Testautomatisering

Labbar, FitNesse, TDD, BDD

- Lab 4
  - Utökad deadline?

Lab 4

Idag

- FM:
  - Lab 1-3 snack
  - FitNesse
  - TDD
  - BDD
- EM:
  - Handledning

- Watir::Wait.until {"OK"}

Lab 1-3

- I Ruby:

- False: false eller nil
- True: Allt annat

- if (”false”)
- puts ”Detta kommer skrivas ut!”
- end

Lab 1-3

- Konotrollera om URL stämmer
- Kontrollera om Text finns

- => Dess genereras dynamiskt och kan vara olika från användare till användare

- Undvika att testa mot dessa

Lab 1-3

- require behövs bara en gång
- Browser.new behövs potentiellt bara en gång

Lab 1-3

- ”if logged_in == true”
- Innebär ”if true == true” eller ”if false == true”
- Ändra till ”if logged_in”
- Eller ännu bättre ”if is_logged_in?”

Lab 1-3

- Verifiera att ”*Obligatorisk” finns på sidan
- Denna text fanns på sidan från början, så den säger ingenting om att ett felmeddelande dykt upp.

Lab 1-3

- Random crashes: Förmodligen så försöker ni använda ett objekt som inte finns ännu

- wait_until_present(button)

Lab 1-3

- Om vi vill ha längden på något:
  - Vi kan göra på massa olika sätt
  - rätt sätt är nästan alltid .length, .count

Lab 1-3

- assert_equal(true, actual, "...")
  - Ersätt med: assert(actual,"...")
- ...också: definiera i detta fall ingen expected under arrange

Lab 1-3

- Använd inte: return ’false’
  - Använd: return false
- ”false” är truthy
- return "false" => potentiella problem med if (...)
- Blanda inte ihop false och "false"

Lab 1-3

- om vi är ute efter beskrivningen av en product
  - använd: product.description
  - använd inte: product.to_s + ex. regexp + match

Lab 1-3

- Cartline - om den hanterar en product
  - initiera med ett objekt av typen Product
  - Initiera inte med en sträng "Product” (generellt)

Lab 1-3

- Håll tungan rätt i mun:

  expected = "-"
  actual = @user.fullname
  assert_equal(expected, ”-”, ”…”)

  Se till att er assert gör något.

  - (Några labbar var misstänkt lika andra labbar ;)
  - Kopiering från nätet är vardag i branschen – jag tänker absolut inte säga att ni inte skall göra det.
  - …När ni kopierar från (felaktiga?) källor på nätet: fundera över vad ni lärt er och ifall ni förstår

Lab 1-3

- Unexptected keyword: end
  - Indentering

```
class
def my_method
#…
end
def my methdod2
#...
end
```

Lab 1-3

- Unexptected keyword: end
  - Indentering

```
class
        def my_method
                #…
        end

        def my methdod2
                #...
        end

end
```

Lab 1-3

- Blanda inte ihop implementation av en applikation med implementation av tester

Lab 1-3

- Slut Lab 1-3 snack

Lab 1-3

Enhetstester

- Problem med mjukvara: För buggigt
  - => Enhetstester (Unit Tests)

TDD

- 3 st. Problem med att utveckla enhetstester sist
  - Vi hinner inte utveckla dem
  - Vi har skrivit för mycket / onödig logik i implementationen av det som skall testas
  - Att veta hur mycket enhetstester som är tillräckligt är ofta väldigt svårt

TDD

- Red -> Green -> Refactor
  - Red: Skriv ett test som inte passerar
  - Green: Skriv minimal implementation för att få testet att passera
  - Refactor: Snygga till implementation/tester

- Problem med att utveckla enhetstester sist
  - Vi hinner inte utveckla dem
    - TDD => Test first, vi kommer inte ha implementation utan tester

TDD

TDD

- Problem med att utveckla enhetstester sist
  - Vi har skrivit för mycket / onödig logik i implementationen av det som skall testas
    - TDD => skriv minimal implementation för att få testet att passera.

TDD

- Problem med att utveckla enhetstester sist
  - Att veta hur mycket enhetstester som är tillräckligt är ofta väldigt svårt
    - TDD => När vi känner att implementationen är klar så har vi tester som täcker det vi implementerat iom. att vi utvecklar testerna först och vi bara har skrivit en minimal implementation.

