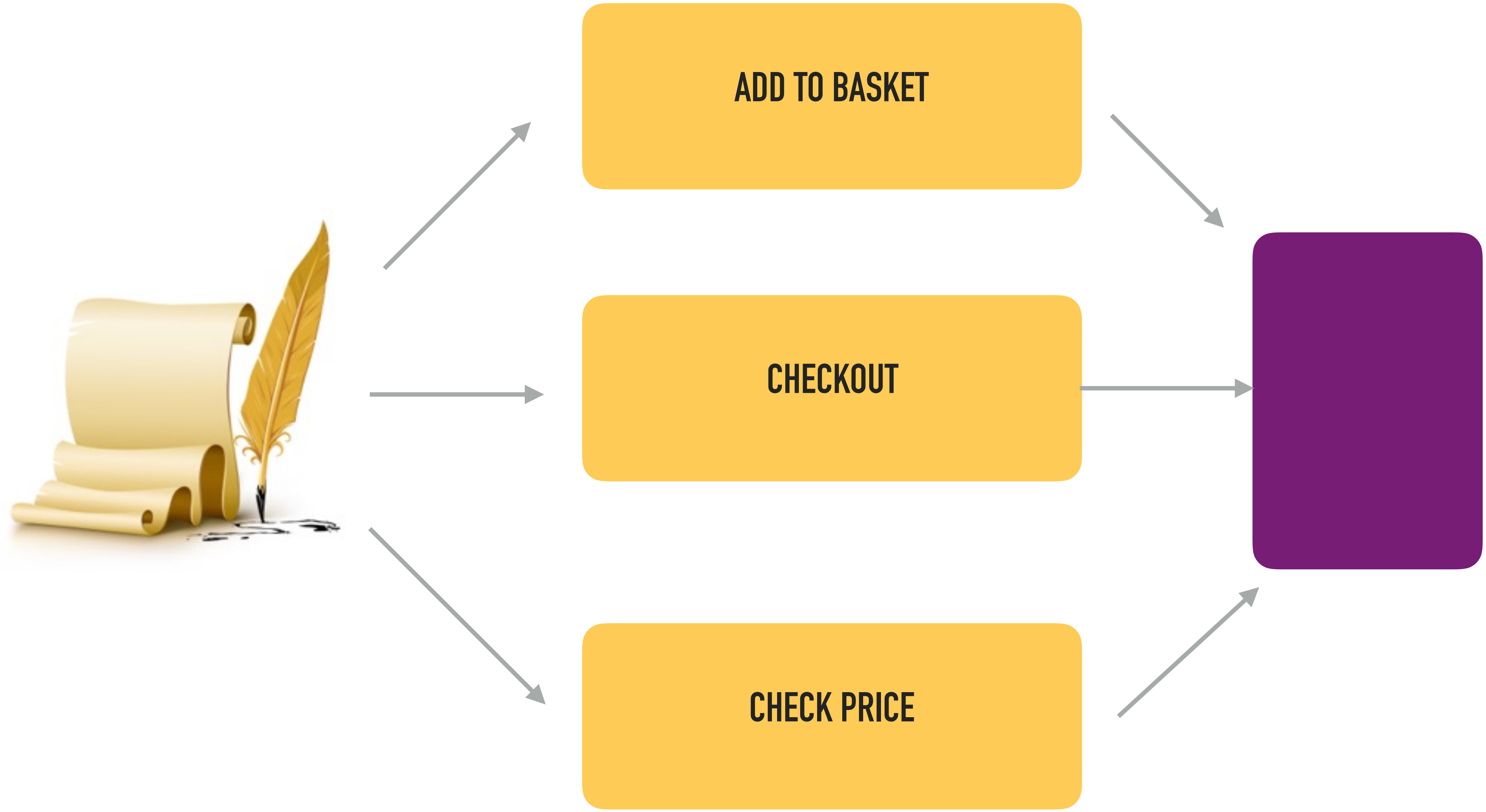
DECOUPLING



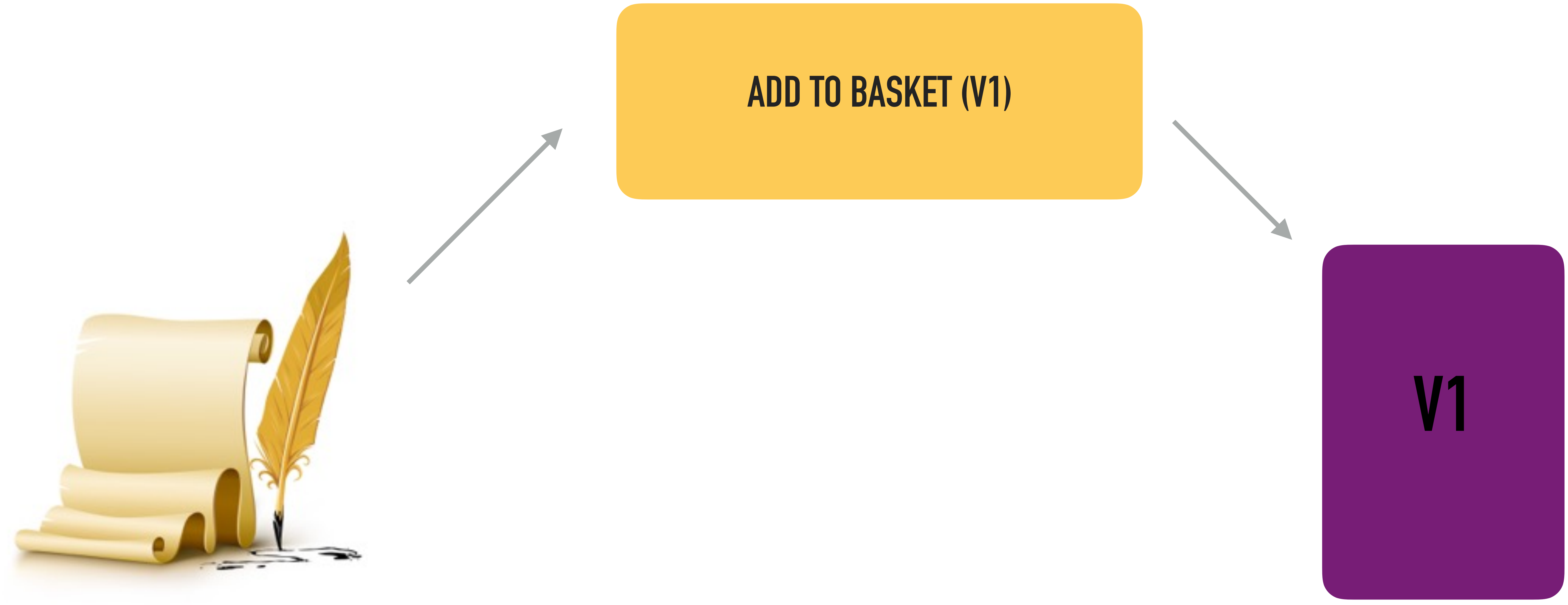
DECOUPLING



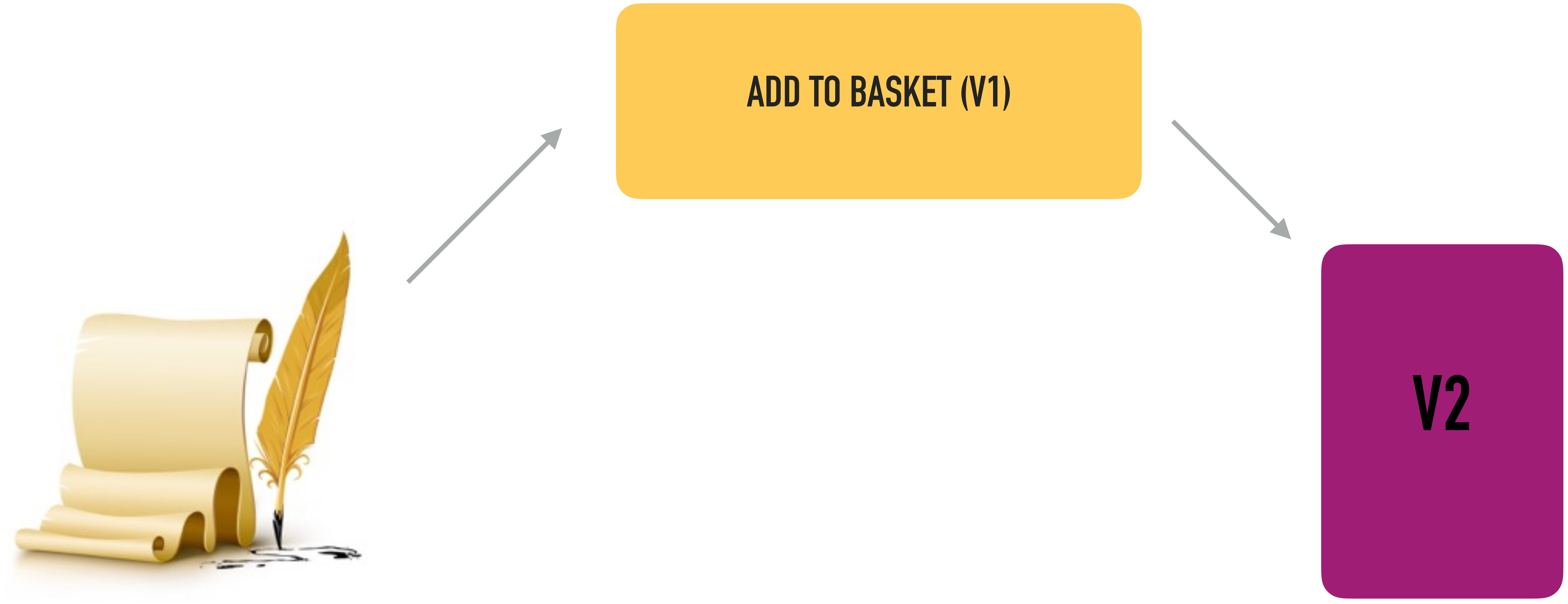
DECOUPLING



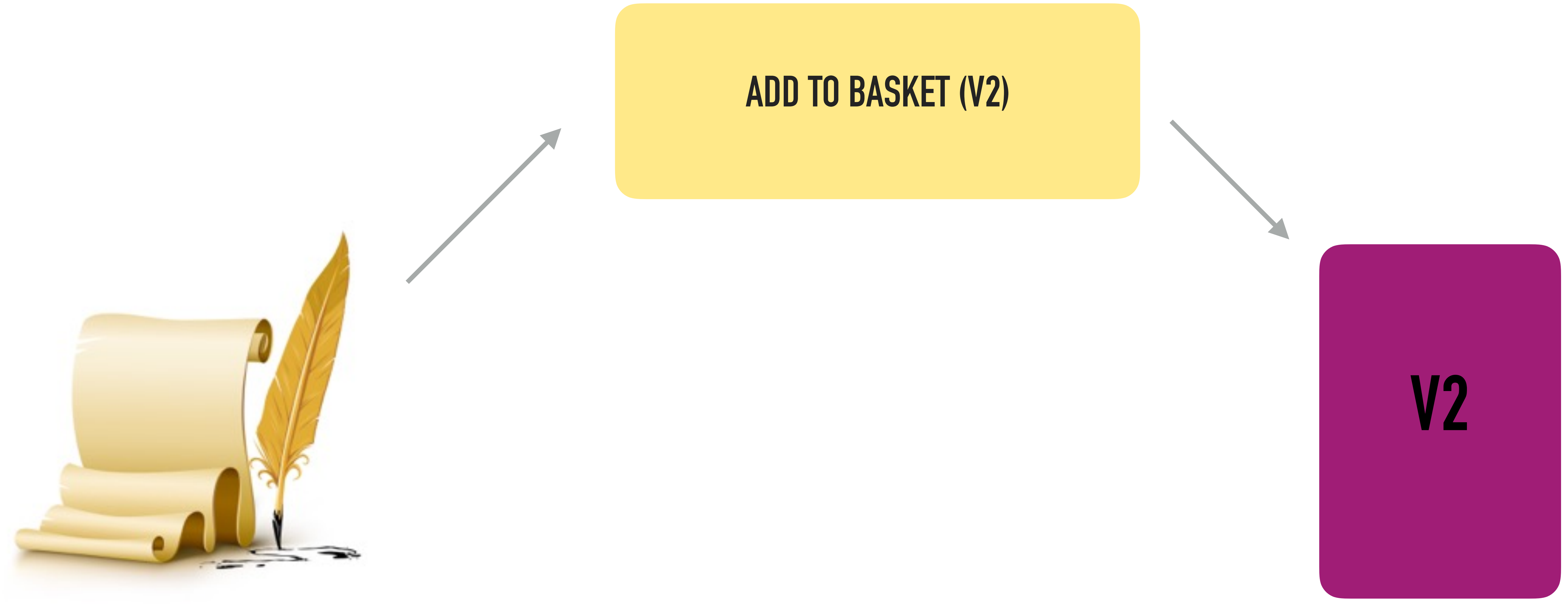
DECOUPLING



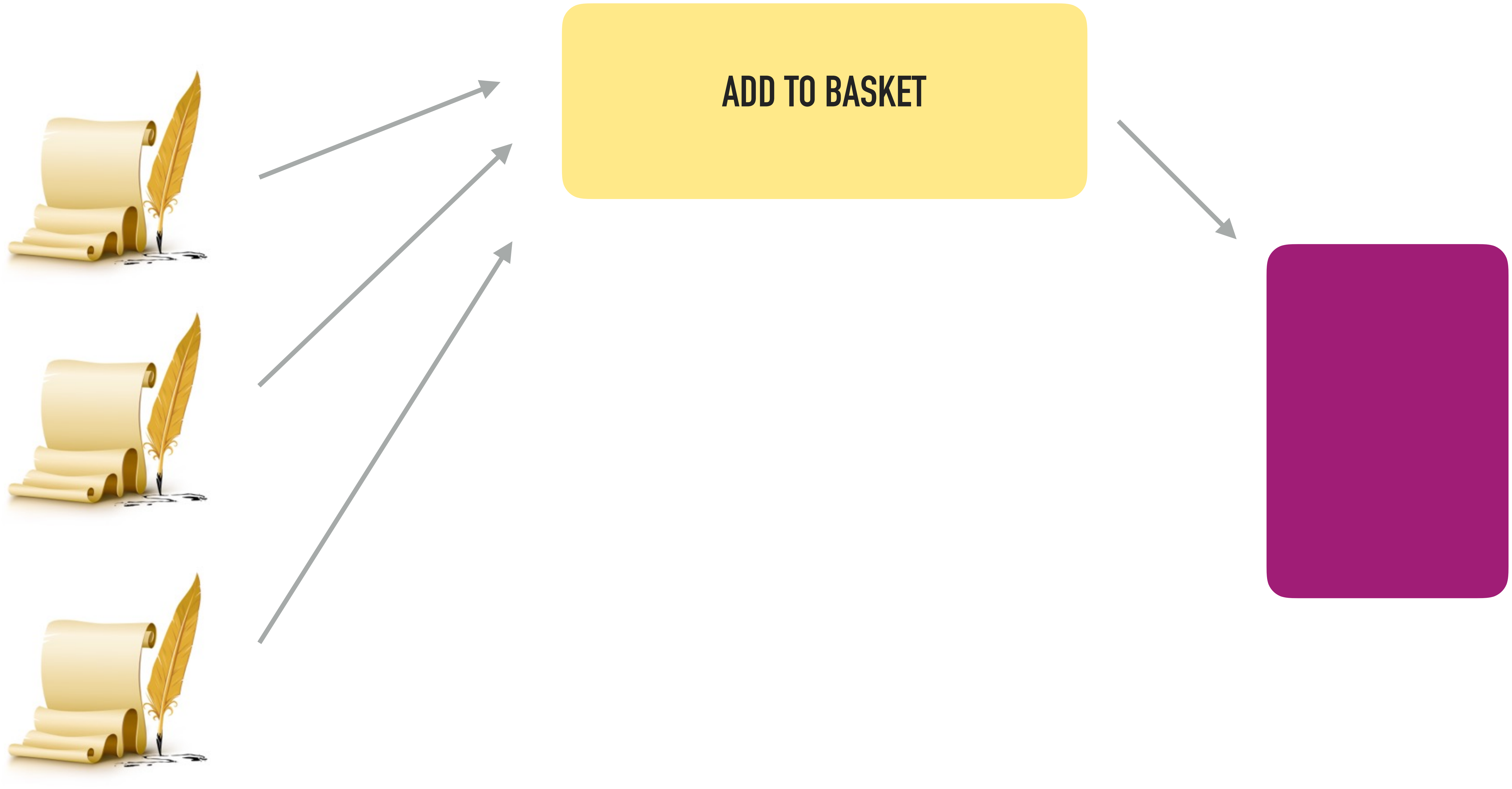
DECOUPLING



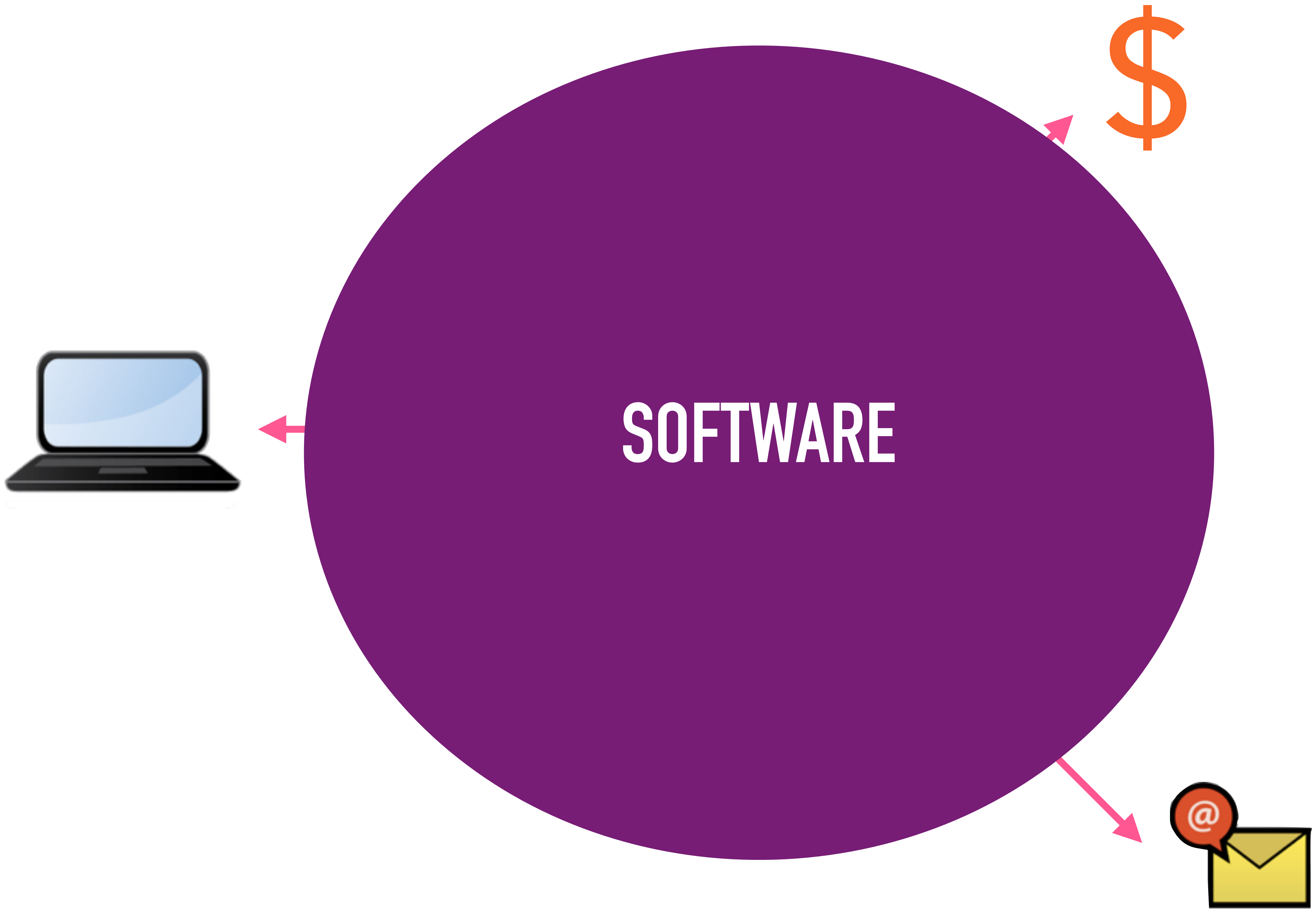
DECOUPLING



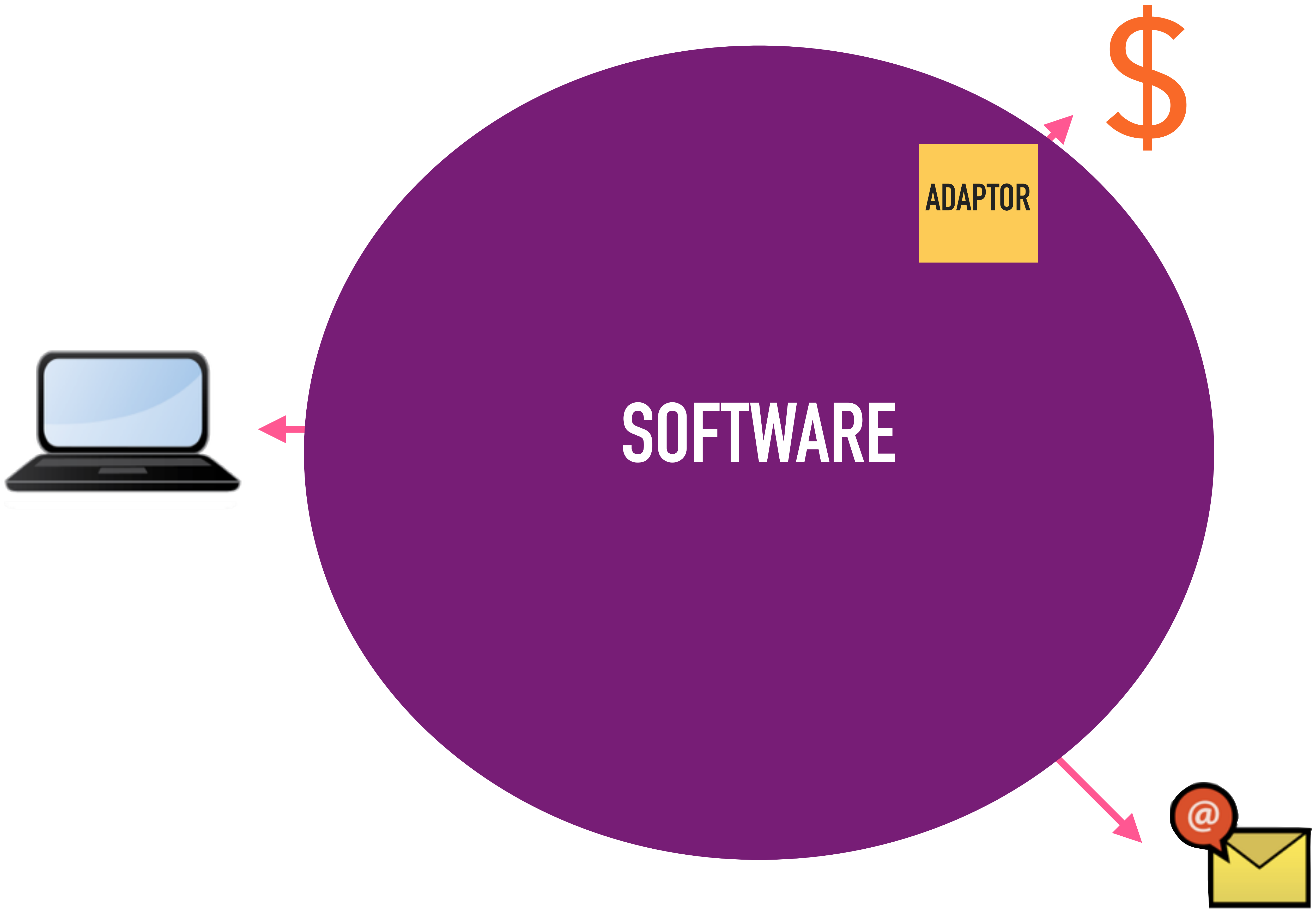
DECOUPLING



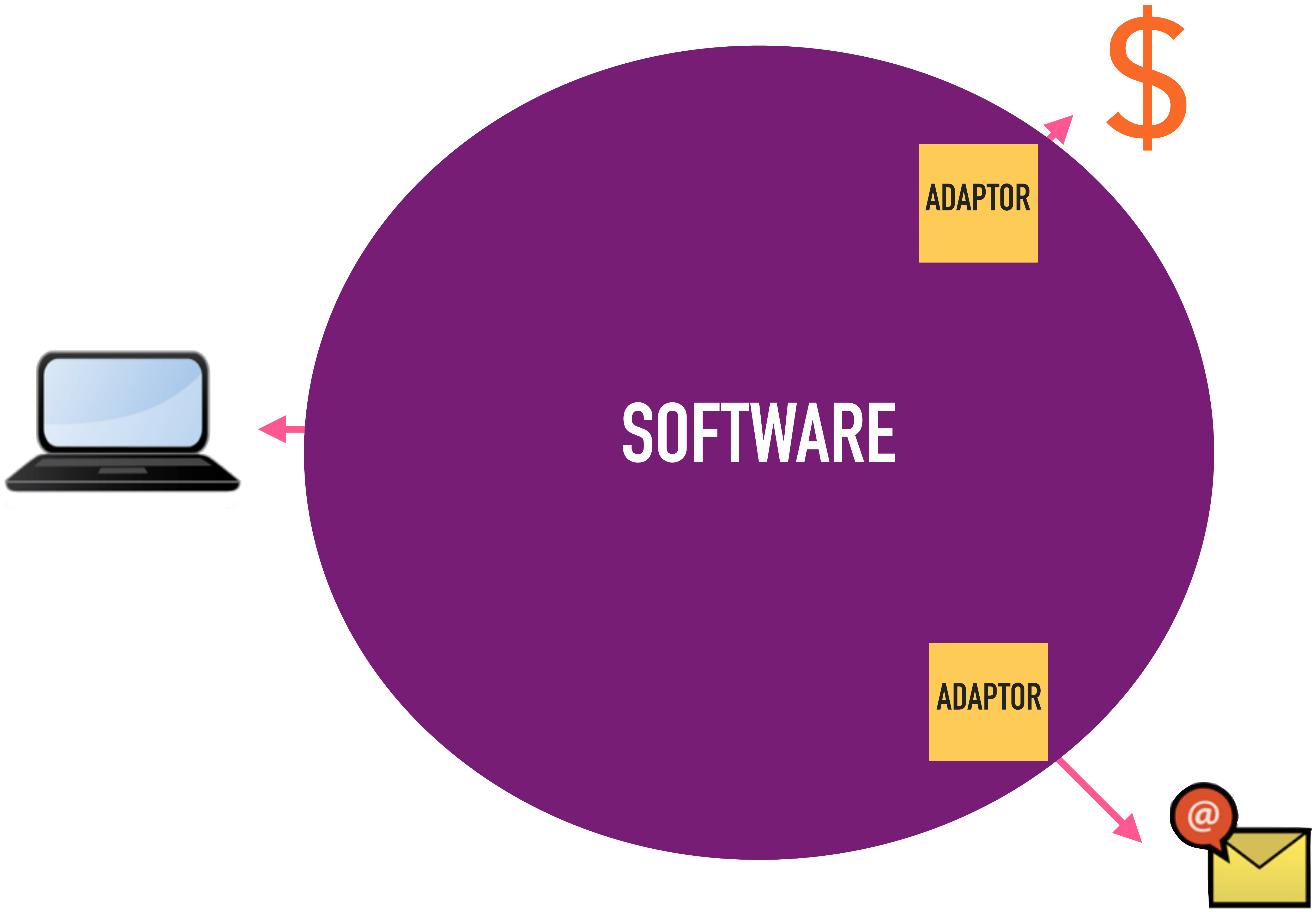
DECOUPLING



