Ente Ospedaliero Cantonale

To TDD or not TDD

Rossano Bersagliere

TDDing in java with full IDE support? Too easy!

TDDing in java with full IDE support? Too easy!

How about…

TDDing in java with full IDE support? Too easy!

How about… BASH Test Driven Development?

Learing as I go

Roadmap:

- Selecting tools and gathering ideas
- Bash TDD
- Fun stuff

Tools and ideas

I need 4 things:

Tools and ideas

I need 4 things:

- Basic knowledge of bash

Tools and ideas

I need 4 things:

- Basic knowledge of bash

Tools and ideas

I need 4 things:

- Basic knowledge of bash
- A junit like testing framework

Tools and ideas

I need 4 things:

- Basic knowledge of bash ☾ 

- A junit like testing framework ☾ shUnit2*

* https://github.com/kward/shunit2

Tools and ideas

I need 4 things:

- Basic knowledge of bash ☾ 

- A junit like testing framework ☾ shUnit2*
- An IDE

* https://github.com/kward/shunit2

Tools and ideas

I need 4 things:

- Basic knowledge of bash ☾ 

- A junit like testing framework ☾ shUnit2*
- An IDE (sort of) ☾ how about 3 tiled shell terminals

* https://github.com/kward/shunit2

Tools and ideas

I need 4 things:

- Basic knowledge of bash ☾ 

- A junit like testing framework ☾ shUnit2*
- An IDE (sort of) ☾ how about 3 tiled shell terminals
- A classic TDD kata

* https://github.com/kward/shunit2

Tools and ideas

I need 4 things:

- Basic knowledge of bash ☾ 

- A junit like testing framework ☾ shUnit2*
- An IDE (sort of) ☾ how about 3 tiled shell terminals
- A classic TDD kata ☾ nth Fibonacci

* https://github.com/kward/shunit2

Tools and ideas – kata

Classic TDD

Nth Fibonacci
Write a function that generates the Fibonacci number for the nth position implementing:

```
int Fibonacci(int position)
```

First Fibonacci numbers in the sequence are:

Position	0	1	2	3	4	5	6	7	8	9
Fibonacci	0	1	1	2	3	5	8	13	21	34

Tools and ideas – scaffolding

```
#! /bin/sh

testTodo() {
  assertEquals 1 1
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
"fibonacci_test.sh" 13L, 150C written                    4,18          All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position
~
"fibonacci.inc" 3L, 94C
```

```
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testTodo

Ran 1 test.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ 
```

Bash TDD

Let’s begin!

Bash TDD – first test

```
#! /bin/sh

testShouldReturn0for0() {
  result=`fibonacci 0`
  assertEquals 0 "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
```

```
"fibonacci_test.sh" 14L, 197C written                          3,1           All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position
```

```
"fibonacci.inc" 3L, 94C
```

```
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
./fibonacci_test.sh: 1: eval: fibonacci: not found
ASSERT:expected:<0> but was:<>
shunit2:ERROR testShouldReturn0for0() returned non-zero return code.

Ran 1 test.

FAILED (failures=1)
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – fake implementation 1

```
#! /bin/sh

testShouldReturn0for0() {
  result=`fibonacci 0`
  assertEquals 0 "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 14L, 197C written                    3,1           All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  echo 0
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci.inc" 7L, 120C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0

Ran 1 test.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – fake implementation 2

```
#! /bin/sh

testShouldReturn0for0() {
  result=`fibonacci 0`
  assertEquals 0 "${result}"
}

testShouldReturn1for1() {
  result=`fibonacci 1`
  assertEquals 1 "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci test.sh" 19L, 279C written                    10,16         All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -eq 0 ]; then
    echo 0
    return
  fi

  echo 1
}
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci.inc" 14L, 202C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1

Ran 2 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – fake implementation 3

```
#! /bin/sh

testShouldReturn0for0() {
  result=`fibonacci 0`
  assertEquals 0 "${result}"
}

testShouldReturn1for1() {
  result=`fibonacci 1`
  assertEquals 1 "${result}"
}

testShouldReturn1for2() {
  result=`fibonacci 2`
  assertEquals 1 "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 24L, 361C written                    14,21         All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -eq 0 ]; then
    echo 0
    return
  fi

  echo 1
}
~
~
~
~
~
~
~
~
~
~
~
"fibonacci.inc" 14L, 202C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2

Ran 3 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – test refactoring

```
#! /bin/sh

testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }

_shouldReturnNumberForPosition() {
  fibonacciNumber="${1}"
  position="${2}"

  result=`fibonacci "${2}"`
  assertEquals "${1}" "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 21L, 452C written                    11,16         All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -eq 0 ]; then
    echo 0
    return
  fi

  echo 1
}
~
~
~
~
~
~
~
~
~
~
~
"fibonacci.inc" 14L, 202C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2

Ran 3 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – obvious implementation