BDD

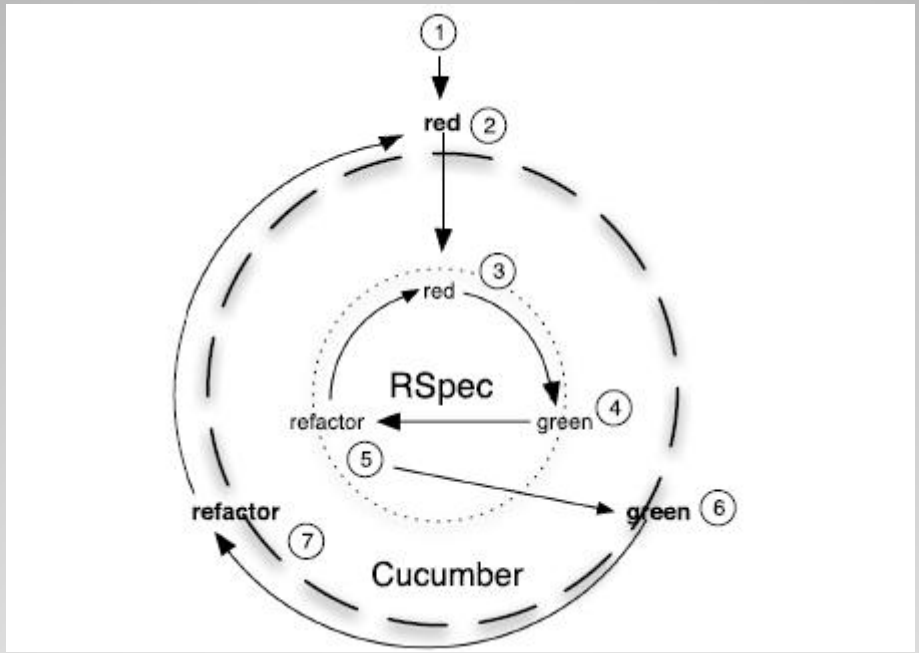
- Ett problem med klassisk TDD: Ingenting i processen tvingar oss att fundera kring affärsvärdet i det vi utvecklar (gäller både tester och implementionen som skall testas).
  - Jag har rackat ned på TDD under denna kursen. Få inte intrycket att TDD är dåligt.
  - Min bedömning: Fortfarande vanligare än BDD
  - De flesta som jobbar med TDD har en strategi för att fånga ovanstående problem.

BDD

- Ett problem med klassisk TDD: Affärsvärde?
  - BDD-lösningen: Lägg till ytterligare en Red -> Green -> Refactor cykel som har ett utifrån-perspektiv.

- 2 Red->Green->Refaktor-cykler
  - Yttre cykel: Affärsvärde?
    - Utifrån-perspektiv
    - Implementera rätt features!
  - Inre cykel: Korrekt implementation?
    - Innifrån-perspektiv
    - Implementera features på rätt sätt!

BDD


1
red
2
3
red
RSpec
refactor
green
4
5
refactor
green
6
7
Cucumber

BDD

FitNesse

- FitNesse
  - Lab 4:
    - Kolla post på hemsidan med frågor jag svarat på.

- FitNesse
  - Hur hänger det ihop

FitNesse

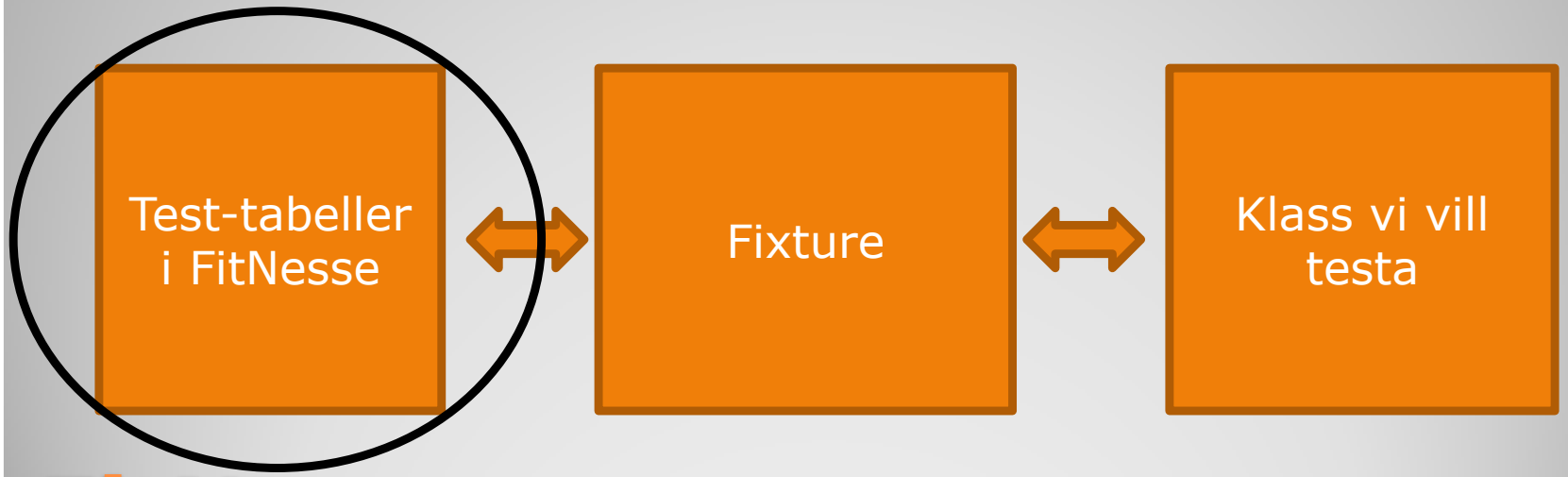
- Målbild:

