codecentric

Wann soll ich mocken?

XP Days Germany

David Völkel

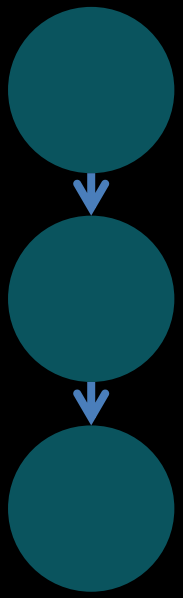
21.11.2016

@davidvoelkel

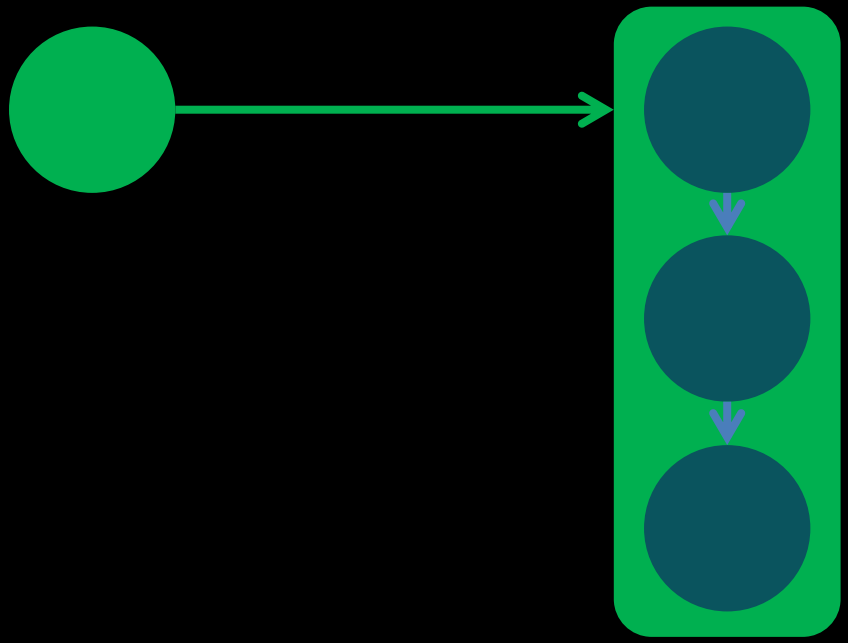
@softwerkskammer
@codecentric

TDD & Design

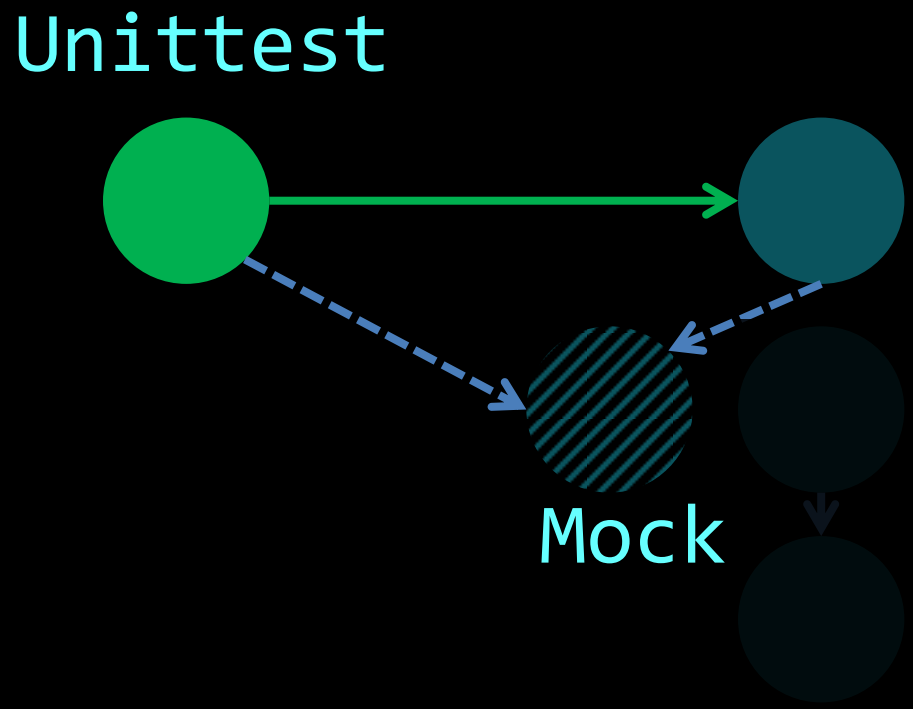
SCHICHTEN TESTEN?