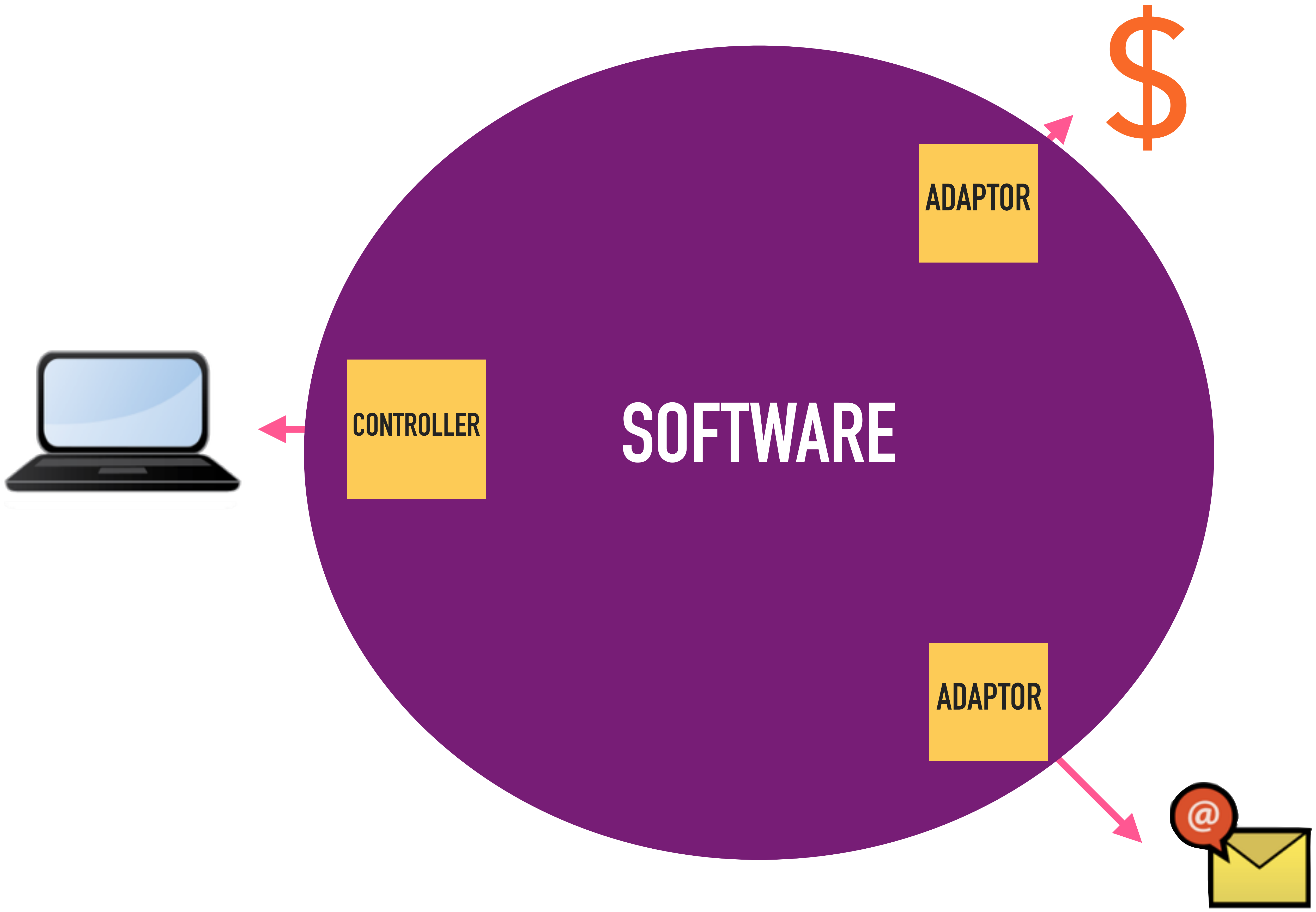
DECOUPLING



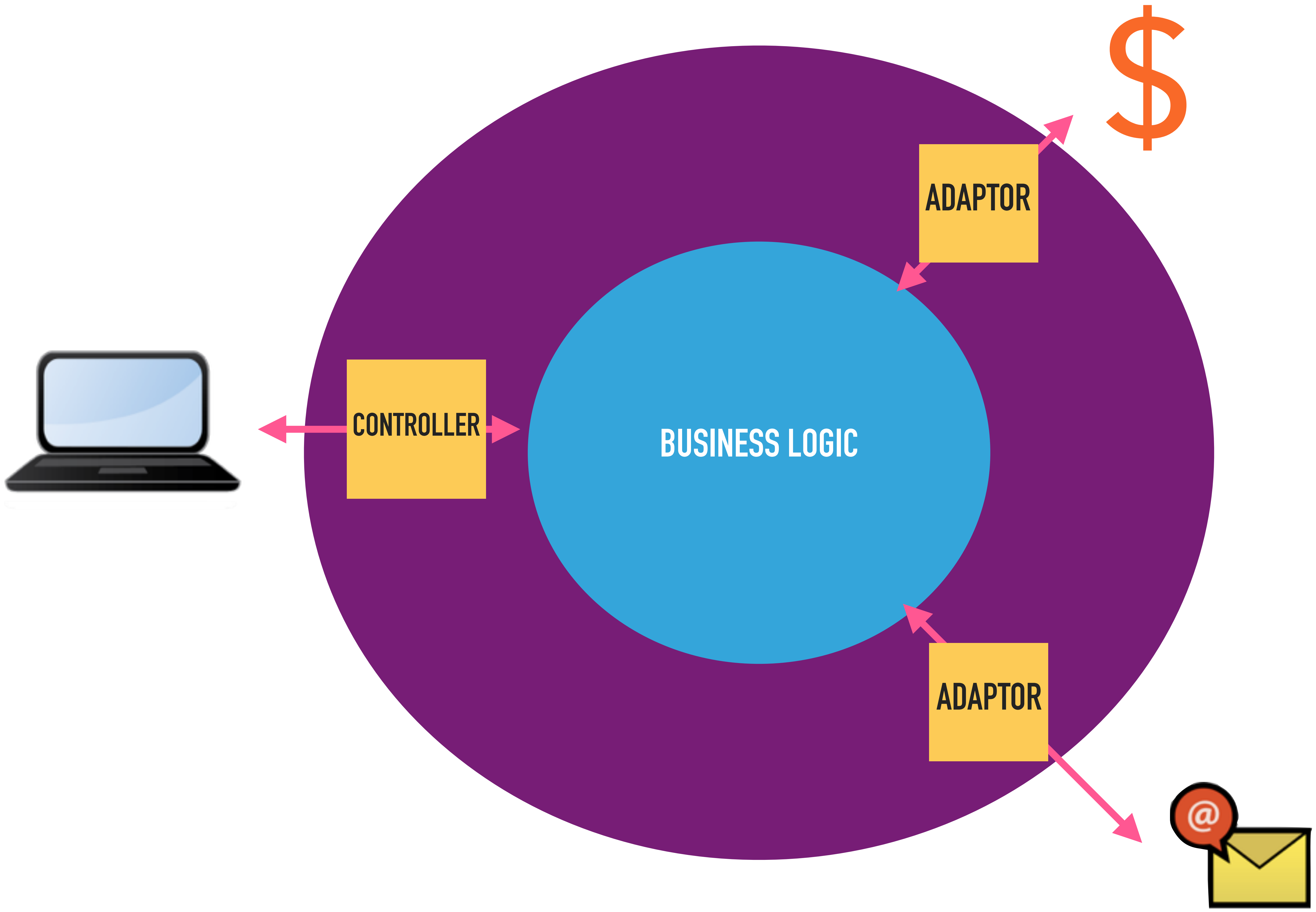
DECOUPLING



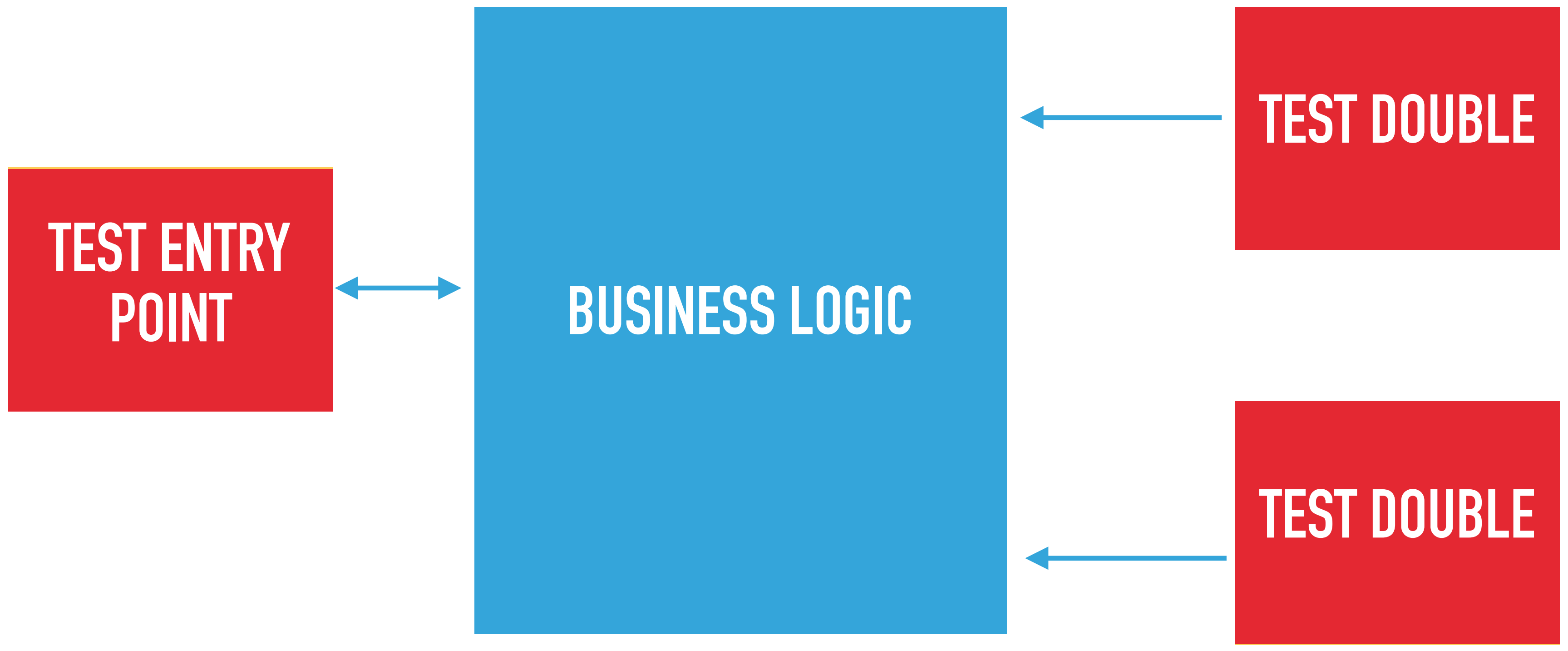
DECOUPLING



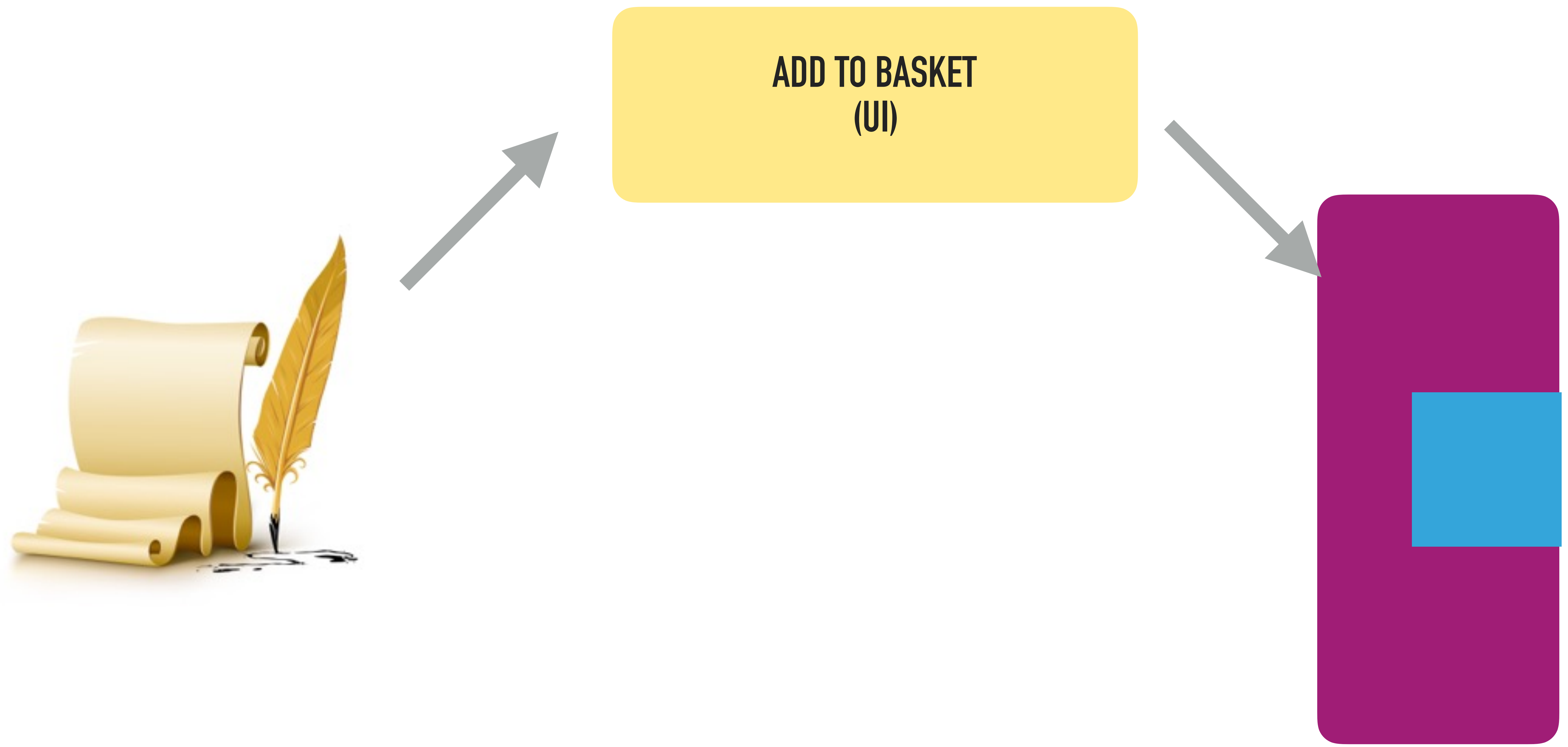
DECOUPLING



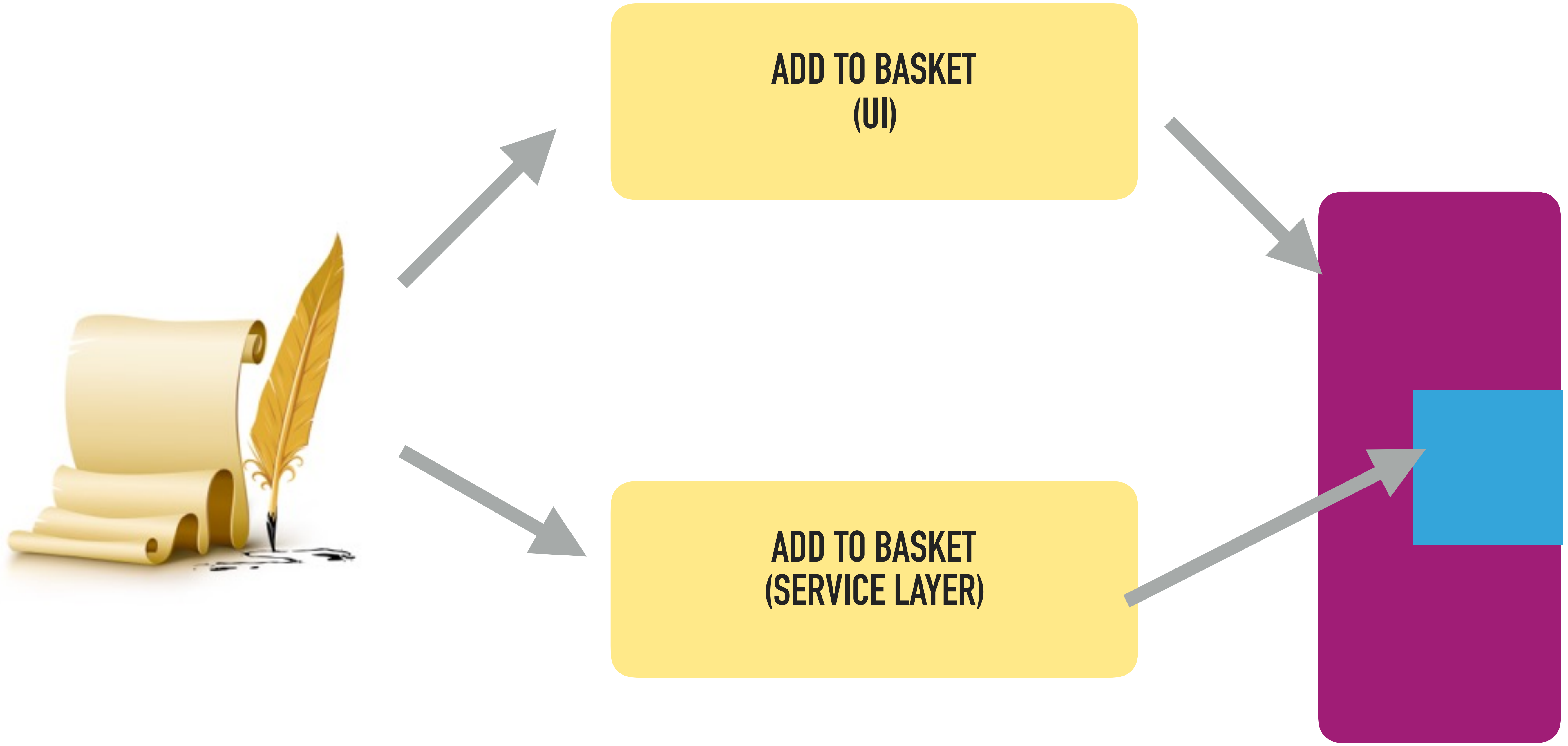
DECOUPLING



DECOUPLING



DECOUPLING



ADD TO BASKET (AT UI LAYER)

```
function addToBasket(string $productName): Basket
{

 … Lots of complicated, fragile code …