FitNesse

- Målbild:
  - När vi kommit hit innebär ett nytt test bara att vi lägger till en ny rad i en tabell på aktuell testsida:
    - Ex: |Nytt|Test|Nytt Test|

FitNesse

- FitNesse Servern behöver startas
  - Ex: java –jar fitnesse-standalone.jar –p 8080
  - Normalt: FitNesse ligger på en dedikerad server och är alltid igång.



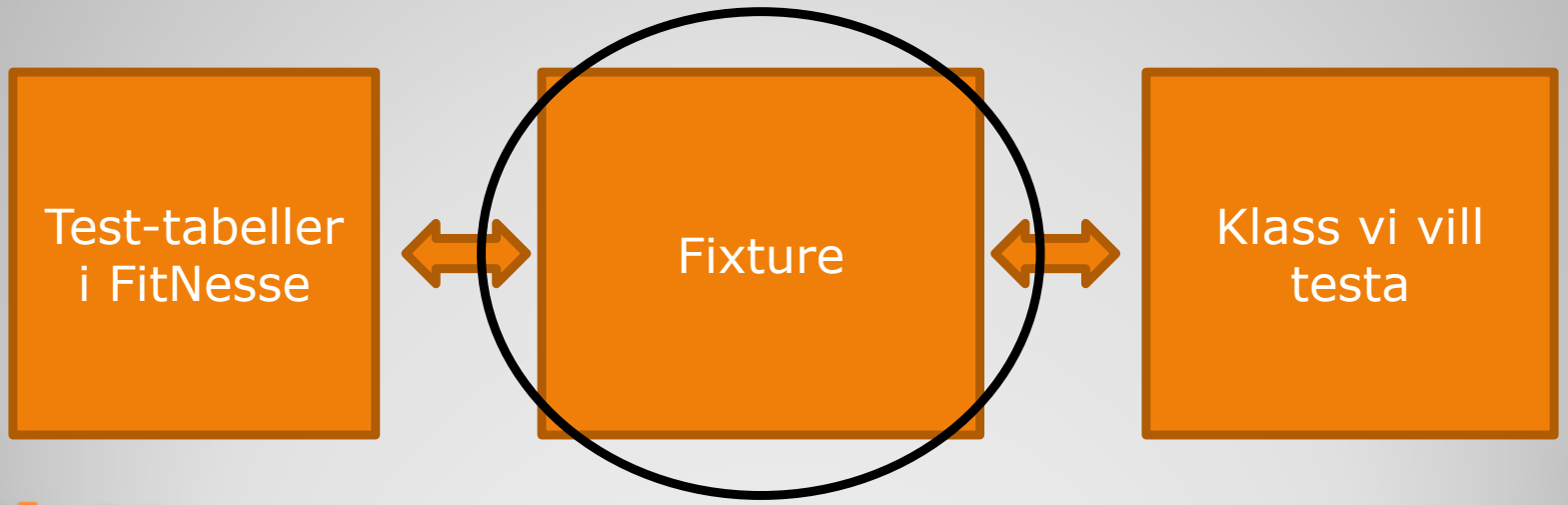
- RubySlim:
  - Kopplar ihop FitNesse med Fixture
  - Behöver initiering på t.ex. en startsida i FitNesse. Vi behöver också en Import-tabell på en SetUp-sida.
  - Lite krångligt, men görs en gång