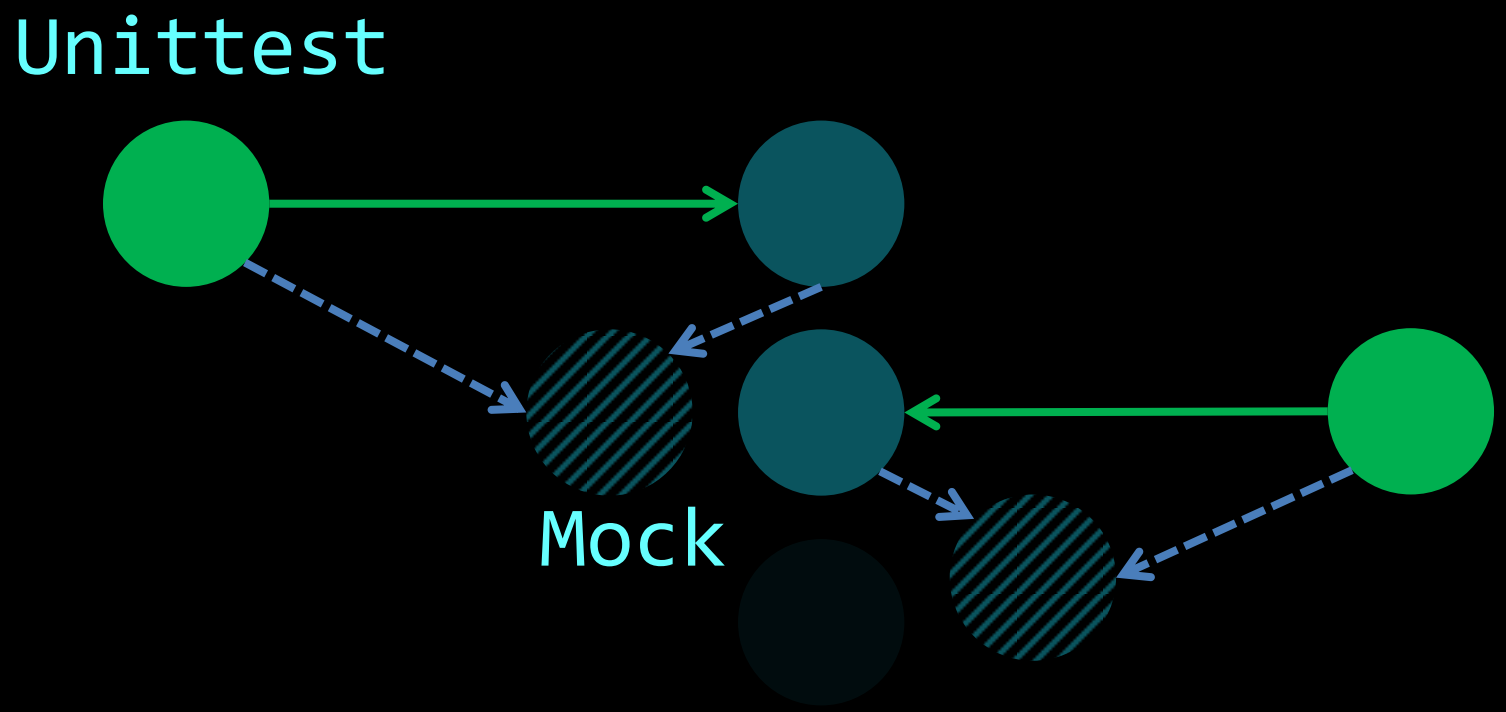
INTEGRIERTER TEST



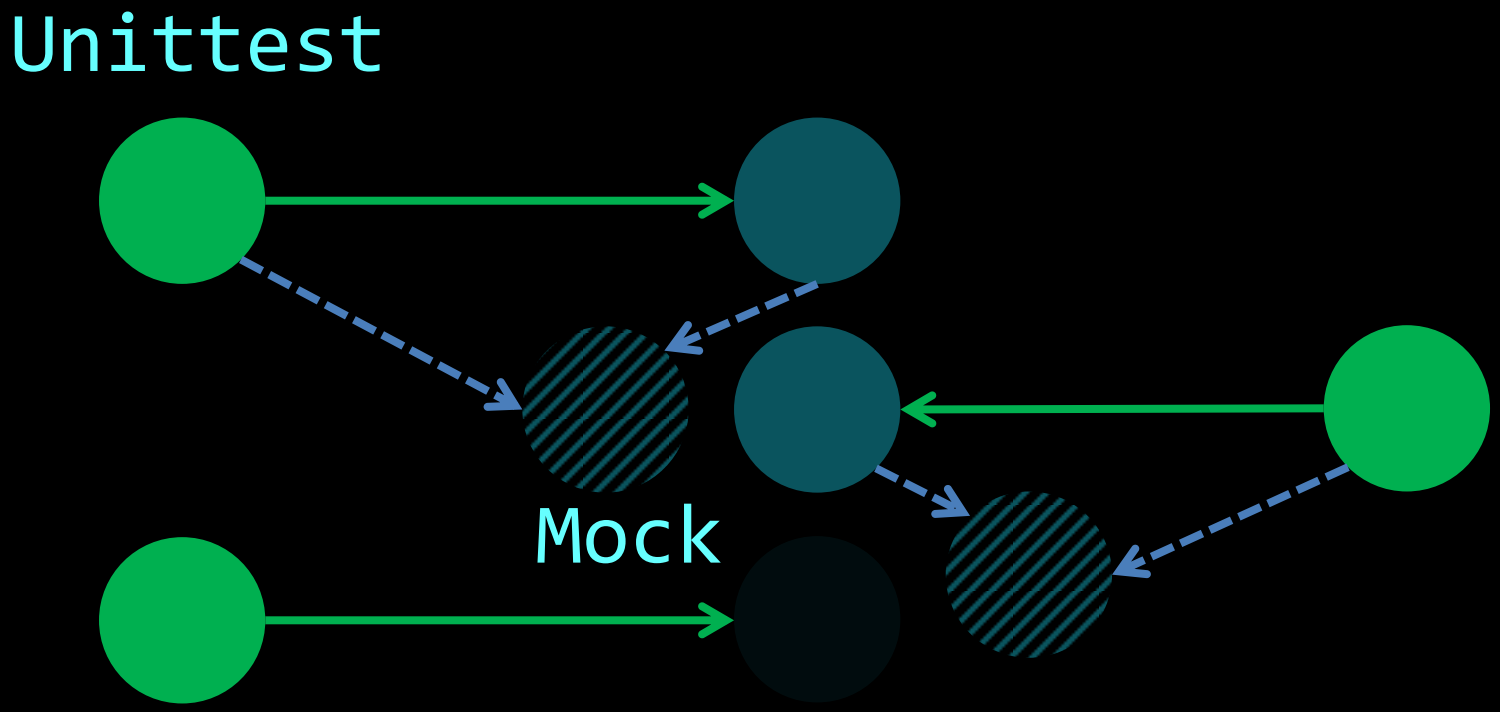
MOCKING



MOCKING



MOCKING



TRADE-OFF

ZU GROSS vs ZU KLEIN

aufwendiges Setup

Fehlerfindung

Testfällen

Langsames Feedback

Refactorability sinkt

Isolationsaufwand

Zu wenig Nutzen

Lesbarkeit

Woran orientieren ?

IDEOLOGIEN

MOCKISTS

CLASSICISTS

"Mocks Aren't Stubs", Martin Fowler

TRADE-OFFS
statt
IDEOLOGIEN!

USE CASES

Anbindung Drittsysteme

MOCKISTS

CLASSICISTS

USE CASES

Bedingte Interaktionen

Anbindung Drittsysteme

MOCKISTS

CLASSICISTS

USE CASES

Bedingte Interaktionen

SWEETSPOT

Anbindung Drittsysteme

MOCKISTS

CLASSICISTS

USE CASES

Geringer Nutzen

Bedingte Interaktionen

SWEETSPOT

Anbindung Drittsysteme

MOCKISTS **CLASSICISTS**

USE CASES

VERMEIDEN

Geringer Nutzen

SWEETSPOT

Bedingte Interaktionen

Anbindung Drittsysteme

MOCKISTS

CLASSICISTS

MOCKING

VERMEIDEN

SWEETSPOT

„BEST PRACTICES“

VERMEIDEN

DONT MOCK VALUES!