```
#! /bin/sh

testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }

_shouldReturnNumberForPosition() {
  fibonacciNumber="${1}"
  position="${2}"

  result=`fibonacci "${2}"`
  assertEquals "${1}" "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 22L, 516C written                    7,0-1         All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -eq 0 ]; then
    echo 0
    return
  fi

  if [ "${position}" -eq 1 ]; then
    echo 1
    return
  fi

  echo $(( $( fibonacci $((position - 2)) ) + $( fibonacci $((position - 1)) ) ))
}
~
~
~
~
~
~
"fibonacci.inc" 19L, 339C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2
testShouldReturn2for3

Ran 4 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – refactoring

```
#! /bin/sh

testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }

_shouldReturnNumberForPosition() {
  fibonacciNumber="${1}"
  position="${2}"

  result=`fibonacci "${2}"`
  assertEquals "${1}" "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 22L, 516C written                    7,0-1         All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -le 1 ]; then
    echo "${position}"
    return
  fi

  previous=$( fibonacci $((position - 2)) )
  second_previous=$( fibonacci $((position - 1)) )

  echo $(( previous + second_previous ))
}
~
~
~
~
~
~
~
~
~
"fibonacci.inc" 17L, 343C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2
testShouldReturn2for3

Ran 4 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD – all kata values

```
#! /bin/sh

testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }
testShouldReturn3for4() { _shouldReturnNumberForPosition 3 4; }
testShouldReturn5for5() { _shouldReturnNumberForPosition 5 5; }
testShouldReturn8for6() { _shouldReturnNumberForPosition 8 6; }
testShouldReturn13for7() { _shouldReturnNumberForPosition 13 7; }
testShouldReturn21for8() { _shouldReturnNumberForPosition 21 8; }
testShouldReturn34for9() { _shouldReturnNumberForPosition 34 9; }

_shouldReturnNumberForPosition() {
  fibonacciNumber="${1}"
  position="${2}"

  result=`fibonacci "${2}"`
  assertEquals "${1}" "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 28L, 906C written                    2,0-1         All
```

```
# fibonacci main code
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -le 1 ]; then
    echo "${position}"
    return
  fi

  previous=$( fibonacci $((position - 2)) )
  second_previous=$( fibonacci $((position - 1)) )

  echo $(( previous + second_previous ))
}
~
~
~
~
~
~
~
~
"fibonacci.inc" 17L, 343C written
```

```
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2
testShouldReturn2for3
testShouldReturn3for4
testShouldReturn5for5
testShouldReturn8for6
testShouldReturn13for7
testShouldReturn21for8
testShouldReturn34for9

Ran 10 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Bash TDD

Done!

```
fibonacci() {
  position="${1}"

  if [ "${position}" -le 1 ]; then
    echo "${position}"
    return
  fi

  previous=$( fibonacci $((position - 2)) )
  second_previous=$( fibonacci $((position - 1)) )

  echo $(( previous + second_previous ))
}
```

Fun stuff

How about:

- Providing an alternative faster implementation

Fun stuff

How about:

- Providing an alternative faster implementation
- Some refactoring: rename function fibonacci to fibonucci

Fun stuff

How about:

- Providing an alternative faster implementation
- Some refactoring: rename function fibonacci to fibonucci
- Using function name to provide testing parameters (simulate csv test input)

Fun stuff – faster implementation

```
#! /bin/sh

testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }
testShouldReturn3for4() { _shouldReturnNumberForPosition 3 4; }
testShouldReturn5for5() { _shouldReturnNumberForPosition 5 5; }
testShouldReturn8for6() { _shouldReturnNumberForPosition 8 6; }
testShouldReturn13for7() { _shouldReturnNumberForPosition 13 7; }
testShouldReturn21for8() { _shouldReturnNumberForPosition 21 8; }
testShouldReturn34for9() { _shouldReturnNumberForPosition 34 9; }

_shouldReturnNumberForPosition() {
  fibonacciNumber="${1}"
  position="${2}"

  result=`fibonacci "${2}"`
  assertEquals "${1}" "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci${ALTERNATE}.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 28L, 918C written                    24,27         All
```

```
# fibonacci non recursive faster implementation
# input: int position
# output: fibonacci number for the given position

fibonacci() {
  position="${1}"

  if [ "${position}" -le 1 ]; then
    echo "${position}"
    return
  fi

  a=0
  b=1
  for i in $(seq 2 ${position}); do
     c=$(( a + b ))

     if [ $a -gt $b ]; then
       b=$c
     else
       a=$c
     fi
  done

  echo $c
"fibonacci_fast.inc" 26L, 389C written
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ALTERNATE=_fast ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2
testShouldReturn2for3
testShouldReturn3for4
testShouldReturn5for5
testShouldReturn8for6
testShouldReturn13for7
testShouldReturn21for8
testShouldReturn34for9

Ran 10 tests.

OK
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$
```