FitNesse

- Koppling till Fixture
  - Sökvägar på startsidan:
    - !path /to/your/lib/fixtures
    - !path /to/your/lib

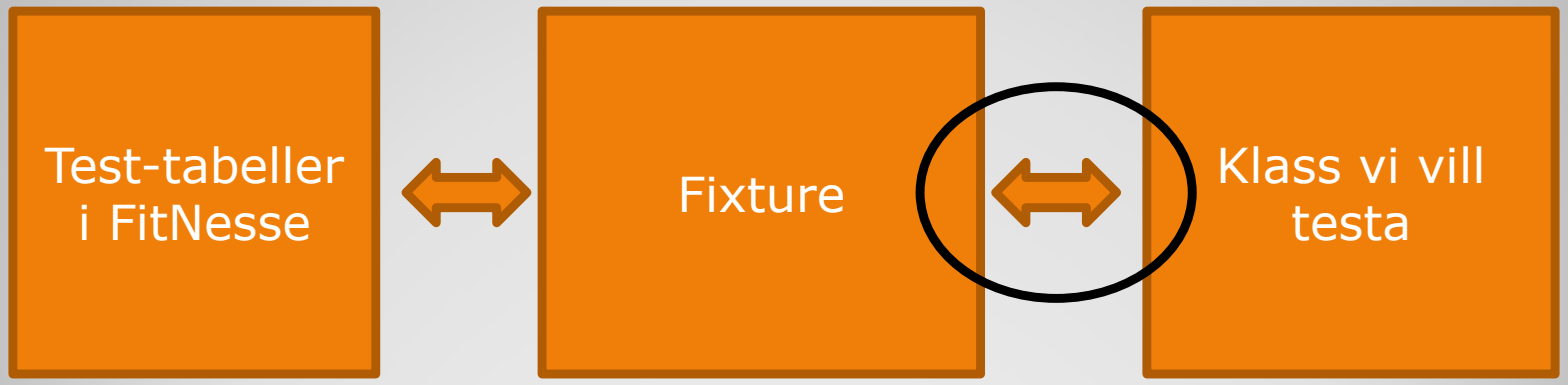
FitNesse

- Fixture-klass
  - Behöver skrivas i ex. ruby
  - Namn på klass: bygger på tabellen i fitnesse
  - Input från tabeller i fitnesse: metod, ”set_”
  - Output till tabeller i fitness: metod



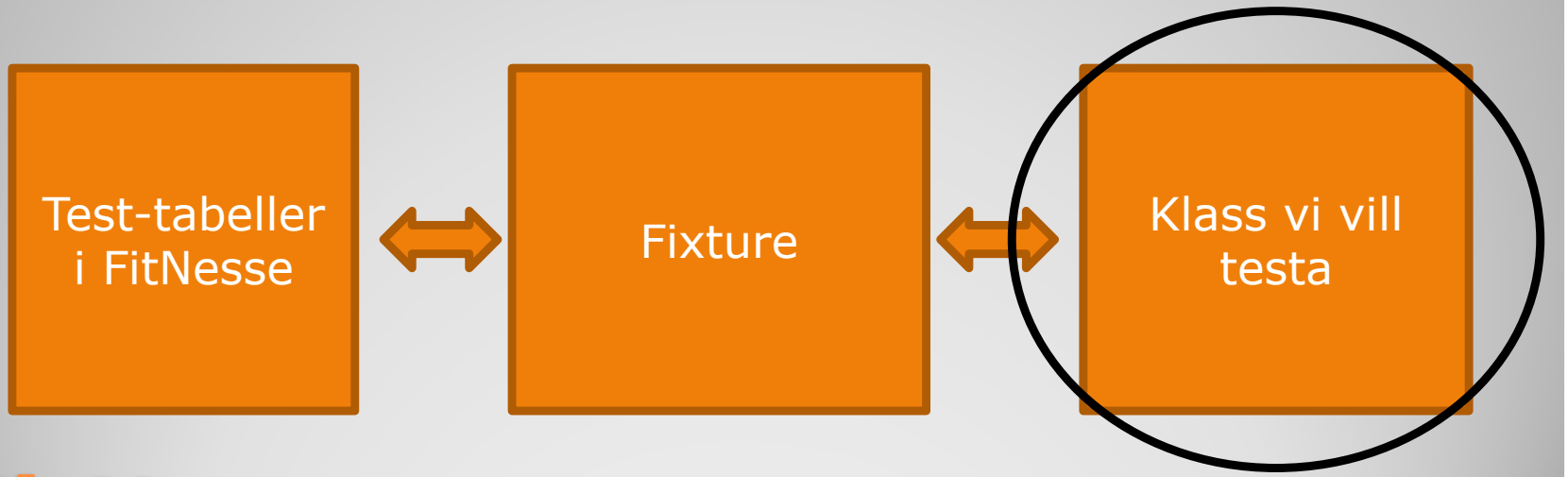
FitNesse

- Koppling mellan Fixture <-> klass som skall testas
  - En ”require” i Fixture-klassen