DONT MOCK VALUES!

```
new Value()
```

DONT MOCK VALUES!

```
new Value()

TestDataBuilderForValue
    .withDefaults()
    .withField(“1“)
```

INTEGRATION OPERATION SEGREGATION PRINCIPLE

"Integration Operation Segregation Principle", Ralf Westphal
"Die kniffligen Fälle beim Testen - Sichtbarkeit", Stefan Lieser

INTEGRATION OPERATION SEGREGATION PRINCIPLE

```
public void sendMailingTo(String email) {

    Customer customer = customerDB.findCustomerBy(email);

    String title = customer.getSex() == Sex.MALE ? "Mr" :
       customer.getMaritialStatus() == MaritialStatus.MARRIED ?
       "Mrs" : "Ms";
    String content = "Hello " + title + ". " + customer.getName() + ",\n\n"
       + "We have a special offer for you.\n\n"
       + "Best regards,\n"
       + "ACME Customer Service";

    mailService.sendMail(email, content);

}
```

INTEGRATION OPERATION SEGREGATION PRINCIPLE

```
public void sendMailingTo(String email) {

    Customer customer = customerDB.findCustomerBy(email);

    String title = customer.getSex() == Sex.MALE ? "Mr" :
       customer.getMaritialStatus() == MaritialStatus.MARRIED ?
       "Mrs" : "Ms";
    String content = "Hello " + title + ". " + customer.getName() + ",\n\n"
       + "We have a special offer for you.\n\n"
       + "Best regards,\n"
       + "ACME Customer Service";

    mailService.sendMail(email, content);

}
```

OPERATION

INTEGRATION OPERATION SEGREGATION PRINCIPLE

```
public void sendMailingTo(String email) {

    Customer customer = customerDB.findCustomerBy(email);          INTEGRATION

    String title = customer.getSex() == Sex.MALE ? "Mr" :
       customer.getMaritialStatus() == MaritialStatus.MARRIED ?
       "Mrs" : "Ms";
    String content = "Hello " + title + ". " + customer.getName() + ",\n\n"
       + "We have a special offer for you.\n\n"
       + "Best regards,\n"                                          OPERATION
       + "ACME Customer Service";

    mailService.sendMail(email, content);                          INTEGRATION

}
```

INTEGRATION OPERATION SEGREGATION PRINCIPLE

```
public void sendMailingTo(String email) {
    Customer customer = customerDB.findCustomerBy(email);

    String content = renderContent(customer);

    mailService.sendMail(email, content);
}

private String renderContent(Customer customer) {
    String title = customer.getSex() == Sex.MALE ? "Mr" :
       customer.getMaritialStatus() == MaritialStatus.MARRIED ?
       "Mrs" : "Ms";
    return "Hello " + title + ". " + customer.getName() + ",\n\n"
       + "We have a special offer for you.\n\n"
       + "Best regards,\n"
       + "ACME Customer Service";
}
```

INTEGRATION

OPERATION

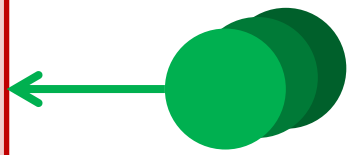
TESTS?

```
public void sendMailingTo(String email) {
    Customer customer = customerDB.findCustomerBy(email);

    String content = renderContent(customer);

    mailService.sendMail(email, content);
}
```

INTEGRATION

N Unittests

```
private String renderContent(Customer customer) {
    String title = customer.getSex() == Sex.MALE ? "Mr" :
       customer.getMaritialStatus() == MaritialStatus.MARRIED ?
       "Mrs" : "Ms";
    return "Hello " + title + ". " + customer.getName() + ",\n\n"
       + "We have a special offer for you.\n\n"
       + "Best regards,\n"
       + "ACME Customer Service";
}
```



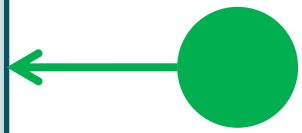
OPERATION

TESTS?