}
```

ADD TO BASKET (AT SERVICE LAYER)

```
function addToBasket(string $productName): Basket
{
 $productService = $container->productService();
 $product = $productService->lookup($productName);

 $basketService = $container->basketService();
 $basket = $basketService->newBasket();

 $basket->addProduct($product);

 return $basket;
}
```

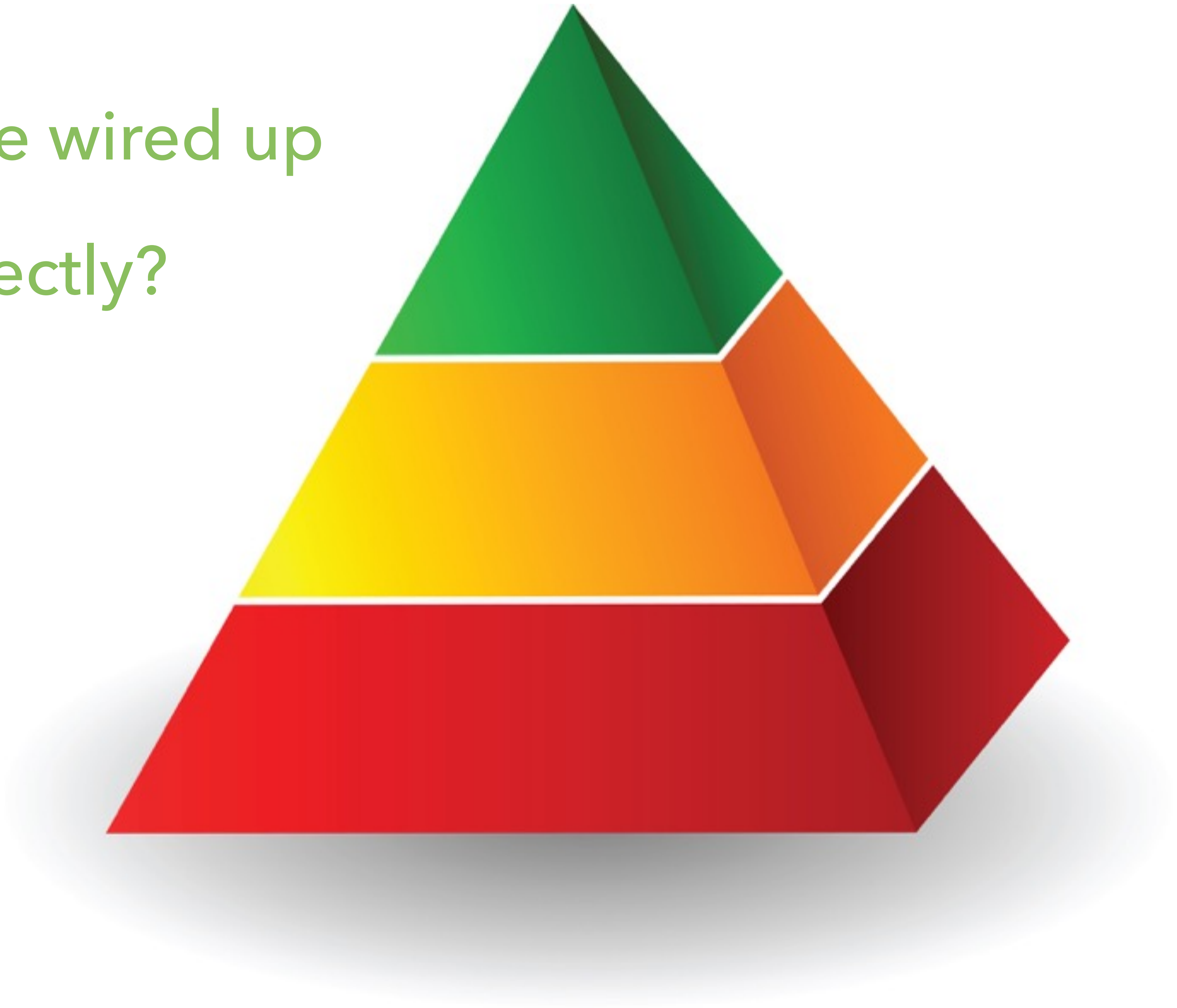
DO THE RIGHT KIND OF TESTS AT THE RIGHT LEVEL