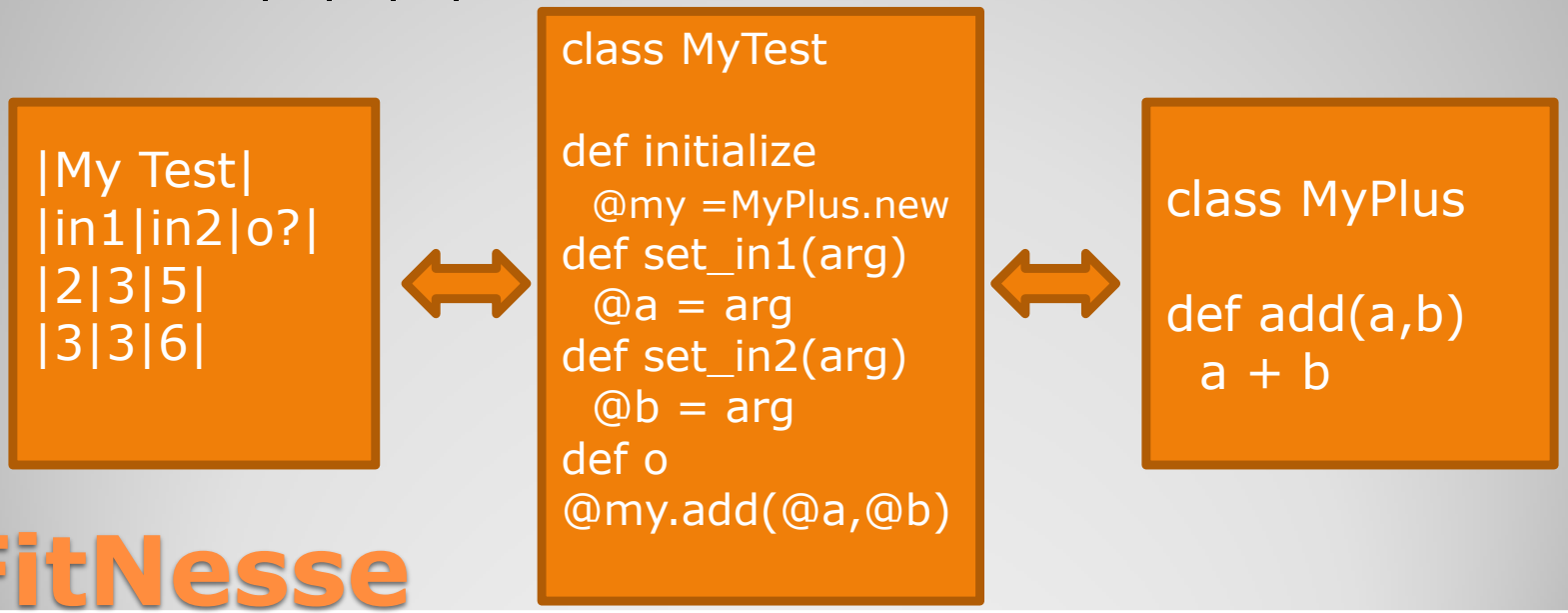
FitNesse

- Klass som skall testas
  - I vårt fall, för G-uppgiften: MyString



FitNesse

- Målbild:
  - När vi kommit hit innebär ett nytt test bara att vi lägger till en ny rad i en tabell på aktuell testsida:
    - Ex: |3|3|6|



FitNesse

- Vi vill testa flera metoder i en klass
  - => utöka fixture med fler input-metoder och output-metoder

  - |MyFixture|
  - |input first1|input first2|first method?|
  - |in|in|expected out|

  - |MyFixture|
  - |input second1|input second2|second method?|
  - |in|in|another expected out|

FitNesse

- Vi vill testa flera metoder i en klass
  - exempel

FitNesse

FitNesse

- Implementation av MyString + Fixture
  - Exempel
    - .replace(str)

Nästa vecka

- Orientering Continuous Integration
- Handledning – samtliga labbar
- Hemtenta

- ...

Fin