1 Integrierter Test

```
public void sendMailingTo(String email) {
    Customer customer = customerDB.findCustomerBy(email);

    String content = renderContent(customer);

    mailService.sendMail(email, content);
}
```

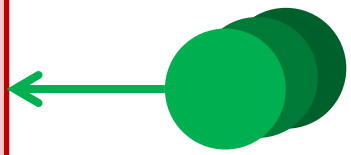
INTEGRATION



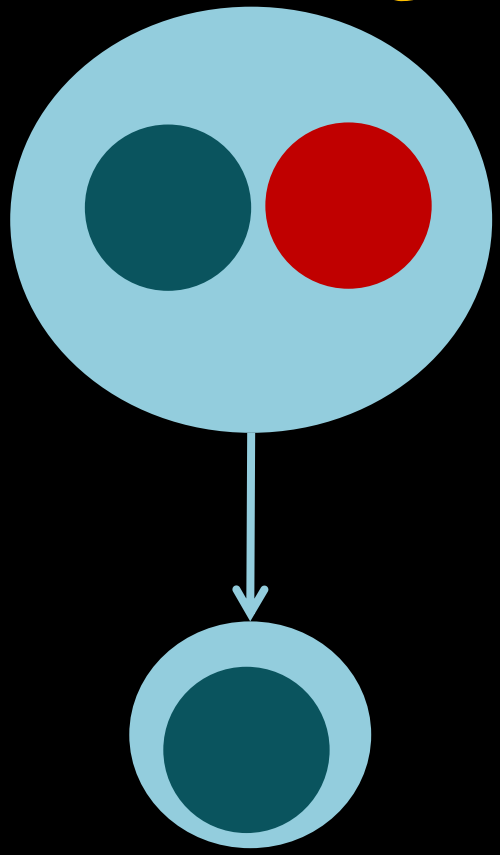
N Unittests

```
private String renderContent(Customer customer) {
    String title = customer.getSex() == Sex.MALE ? "Mr" :
       customer.getMaritialStatus() == MaritialStatus.MARRIED ?
       "Mrs" : "Ms";
    return "Hello " + title + ". " + customer.getName() + ",\n\n"
       + "We have a special offer for you.\n\n"
       + "Best regards,\n"
       + "ACME Customer Service";
}
```