Business logic

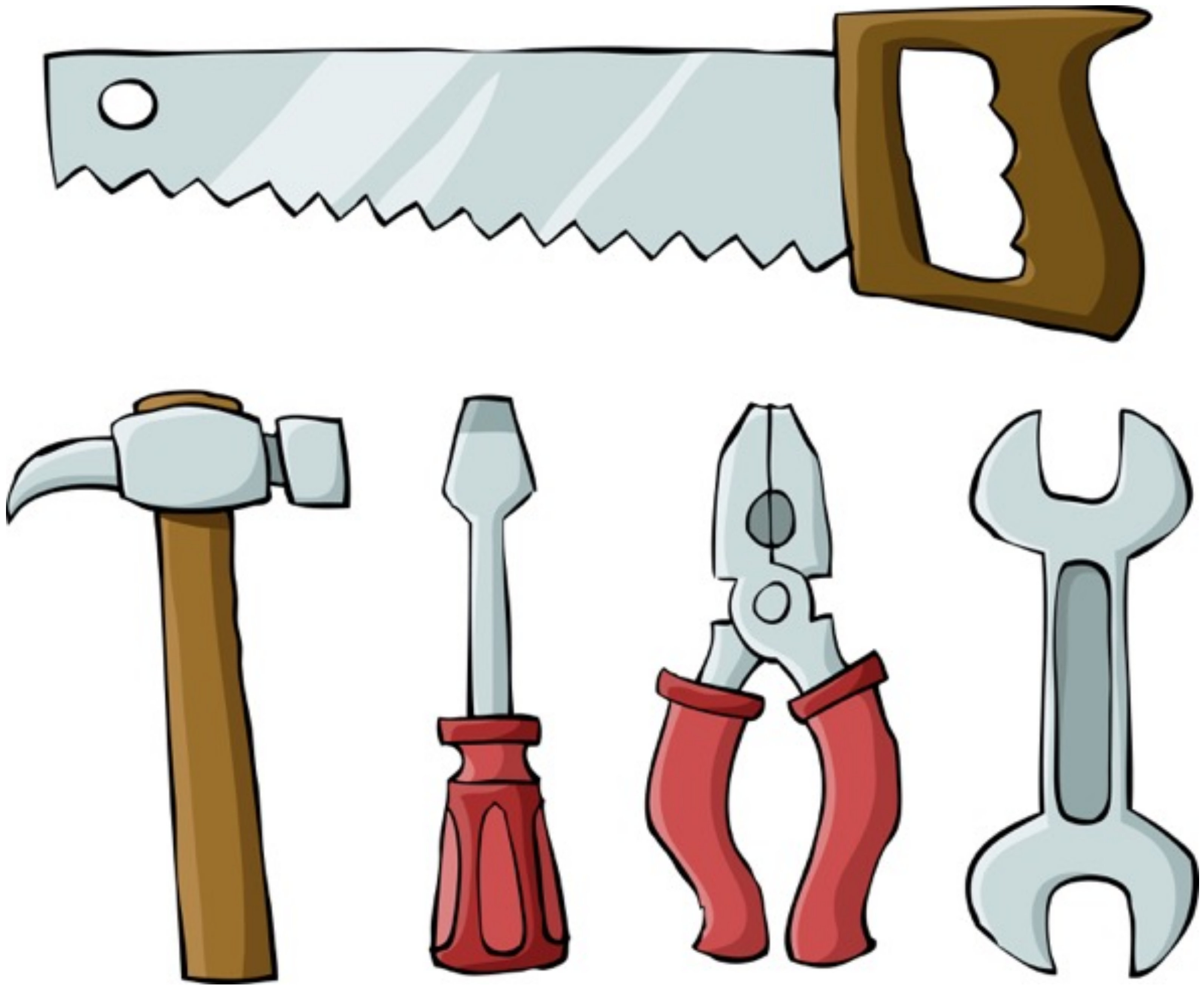
DO THE RIGHT KIND OF TESTS AT THE RIGHT LEVEL

Have we wired up

UI correctly?



DECOUPLING



TAKE AWAY

TAKE AWAY

- Do the right kind of tests at the right levels:
  - Business logic tested at the service layer
  - Test UI to make sure it's wired to business logic correctly

TAKE AWAY

- Do the right kind of tests at the right levels:
  - Business logic tested at the service layer
  - Test UI to make sure it’s wired to business logic correctly
- Architect your code well

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

AGENDA

- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests
- Building objects
- Tips
- Summary

UNIT TESTS

Unit

UNIT TEST EXAMPLE – SOFTWARE UNDER TEST

```
class PasswordValidator
{

   /**
    * Returns true if password meets following criteria:
    *
    * - 8 or more characters
    * - at least 1 digit
    * - at least 1 upper case letter
    * - at least 1 lower case letter
    */
   public function isValid(string $password) : bool
```

UNIT TEST EXAMPLE – TEST CASES REQUIRED

UNIT TEST EXAMPLE – TEST CASES REQUIRED

- Valid passwords:
    - “Passw0rd”

UNIT TEST EXAMPLE - TEST CASES REQUIRED

- Valid passwords:
  - “Passw0rd”
- Invalid passwords:
  - “Passw0r” - too short (everything else is good)
  - “Password” - no digit
  - “passw0rd” - no upper case letters
  - “PASSW0RD” - no lower case letters

NEW REQUIREMENT

```
class PasswordValidator
{

  /**
   * Returns true if password meets following criteria:
   *
   * - 8 or more characters
   * - at least 1 digit
   * - at least 1 upper case letter
   * - at least 1 lower case letter
   * - not one of the user's previous 5 passwords
   */
   public function isValid(string $password, User $user) : bool
```

UNIT TESTS

EXISTING PASSWORD VALIDATION RULES

UNIT TESTS

EXISTING PASSWORD VALIDATION RULES

CHECK IF LAST 5 PASSWORDS

UNIT TESTS

EXISTING PASSWORD VALIDATION RULES

CHECK IF LAST 5 PASSWORDS

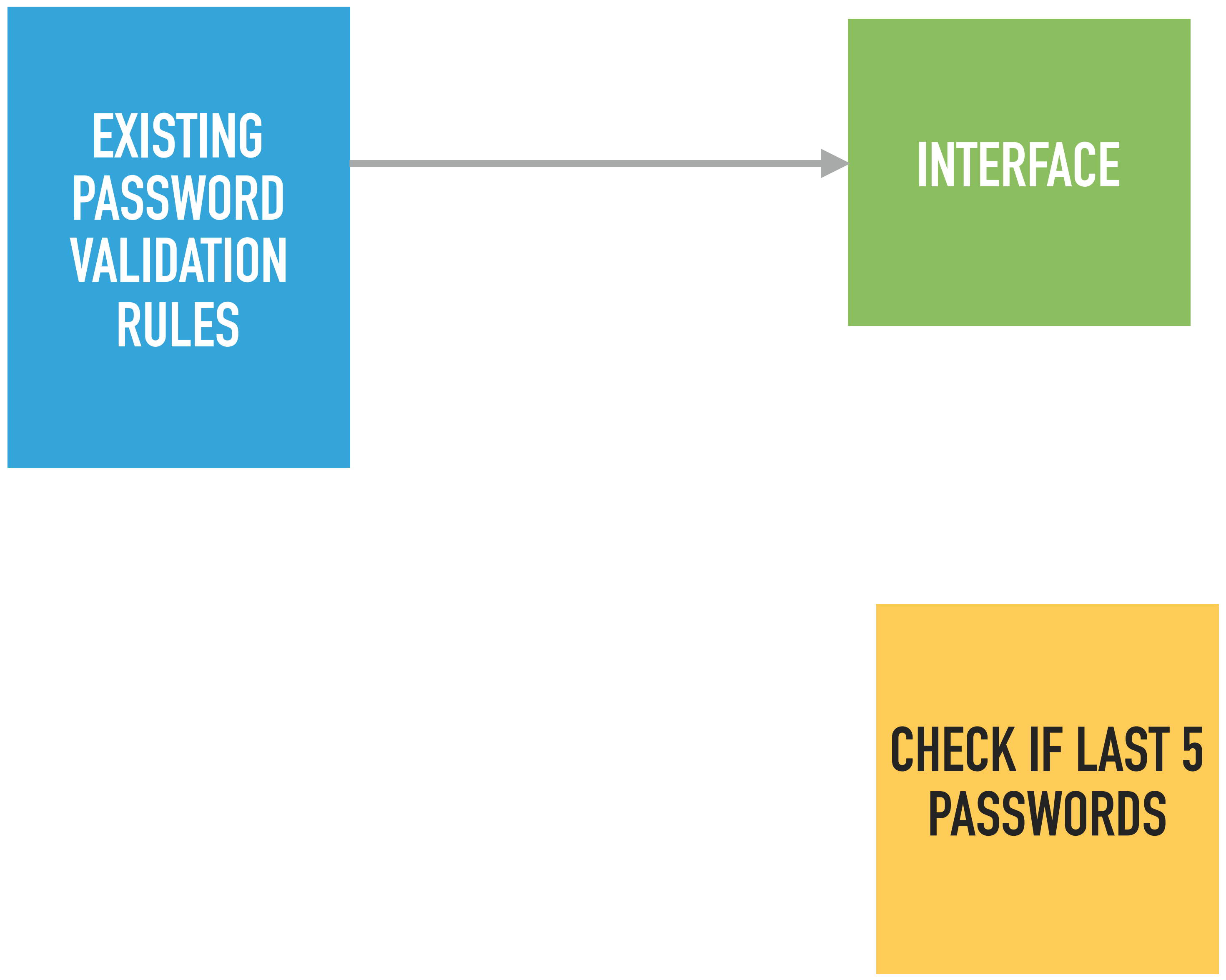
UNIT TESTS

EXISTING PASSWORD VALIDATION RULES

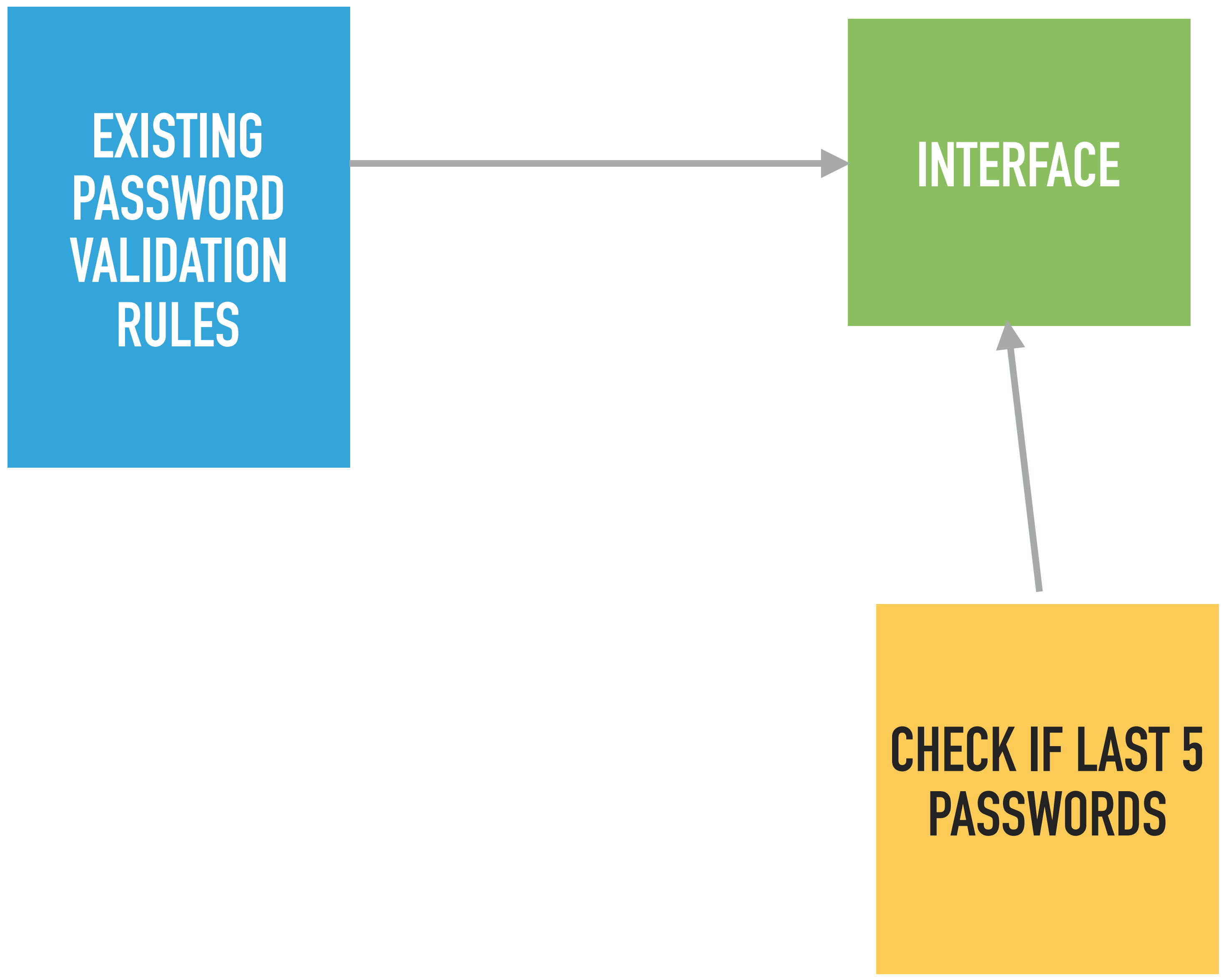
INTERFACE

CHECK IF LAST 5 PASSWORDS

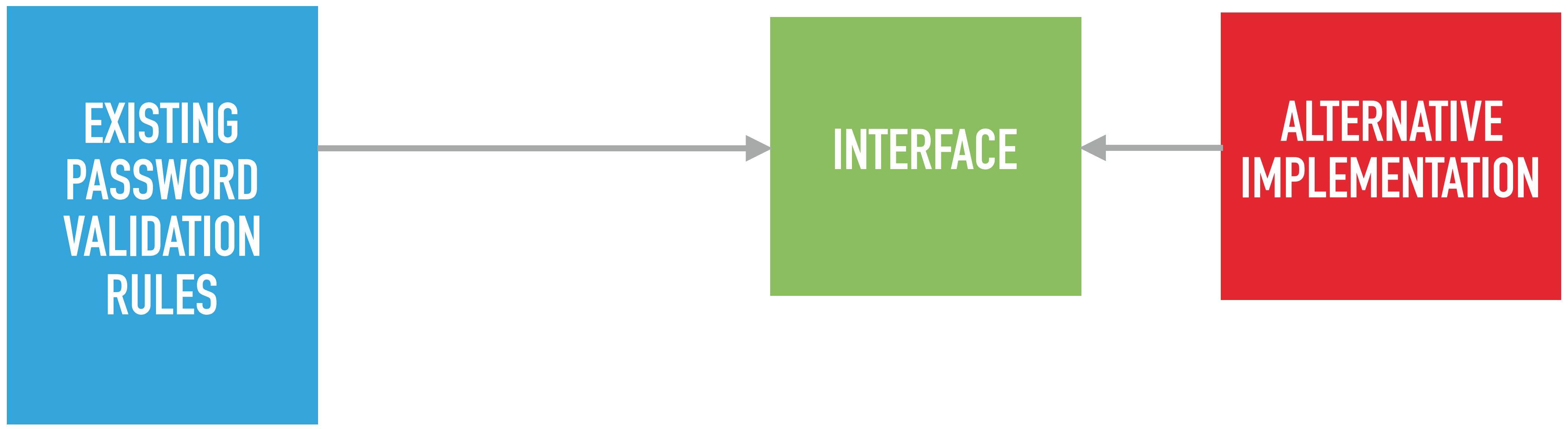
UNIT TESTS


EXISTING PASSWORD VALIDATION RULES
INTERFACE
CHECK IF LAST 5 PASSWORDS

UNIT TESTS


EXISTING PASSWORD VALIDATION RULES
INTERFACE
CHECK IF LAST 5 PASSWORDS

UNIT TESTS


EXISTING PASSWORD VALIDATION RULES
INTERFACE
ALTERNATIVE IMPLEMENTATION

PREVIOUS PASSWORD CHECKER INTERFACE

```
interface PreviousPasswordChecker
{

  /**
   * Returns true if password has been used by user
   * in previous 5 passwords
   */
   public function isPreviouslyUsed(
        string $password,
        User $user
   ): bool;

}
```