Fun stuff – rename fibonacci to fibonucci

```
#! /bin/sh

testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }
testShouldReturn3for4() { _shouldReturnNumberForPosition 3 4; }
testShouldReturn5for5() { _shouldReturnNumberForPosition 5 5; }
testShouldReturn8for6() { _shouldReturnNumberForPosition 8 6; }
testShouldReturn13for7() { _shouldReturnNumberForPosition 13 7; }
testShouldReturn21for8() { _shouldReturnNumberForPosition 21 8; }
testShouldReturn34for9() { _shouldReturnNumberForPosition 34 9; }

_shouldReturnNumberForPosition() {
  fibonacciNumber="${1}"
  position="${2}"

  result=`fibonucci "${2}"`
  assertEquals "${1}" "${result}"
}

oneTimeSetUp() {
  # Load include to test.
  . ./fibonacci${ALTERNATE}.inc
}

# Load and run shUnit2.
. ../shunit2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"fibonacci_test.sh" 28L, 918C                                 1,1           All
```

```
# fibonucci main code
# input: int position
# output: fibonucci number for the given position

fibonucci() {
  position="${1}"

  if [ "${position}" -le 1 ]; then
    echo "${position}"
    return
  fi

  previous=$( fibonucci $((position - 2)) )
  second_previous=$( fibonucci $((position - 1)) )

  echo $(( previous + second_previous ))
}
~
~
~
~
~
~
~
~
~
"fibonacci.inc" 17L, 343C                                                    1,1           All
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ sed -i 's/fibonacci /fibonucci /;s/fibonacci()/fibonucci()/' fibonacci*
eoc12000@ictvt286:~/Downloads/ttt/shunit2-master/examples$ ./fibonacci_test.sh
testShouldReturn0for0
testShouldReturn1for1
testShouldReturn1for2
testShouldReturn2for3
testShouldReturn3for4
testShouldReturn5for5
testShouldReturn8for6
testShouldReturn13for7
testShouldReturn21for8
testShouldReturn34for9

Ran 10 tests.

OK
```

Fun stuff – simulate csv test input

Changing

```
testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }
testShouldReturn3for4() { _shouldReturnNumberForPosition 3 4; }
testShouldReturn5for5() { _shouldReturnNumberForPosition 5 5; }
testShouldReturn8for6() { _shouldReturnNumberForPosition 8 6; }
testShouldReturn13for7() { _shouldReturnNumberForPosition 13 7; }
testShouldReturn21for8() { _shouldReturnNumberForPosition 21 8; }
testShouldReturn34for9() { _shouldReturnNumberForPosition 34 9; }
```

Fun stuff – simulate csv test input

Changing

```
testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }
testShouldReturn3for4() { _shouldReturnNumberForPosition 3 4; }
testShouldReturn5for5() { _shouldReturnNumberForPosition 5 5; }
testShouldReturn8for6() { _shouldReturnNumberForPosition 8 6; }
testShouldReturn13for7() { _shouldReturnNumberForPosition 13 7; }
testShouldReturn21for8() { _shouldReturnNumberForPosition 21 8; }
testShouldReturn34for9() { _shouldReturnNumberForPosition 34 9; }
```

To

```
testShouldReturn_0_for_0_() { _shouldReturnNumberForPosition; }
testShouldReturn_1_for_1_() { _shouldReturnNumberForPosition; }
testShouldReturn_1_for_2_() { _shouldReturnNumberForPosition; }
testShouldReturn_2_for_3_() { _shouldReturnNumberForPosition; }
testShouldReturn_3_for_4_() { _shouldReturnNumberForPosition; }
testShouldReturn_5_for_5_() { _shouldReturnNumberForPosition; }
testShouldReturn_8_for_6_() { _shouldReturnNumberForPosition; }
testShouldReturn_13_for_7_() { _shouldReturnNumberForPosition; }
testShouldReturn_21_for_8_() { _shouldReturnNumberForPosition; }
testShouldReturn_34_for_9_() { _shouldReturnNumberForPosition; }
```

Fun stuff – simulate csv test input

Changing

```
testShouldReturn0for0() { _shouldReturnNumberForPosition 0 0; }
testShouldReturn1for1() { _shouldReturnNumberForPosition 1 1; }
testShouldReturn1for2() { _shouldReturnNumberForPosition 1 2; }
testShouldReturn2for3() { _shouldReturnNumberForPosition 2 3; }
testShouldReturn3for4() { _shouldReturnNumberForPosition 3 4; }
testShouldReturn5for5() { _shouldReturnNumberForPosition 5 5; }
testShouldReturn8for6() { _shouldReturnNumberForPosition 8 6; }
testShouldReturn13for7() { _shouldReturnNumberForPosition 13 7; }
testShouldReturn21for8() { _shouldReturnNumberForPosition 21 8; }
testShouldReturn34for9() { _shouldReturnNumberForPosition 34 9; }
```

To

Thank you for listening

Special thanks to Alcor Accademy for the «Classic TDD – nth Fibonacci» slide.

About me

Thank you for listening

Special thanks to Alcor Accademy for the «Classic TDD – nth Fibonacci» slide.

About me