OPERATION



PUSH LOGIC DOWN THE STACK

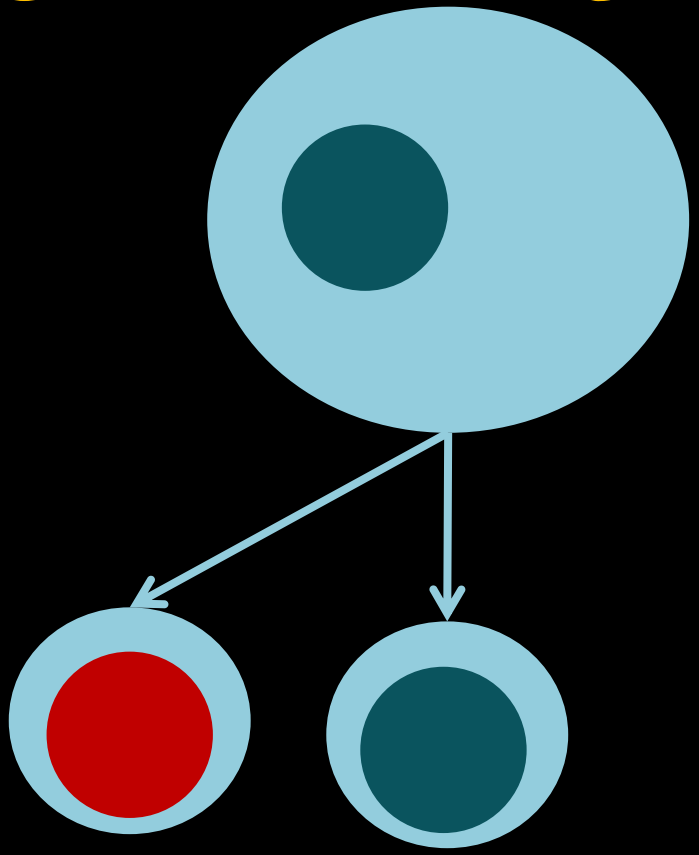


INTEGRATION

OPERATION

Siehe "The Failures of “Intro to TDD”" - Justin Searls

PUSH LOGIC DOWN THE STACK



INTEGRATION

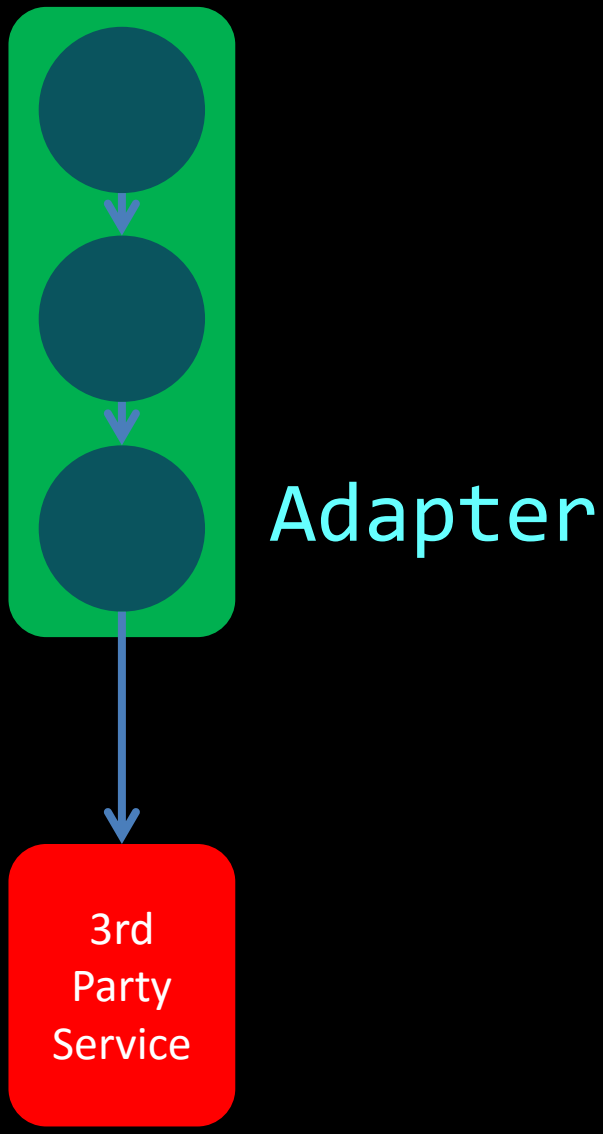
OPERATION

SWEETSPOT

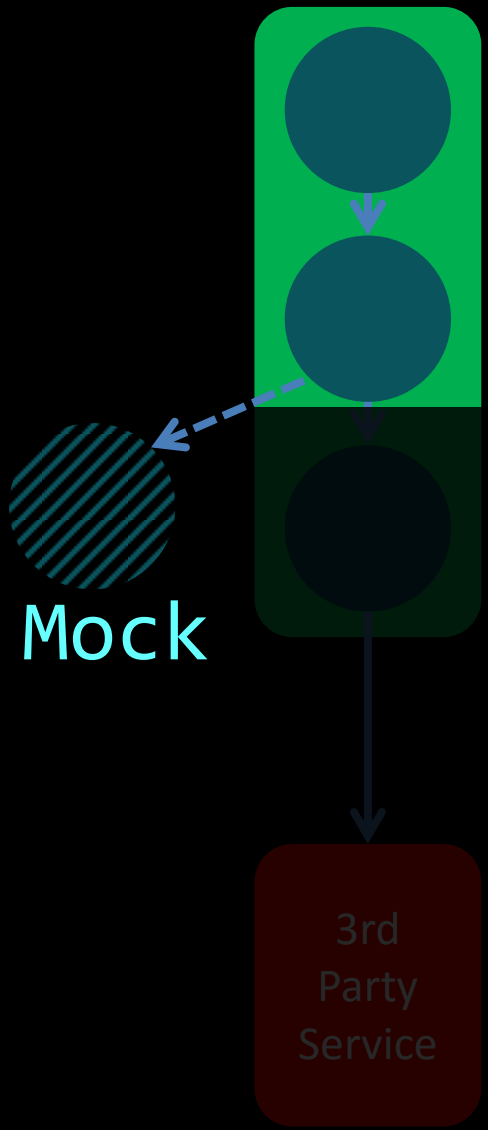
BEDINGTE INTERAKTION

```
public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        return "Welcome " + username;
    } else {
        return "Username ' " + username + "' "
         + "already taken, please choose another";
    }
}
```

SYSTEM GRENZEN



SYSTEM GRENZEN



OUTSIDE-IN DESIGN

OUTSIDE-IN DESIGN

Alternative “Fake it“

BEST PRACTICES

NO OVERSPECIFICATION!