PASSWORD VALIDATOR TEST REVISITED

PASSWORD VALIDATOR TEST REVISITED

- Update existing tests to account for:
    - Any calls to RecentPasswordChecker

PASSWORD VALIDATOR TEST REVISITED

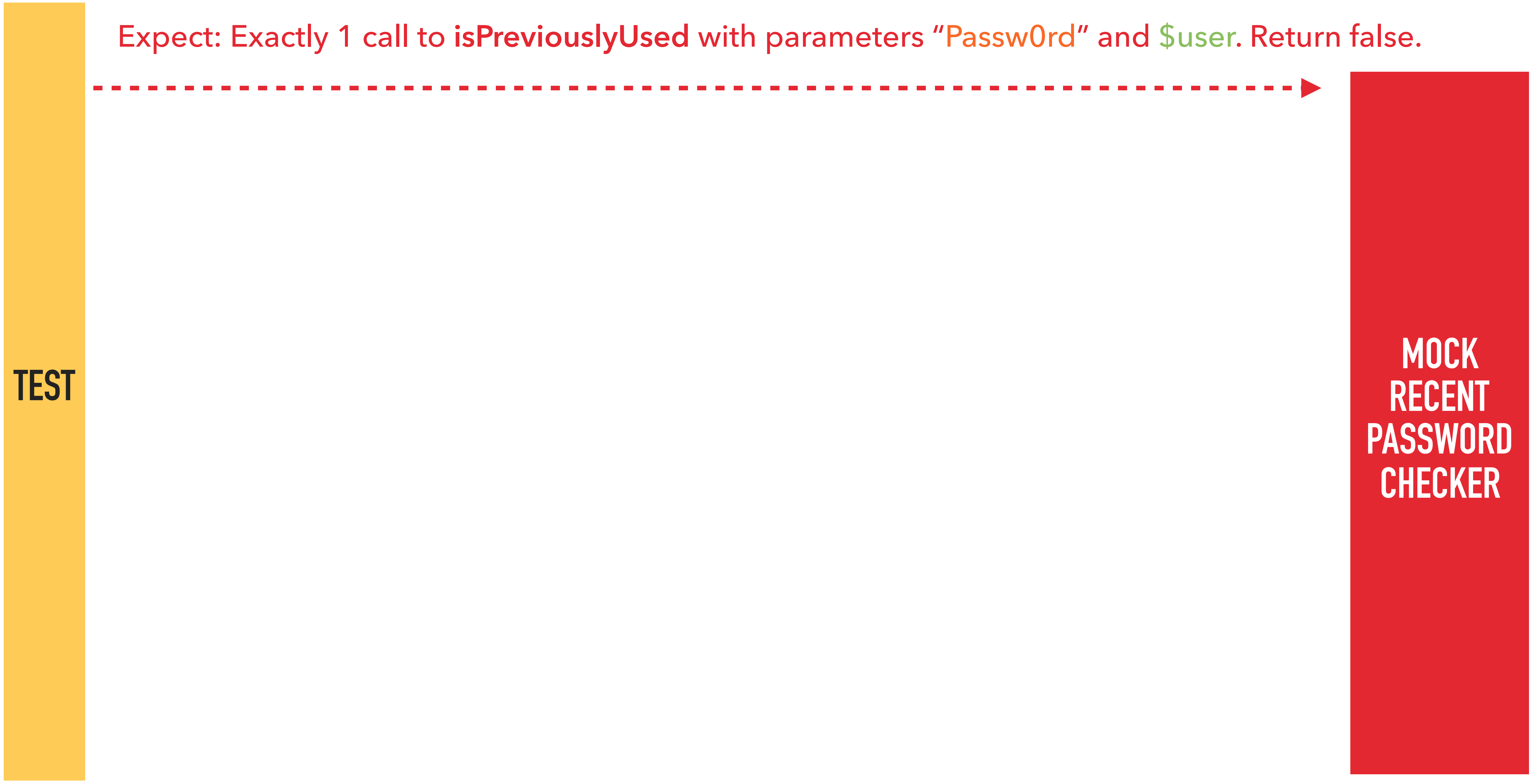
- Update existing tests to account for:
  - Any calls to RecentPasswordChecker
- New tests
  - Valid password. Has been recently used
  - Valid password. Has NOT been recently used

NEW TEST: VALID PASSWORD, NOT RECENTLY USED

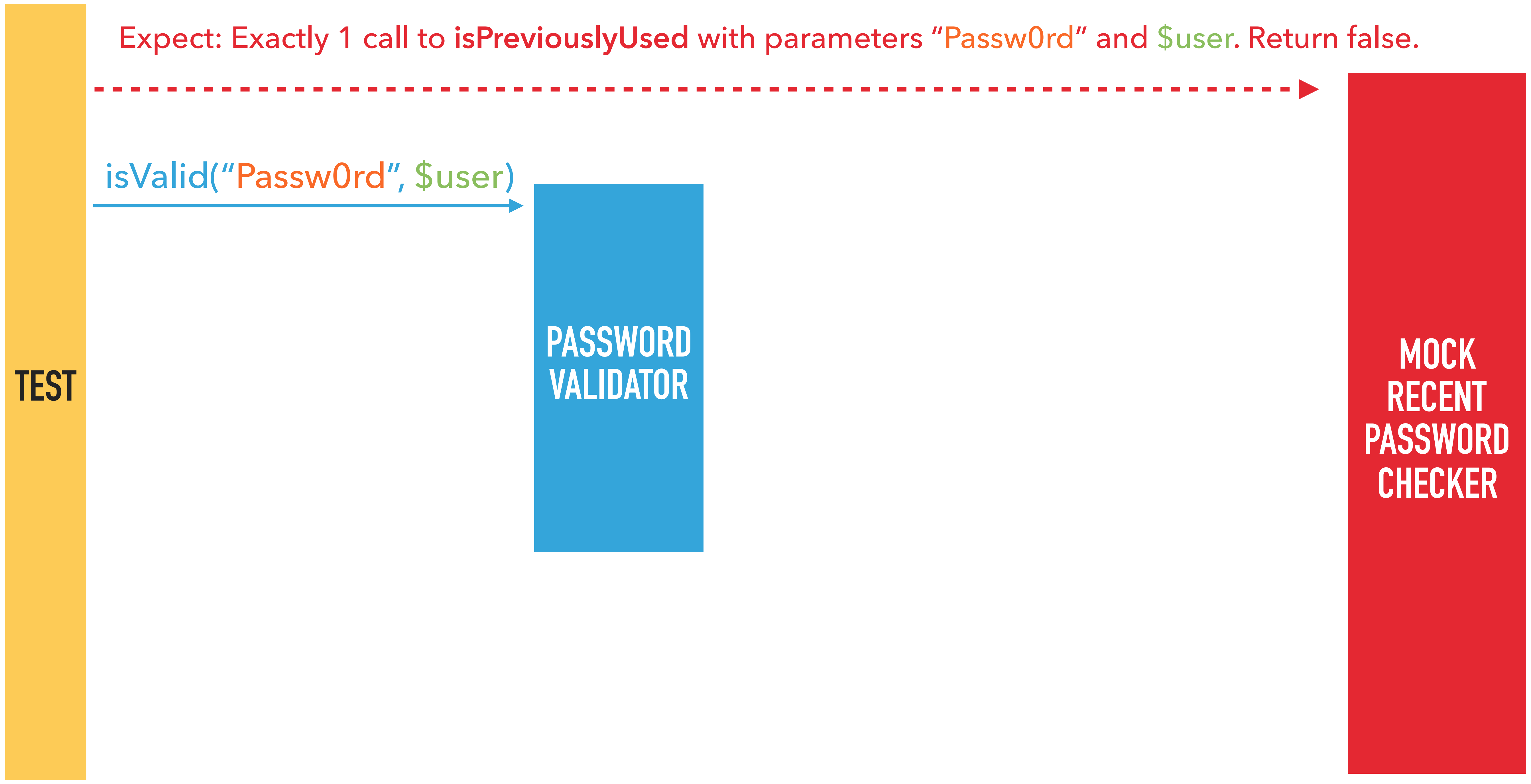
NEW TEST: VALID PASSWORD, NOT RECENTLY USED

TEST

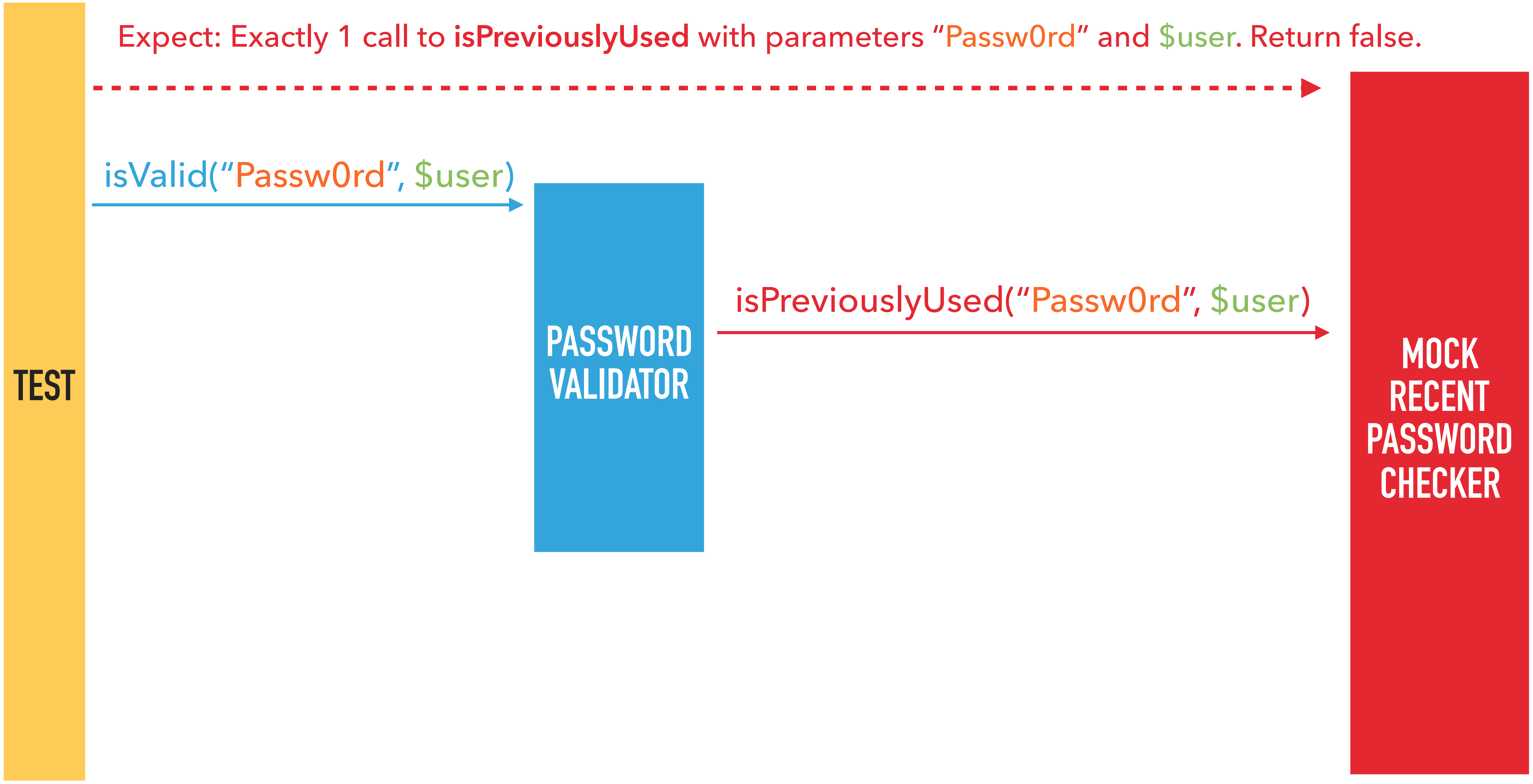
NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters "Passw0rd" and $user. Return false.

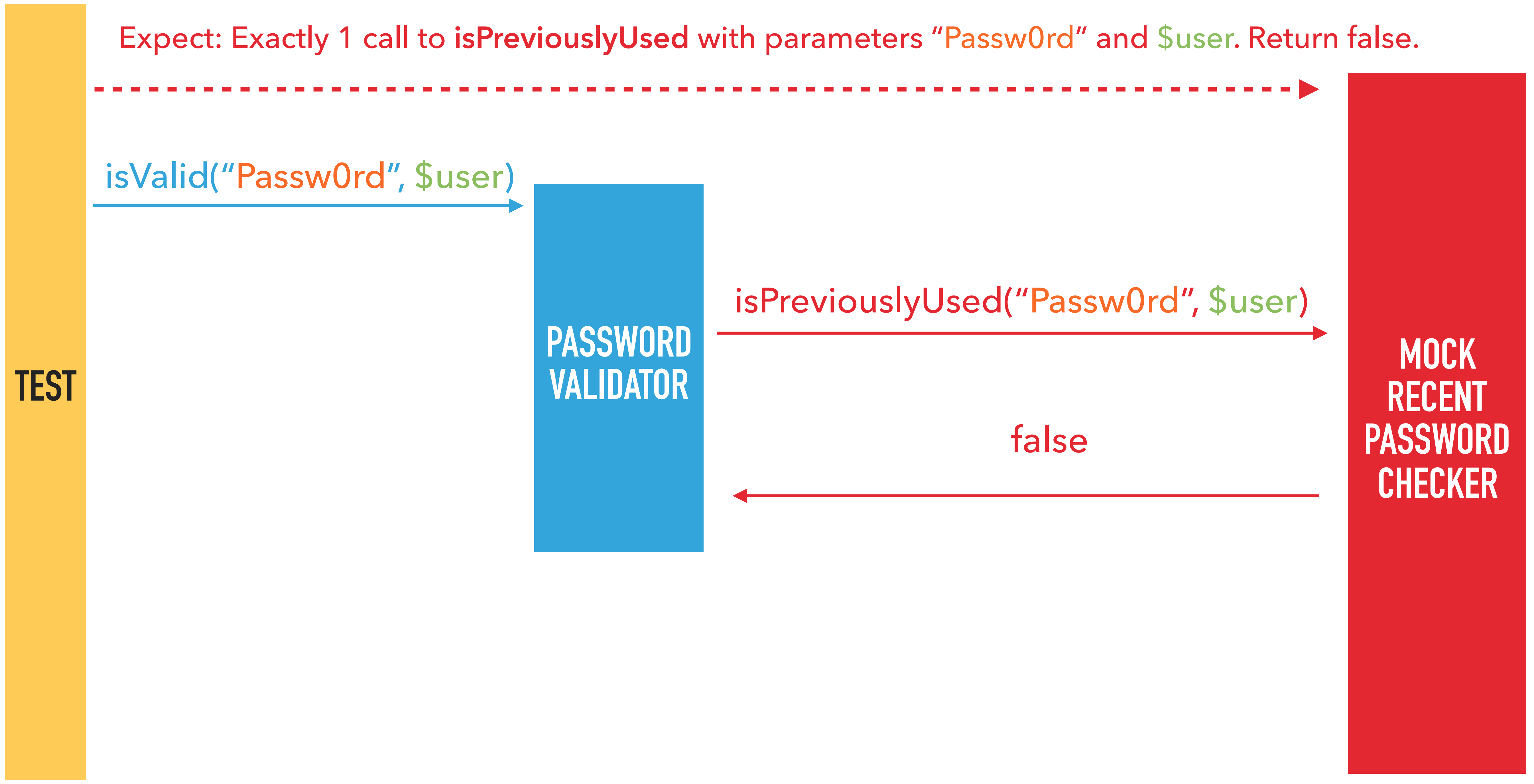
NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters “Passw0rd” and $user. Return false.
isValid(“Passw0rd”, $user)
TEST
PASSWORD VALIDATOR
MOCK RECENT PASSWORD CHECKER

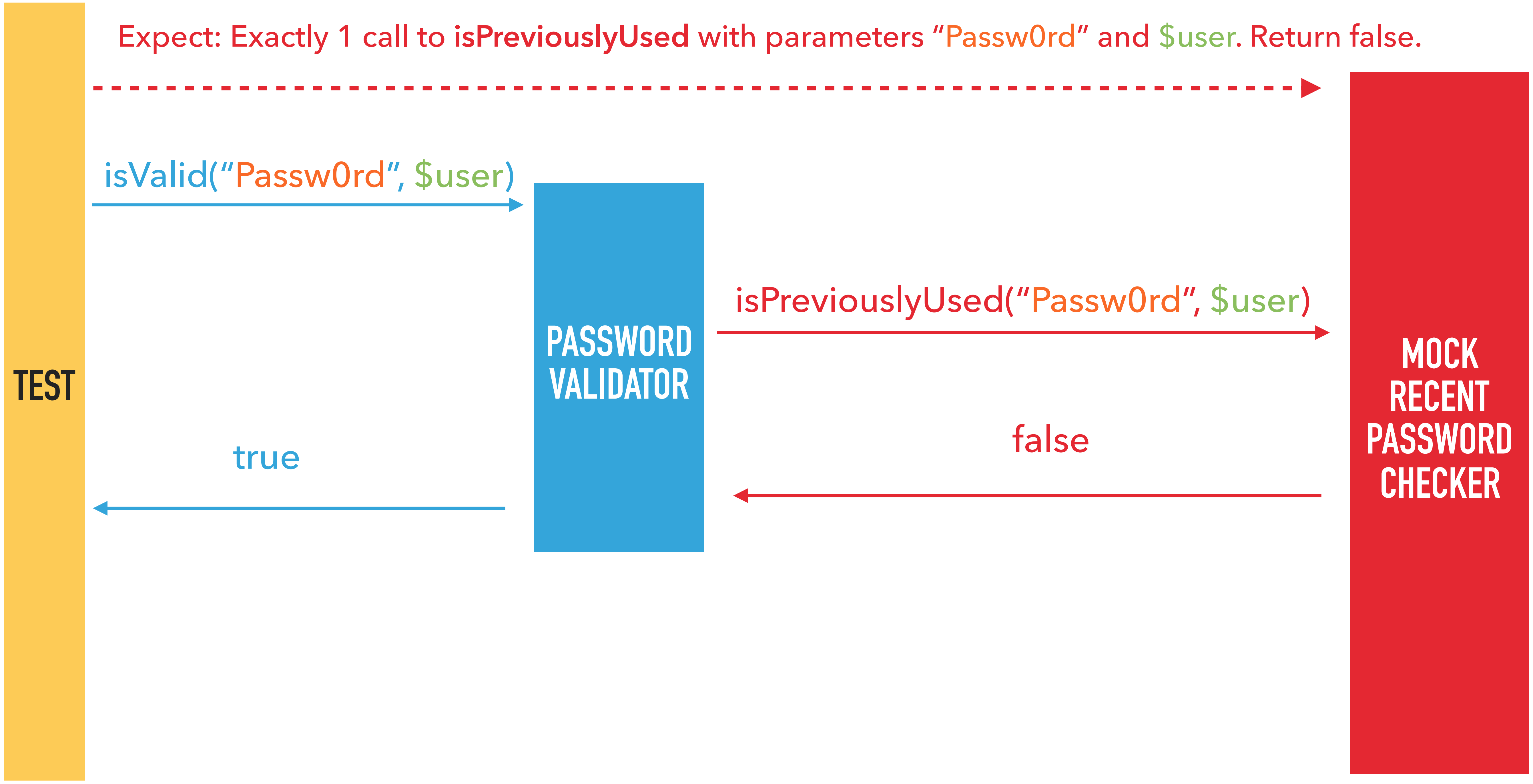
NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to isPreviouslyUsed with parameters "Passw0rd" and $user. Return false.
isValid("Passw0rd", $user)
isPreviouslyUsed("Passw0rd", $user)
TEST
PASSWORD VALIDATOR
MOCK RECENT PASSWORD CHECKER

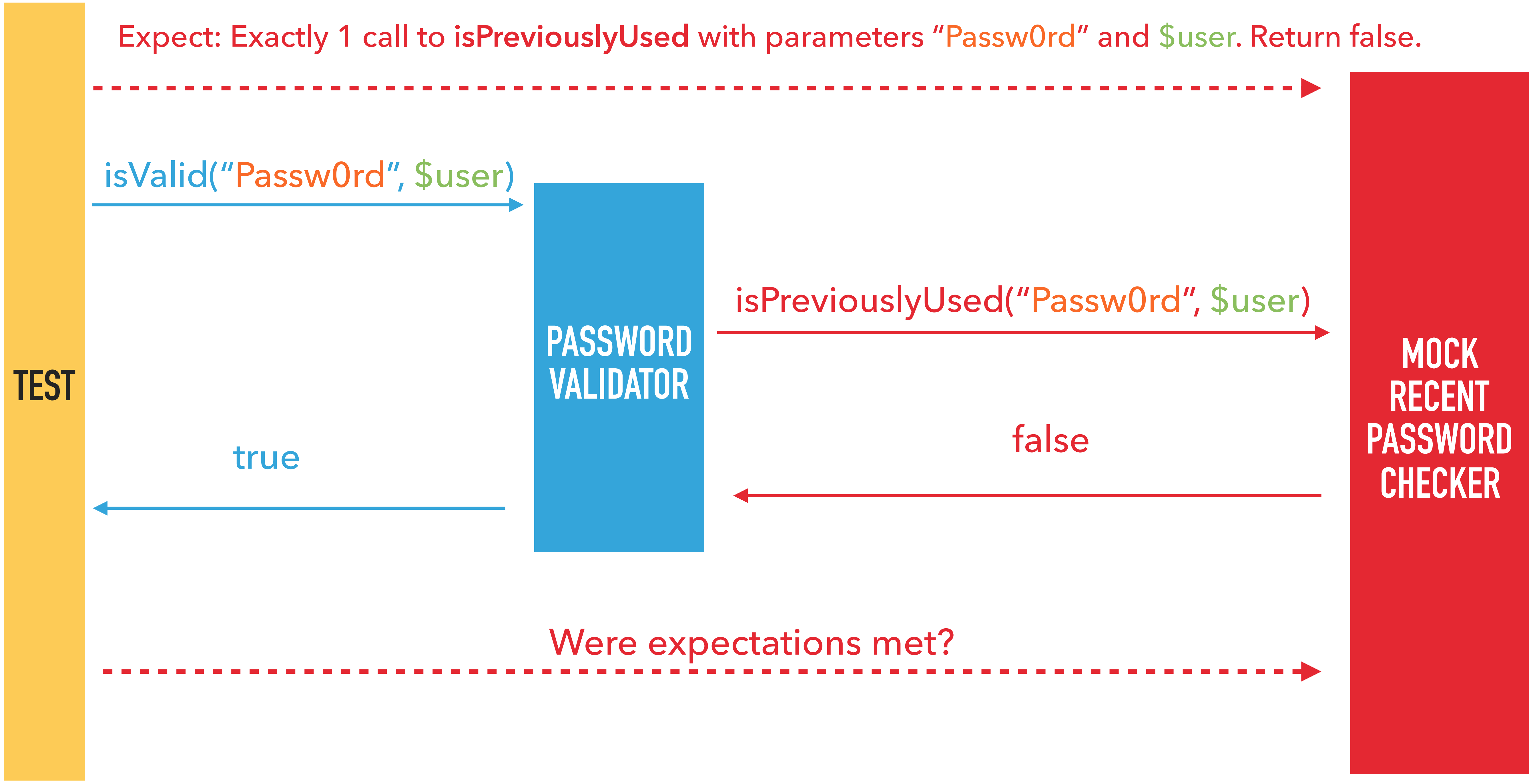
NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters “Passw0rd” and $user. Return false.
isValid(“Passw0rd”, $user)
TEST
PASSWORD VALIDATOR
isPreviouslyUsed(“Passw0rd”, $user)
false
MOCK RECENT PASSWORD CHECKER

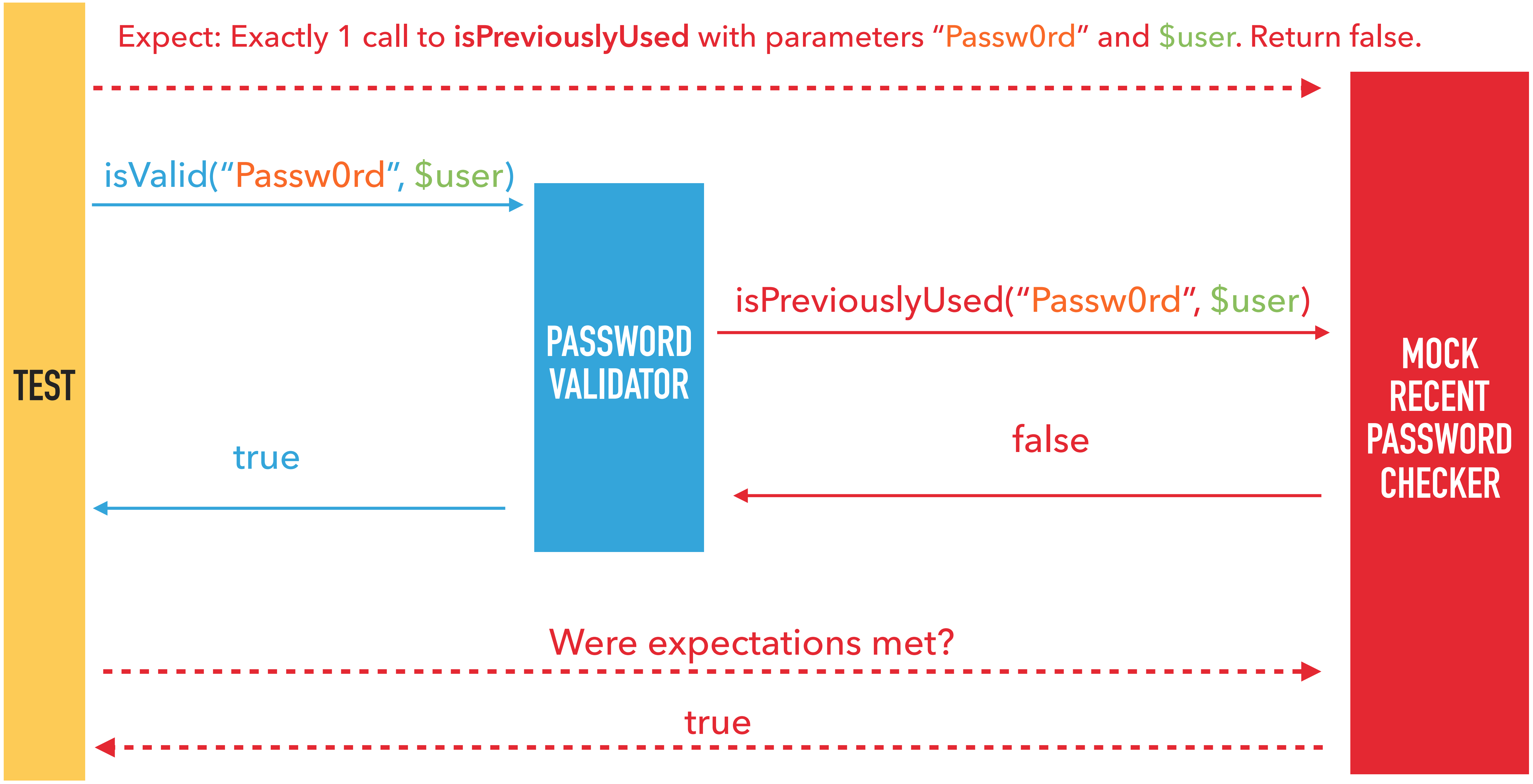
NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters "Passw0rd" and $user. Return false.
isValid("Passw0rd", $user)
TEST
PASSWORD VALIDATOR
isPreviouslyUsed("Passw0rd", $user)
false
true
MOCK RECENT PASSWORD CHECKER

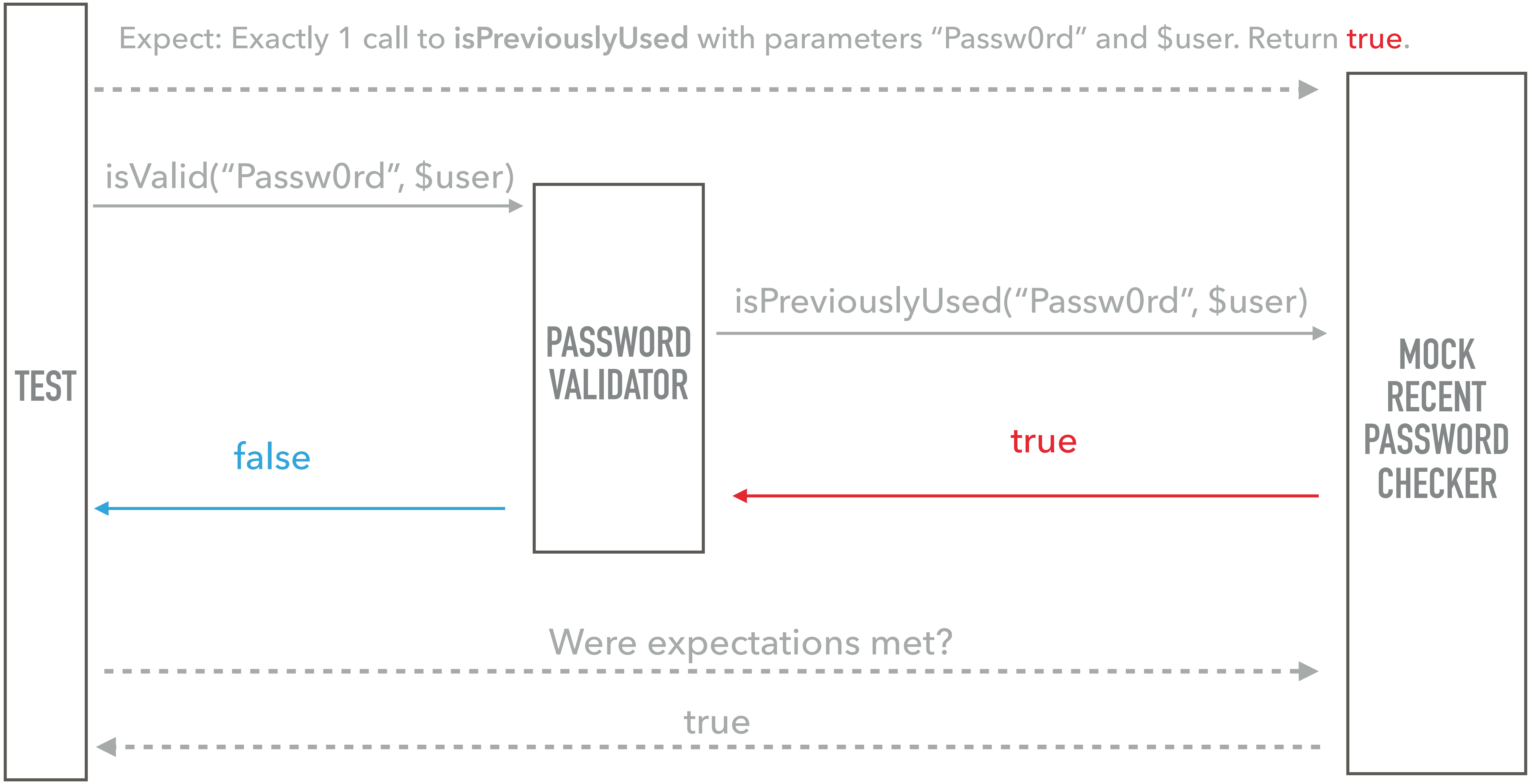
NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to isPreviouslyUsed with parameters "Passw0rd" and $user. Return false.
isValid("Passw0rd", $user)
TEST
PASSWORD VALIDATOR
isPreviouslyUsed("Passw0rd", $user)
false
true
MOCK RECENT PASSWORD CHECKER
Were expectations met?