NO OVERSPECIFICATION!

"Specify exactly what should happen but no more"

REIHENFOLGE?

COMMAND & QUERY SEPARATION

```
public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
    } else {
        ...
    }
}
```

COMMAND & QUERY SEPARATION

```
public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
    } else {
        ...
    }
}
```

COMMAND & QUERY SEPARATION

```
public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
    } else {
        ...
    }
}
```

COMMAND & QUERY

```
@Test public void signup() throws Exception {
    ...
    when(userDB.findUserBy(anyString())).thenReturn(null);

    mailingService.signup(username);

    verify(userDB).createUser(new User(username));
}

public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
    } else {
        ...
    }
}
```

COMMAND & QUERY

```
@Test public void signup() throws Exception {
    ...
    when(userDB.findUserBy(anyString())).thenReturn(null);

    mailingService.signup(username);

    verify(userDB).createUser(new User(username));
}

public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
    } else {
        ...
    }
}
```

COMMAND & QUERY

```
@Test public void signup() throws Exception {
    ...
    when(userDB.findUserBy(anyString())).thenReturn(null);

    mailingService.signup(username);

    verify(userDB).createUser(new User(username));
}

public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
    } else {
        ...
    }
}
```

COMMAND & QUERY

```
@Test public void signup() throws Exception {
    ...
    when(userDB.findUserBy(anyString())).thenReturn(null);

    mailingService.signup(username);

    verify(userDB).createUser(new User(username));
}

public String signup(String username) throws Exception {
    if(userDB.findUserBy(username) == null) {
        userDB.createUser(new User(username));
        ...
```

„Allow Queries, expect Commands!“

LISTEN TO YOUR TESTS!

Image by M.J. Moneymaker

LISTEN TO YOUR TESTS!

```
# Verifications
```

Image by M.J. Moneymaker

LISTEN TO YOUR TESTS!

```
# Verifications
  ~~Overspecification~~!
```

Image by M.J. Moneymaker

LISTEN TO YOUR TESTS!

Verifications

~~Overspecification~~!

Dependencies

Image by M.J. Moneymaker

LISTEN TO YOUR TESTS!

```
# Verifications
  ~~Overspecification~~!
# Dependencies
  Extract Class!
```

Image by M.J. Moneymaker

LISTEN TO YOUR TESTS!

```
# Verifications
  ~~Overspecification~~!
# Dependencies
  Extract Class!
# Interactions
```

Image by M.J. Moneymaker

LISTEN TO YOUR TESTS!

```
# Verifications
  Overspecification!
# Dependencies
  Extract Class!
# Interactions
  Tell, don‘t ask!
```

Image by M.J. Moneymaker

TELL DONT ASK

```
public void volumeUpClicked() {
    int volume = speaker.getVolume();
    if (volume < speaker.getMaximumVolume()) {
        speaker.setVolume(volume++);
    }
}
```

DONT ASK

```
public void volumeUpClicked() {
    int volume = speaker.getVolume();
    if (volume < speaker.getMaximumVolume()) {
        speaker.setVolume(volume++);
    }
}
```

TELL DONT ASK

```
public void volumeUpClicked() {
    int volume = speaker.getVolume();
    if (volume < speaker.getMaximumVolume()) {
        speaker.setVolume(volume++);
    }
}
```

```
public void volumeUpClicked() {
    speaker.putUpVolume();
}
```

```
class Speaker {
    public void putUpVolume() {
        if (this.volume < this.maximum
            this.volume++;
        }
    }

    ...
```

VERMEIDEN

SWEETSPOT

BEST PRACTICES

MOCKIST
ODER
CLASSICIST?

~~MOCKIST~~

~~ODER~~

~~CLASSICIST?~~

TRADE-OFFS!

QUELLEN

- „Growing Object Oriented Systems“, Nat Pryce, Steve Freeman
- "Mocks Aren't Stubs", Martin Fowler
- "Integration Operation Segregation Principle", Ralf Westphal
- "Die kniffligen Fälle beim Testen - Sichtbarkeit", Stefan Lieser

codecentric

Q&A ?!

Licence