NEW TEST: VALID PASSWORD, NOT RECENTLY USED


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters "Passw0rd" and $user. Return false.
isValid("Passw0rd", $user)
TEST
PASSWORD VALIDATOR
isPreviouslyUsed("Passw0rd", $user)
false
true
MOCK RECENT PASSWORD CHECKER
Were expectations met?
true

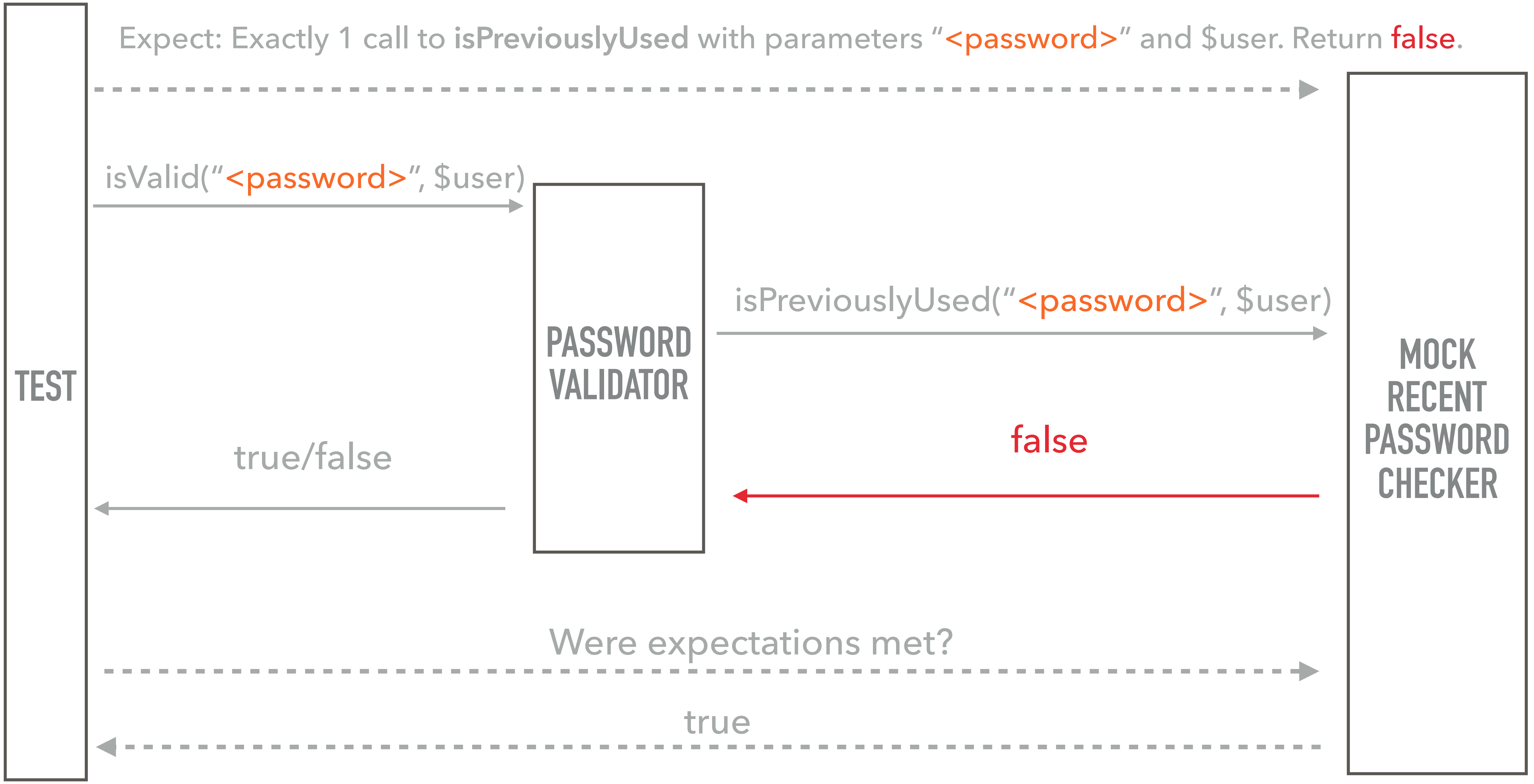
NEW TEST: VALID PASSWORD, BUT RECENTLY USED


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters “Passw0rd” and $user. Return true.
isValid(“Passw0rd”, $user)
TEST
PASSWORD VALIDATOR
isPreviouslyUsed(“Passw0rd”, $user)
MOCK RECENT PASSWORD CHECKER
true
false
Were expectations met?
true

EXISTING CODE

```
class PasswordValidator
{
  public function isValid(string $password, User $user) : bool
  {
     if ($this->recentPasswordChecker->isRecentPassword(
              $password, $user)) {
       return false;
     }

     if (… password too short …) return false;
     if (… password has no digit …) return false;

    … remaining checks …

    return true;
```

EXISTING TESTS


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters "<password>" and $user. Return false.
isValid("<password>", $user)
PASSWORD VALIDATOR
isPreviouslyUsed("<password>", $user)
false
true/false
TEST
MOCK RECENT PASSWORD CHECKER
Were expectations met?
true

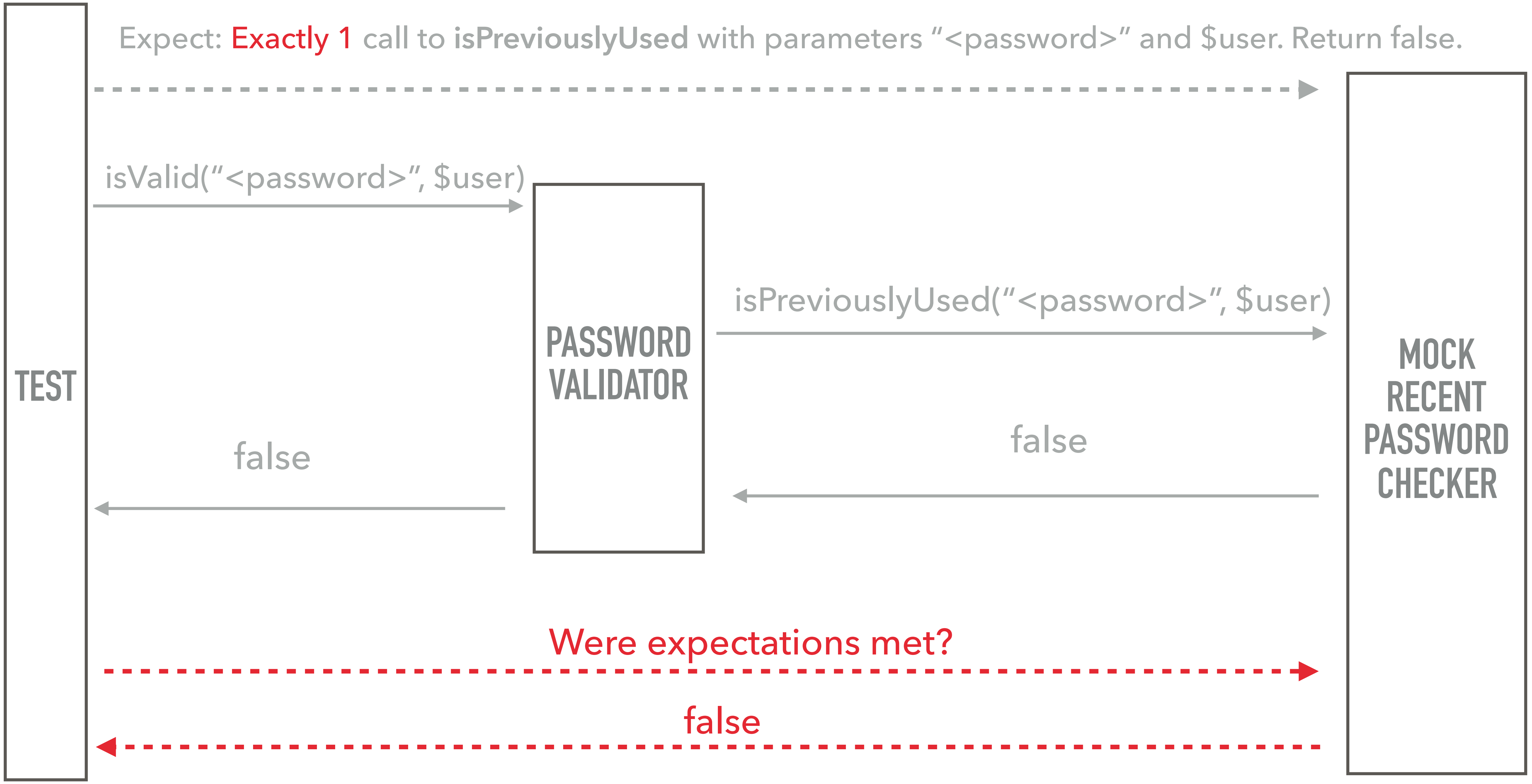
EXISTING CODE

```
class PasswordValidator
{
  public function isValid(string $password, User $user) : bool
  {
     if ($this->recentPasswordChecker->isRecentPassword(
              $password, $user)) {
       return false;
     }

     if (… password too short …) return false;
     if (… password has no digit …) return false;
    … remaining checks …

    return true;
```

EXISTING CODE (REFACTORED)

```
class PasswordValidator
{
  public function isValid(string $password, User $user) : bool
  {
     if (… password too short …) return false;
     if (… password has no digit …) return false;

     if ($this->recentPasswordChecker->isRecentPassword(
              $password, $user)) {
       return false;
     }

    … remaining checks …

    return true;
```

EXISTING TESTS: AFTER REFACTOR


Expect: Exactly 1 call to **isPreviouslyUsed** with parameters "<password>" and $user. Return false.
isValid("<password>", $user)
isPreviouslyUsed("<password>", $user)
false
false
TEST
PASSWORD VALIDATOR
MOCK RECENT PASSWORD CHECKER
Were expectations met?
false

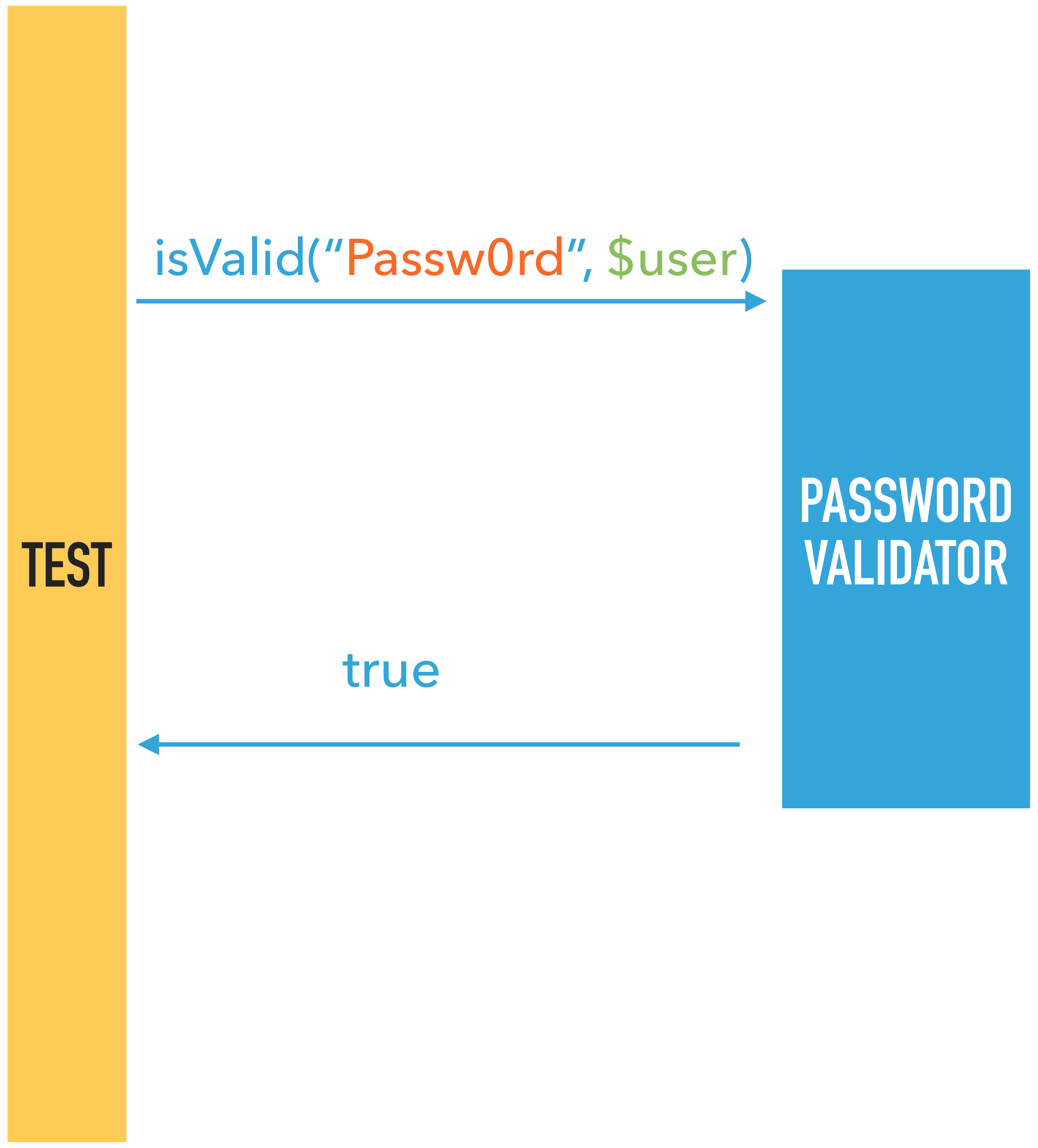
EXISTING CODE (REFACTORED)

```
class PasswordValidator
{
  public function isValid(string $password, User $user) : bool
  {
     if (… password too short …) return false;
     if (… password has no digit …) return false;

     if ($this->recentPasswordChecker->isRecentPassword(
              $password, $user)) {
       return false;
     }

    … remaining checks …

    return true;
```

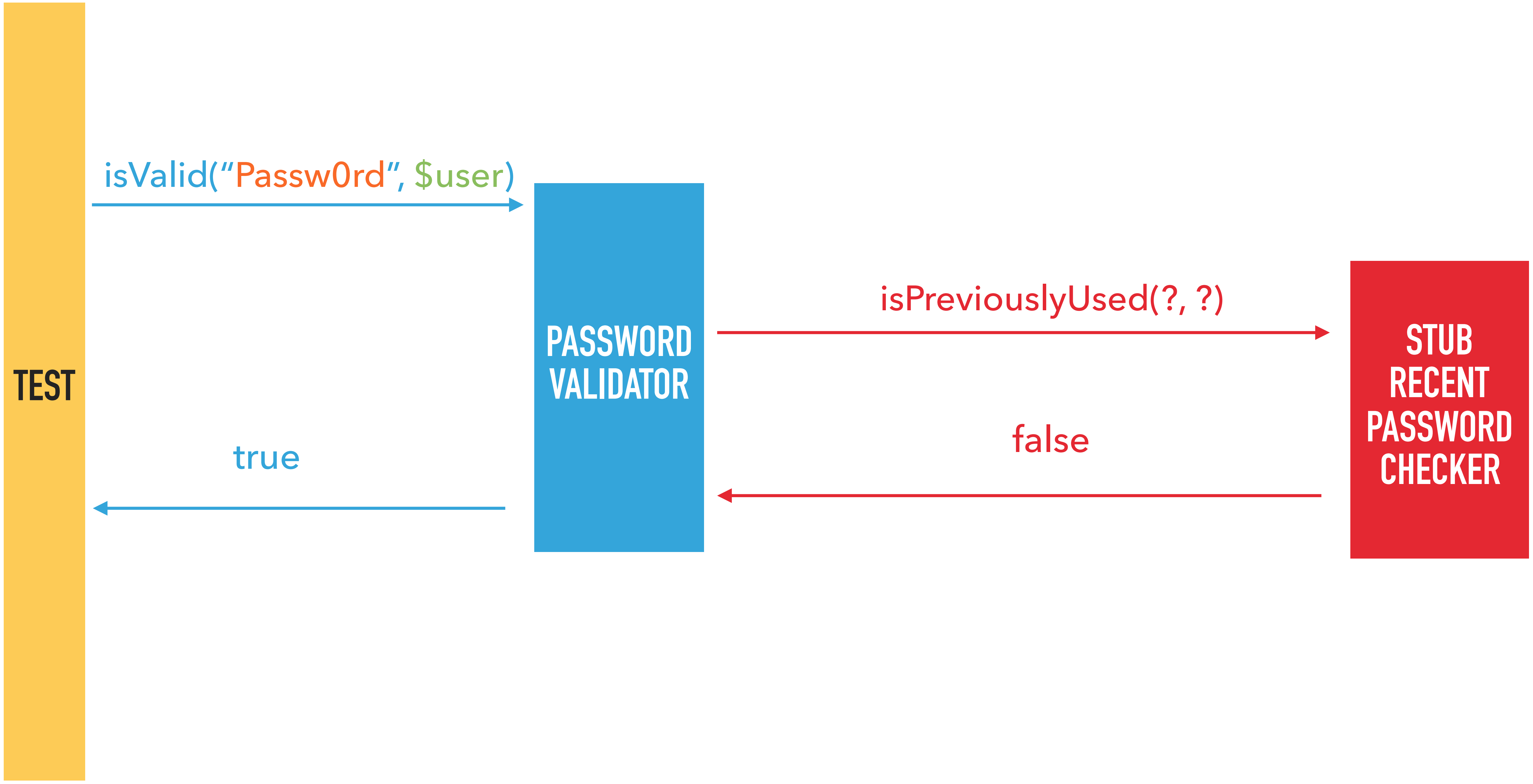
UNIT TESTS

UNIT TESTS

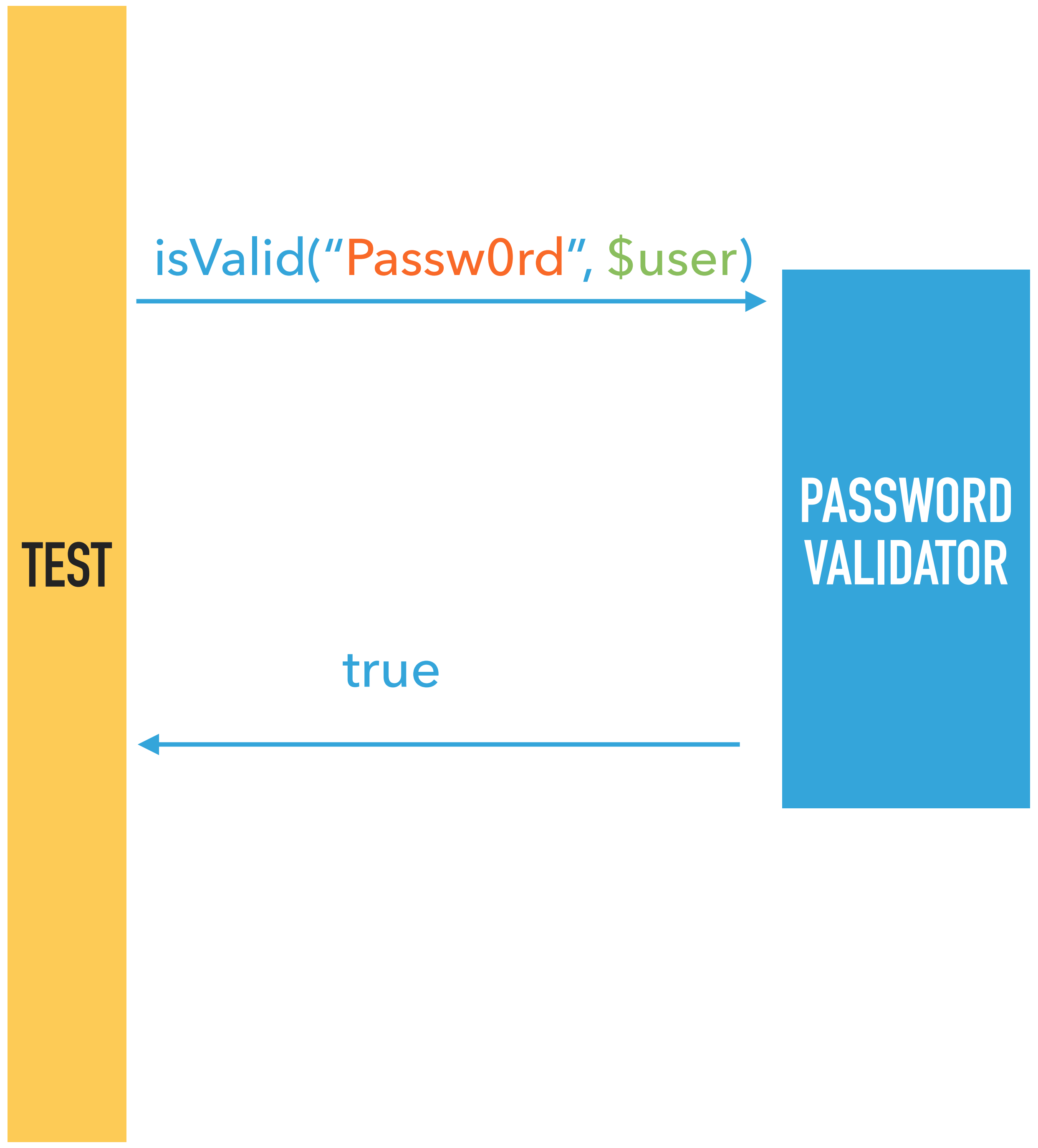
USE A STUB



USE A STUB


TEST
isValid("Passw0rd", $user)
PASSWORD VALIDATOR
isPreviouslyUsed(?, ?)
STUB RECENT PASSWORD CHECKER
false
true

USE A STUB



USE STUBS UNLESS YOU REALLY NEED MOCKS

- Mocks increase coupling between tests and code
  - Only use them when you really need to
  - Test harder to write
  - Reduces ability to refactor

WHY

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

AGENDA

- Why
- Terminology
- Do the right kind of tests at the right level
- Unit tests
- Building objects
- Tips
- Summary

BUILDING DATA FIXTURES

HAND BUILDING

```
$user = $this->userService->registerUser(
                “anna@acme.com”,
                “Anna”,
                “Passw0rd”);
```

HAND BUILDING

```
$user = $this->userService->registerUser(
                  “anna@acme.com”,
                  “Anna”,
                  “Passw0rd”);
$userService->completeRegistration(
        $user->getConfirmationToken);
```

HAND BUILDING

```
$user = $this->userService->registerUser(
                      “anna@acme.com”,
                      “Anna”,
                      “Passw0rd”)
$userService->completeRegistration(
          $user->getConfirmationToken);
```

OBJECT MOTHER

```
$user = $this->userObjectMother->getAnna();

// User will have default values for name,
// email, etc
```

OBJECT MOTHER: IMPLEMENTATION

```
class UserObjectMother {

    …

    public function getAnna(): User {

        $user = $userService->registerUser(
                    “anna@acme.com”,
                    “Anna”,
                    “Passw0rd”);



        return $user;
}
```

OBJECT MOTHER: IMPLEMENTATION

```
class UserObjectMother {

   …

   public function getAnna(): User {

      $user = $userService->registerUser(
                 “anna@acme.com”,
                 “Anna”,
                 “Passw0rd”);

      $userService->confirmRegistration(
          $user, $user->getToken());

      return $user;
}
```

TEST BUILDER: 1

```
$userBuilder = $this->getUserBuilder();
$user = $userBuilder->build();

// User will have default values for
// name, email, etc
```

USING A TEST BUILDER (2)

```
$userBuilder = $this->getUserBuilder();
$user = $userBuilder
          ->name(“Annabelle”)
          ->password(“Passw4rd”)
          ->previousPasswords([
              “Passw1rd”,
              “Passw2rd”,
              “Passw3rd”,
           ])
          ->build();
```

DEFER TO OTHER OBJECT MOTHERS / BUILDERS

```
class ProductObjectMother
{
 public function getCleanCodeBook(): Product {




    $product = … create Product …


    return $product;
}
```

DEFER TO OTHER OBJECT MOTHERS / BUILDERS

```
class ProductObjectMother
{
 public function getCleanCodeBook(): Product {

     $supplier = $this->supplierObjectMother
          ->getIPadsForUs();

     $product = … create Product …
     $product->setSupplier($supplier);

     return $product;
}
```

BUILDING TEST OBJECTS

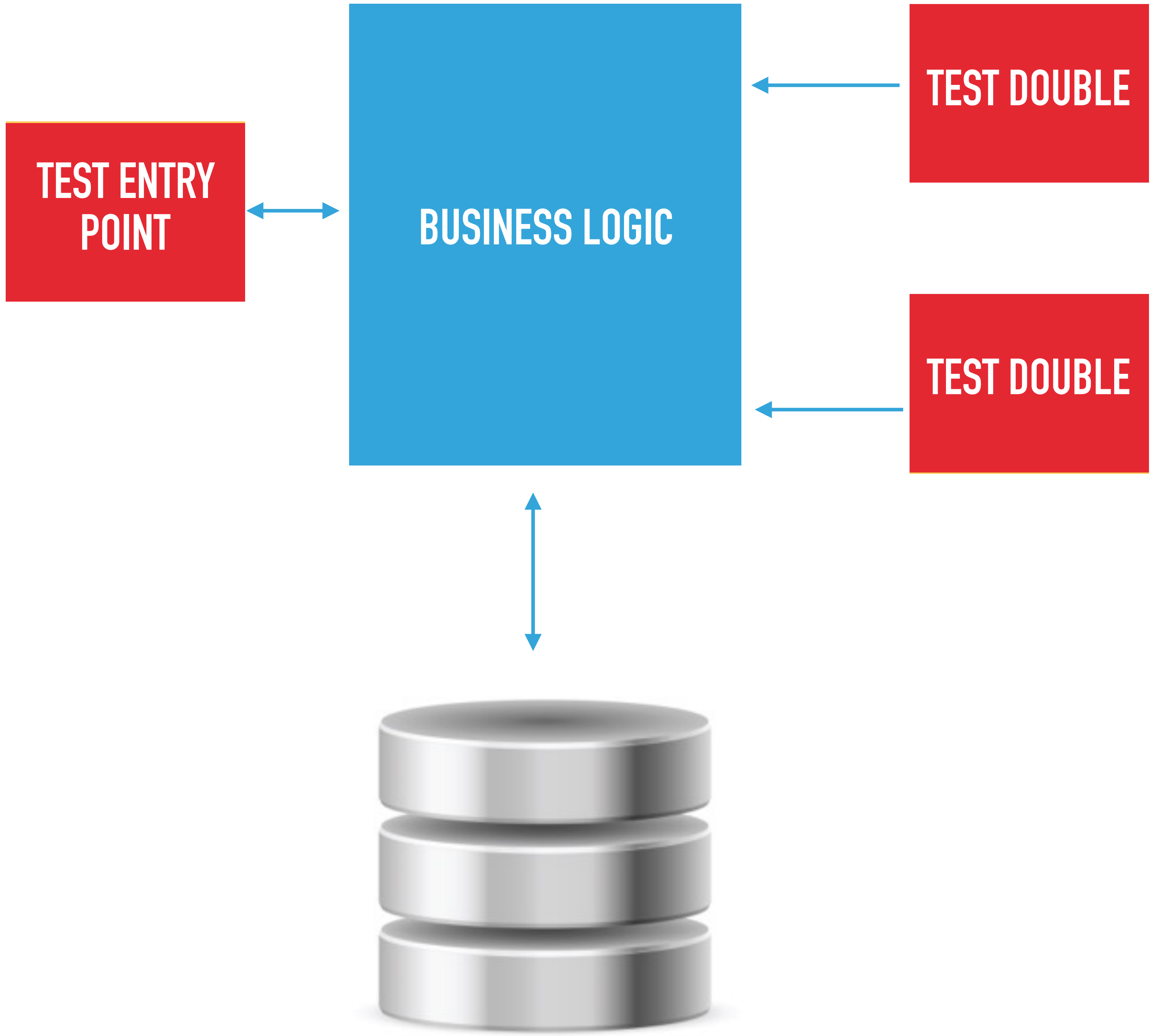
SEEDING A DATABASE

```
users:
  - name: Anna
    email: anna@acme.com
    password: Passw1rd


  - name: Bob
    email: bob@example.com
    password: Passw5rd
```

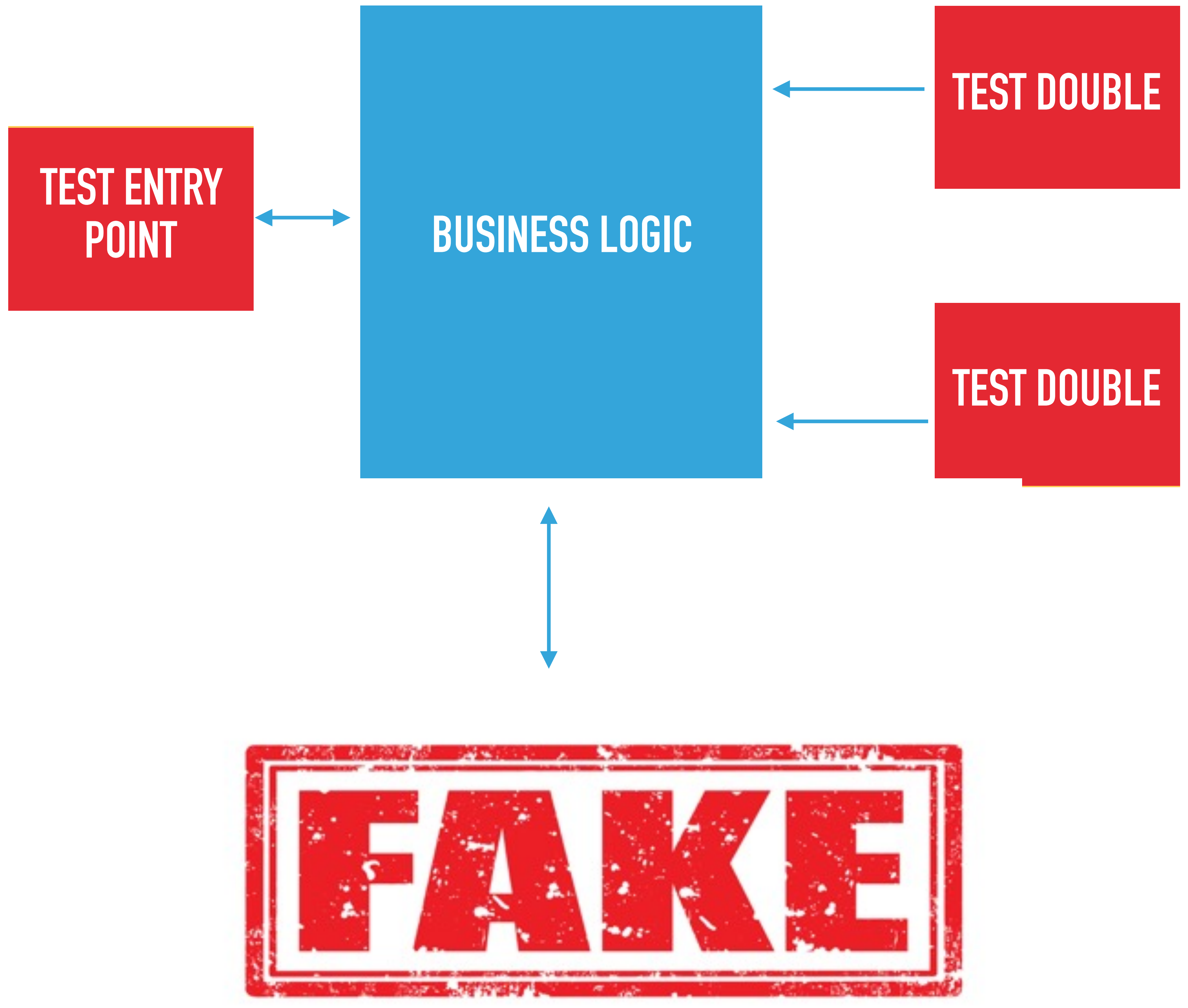
SEEDING A DATABASE

```
users:
  - name: Anna
    email: anna@acme.com
    password: Passw1rd


  - name: Bob
    email: bob@example.com
    password: Passw5rd
```

BUILDING TEST OBJECTS



BUILDING TEST OBJECTS



HYBRID

HOW DO WE BUILD THE TEST USER OBJECT?

- Hand build what is required
- Seed the database
- Object mother
- Test Builder

TAKE AWAY

OBJECT MOTHER AND TEST BUILDER BENEFITS

OBJECT MOTHER AND TEST BUILDER BENEFITS

- Single place where test business object built
    - Easy to find
    - Easy to update

OBJECT MOTHER AND TEST BUILDER BENEFITS

- Single place where test business object built
  - Easy to find
  - Easy to update
- Defer to other Object Mothers / Test Builders

OBJECT MOTHER AND TEST BUILDER BENEFITS

- Single place where test business object built
  - Easy to find
  - Easy to update
- Defer to other Object Mothers / Test Builders
- Decoupling our tests from the software under test
  - More robust to change
  - Easier to refactor

WHY

DECOUPLED TESTS REDUCE THE DEVELOPMENT AND MAINTENANCE COSTS OF THE TEST SUITE.

SUMMARY

SUMMARY

- Decoupling is good

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Test business logic at the service layer

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Test business logic at the service layer
  - Test UI is correctly wired up to service layer

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Test business logic at the service layer
  - Test UI is correctly wired up to service layer
- Prefer stubs to mocks (unless you really need them)

SUMMARY

- Decoupling is good
  - Reduces development and maintenance costs
- Do the right kind of tests at the right level
  - Test business logic at the service layer
  - Test UI is correctly wired up to service layer
- Prefer stubs to mocks (unless you really need them)
- Building objects using Object Mother / Builder patterns

Feedback

IMAGE CREDITS

- Decouple © Can Stock Photo / iqoncept
- Story © Can Stock Photo / Palto
- Man On Moon: © Can Stock Photo / openlens
- Confession © Can Stock Photo / lenm
- Pyramid © Can Stock Photo / Arcady
- Feedback © Can Stock Photo / kikkerdirk
- Scripts © Can Stock Photo / LoopAll
- Tools © Can Stock Photo / dedMazay
- Builder © Can Stock Photo / aleksangel
- Database © Can Stock Photo / dvarg
- Fake © Can Stock Photo / carmendorin
- People chatting © Can Stock Photo / studioworkstock
- Seeding: © Can Stock Photo / italianestro
- Banking app © Can Stock Photo / tashka2000
- Old Telephone © Can Stock Photo / barneyboogles
- Bank © Can Stock Photo / dolgachov
- Coupler © Can Stock Photo / ArtImages
- Bank Building © Can Stock Photo / dvarg
- Online Shopping © Can Stock Photo / Wetzkaz