

CIS 120
Lab 7: Testing and JUnit

October 21–22, 2009

Outline

Administrivia

Testing

JUnit

JUnit practice

- ▶ Attendance
- ▶ SpellCheck Homework? (due Wednesday 10/28)



Survey

Please take 7 minutes to help us improve CIS 110/120!
Survey link on the course web page.

Outline

Administrivia

Testing

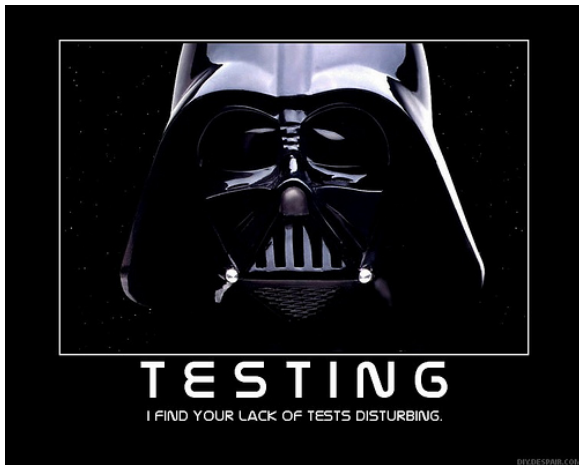
JUnit

JUnit practice

Why write automated tests?



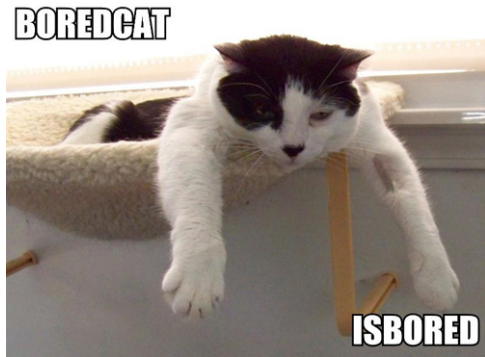
Why write automated tests?



Just because we're told to?

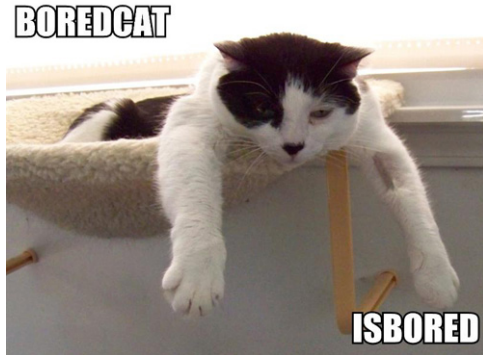
Why **not** write tests?

- ▶ Boring



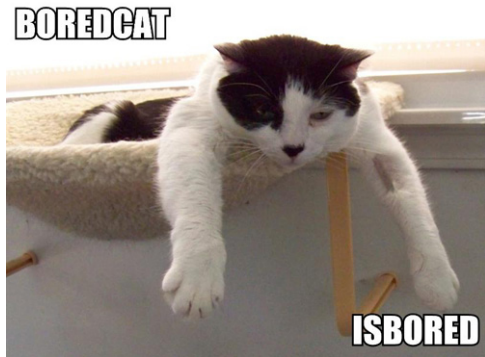
Why **not** write tests?

- ▶ Boring
- ▶ Waste of time



Why **not** write tests?

- ▶ Boring
- ▶ Waste of time
- ▶ Only for losers who aren't smart enough to write good code



The reality

Boring?

The reality

~~Boring?~~

- ▶ Coming up with good tests is a fun puzzle.
- ▶ Forces you to think deeply about the problem.

The reality

Waste of time?

The reality

~~Waste of time?~~

- ▶ Takes more time in the short term. . .
- ▶ . . .but saves a lot of time spent writing and debugging code!
- ▶ Helps communicate your intentions to others.

The reality

Only for dumb losers?

The reality

~~Only for dumb losers?~~

Everyone makes mistakes!

Outline

Administrivia

Testing

JUnit

JUnit practice

JUnit

Simple Java framework for writing automated “unit tests”.

- ▶ A test is any method that starts with “test”
- ▶ Many “assert. . .” methods to help us express expected results
- ▶ Built-in Eclipse support

assert. . .

- ▶ ... Equals
- ▶ ... True
- ▶ ... Null
- ▶ ... NotNull
- ▶ ... Same
- ▶ ... NotSame

Demo!

JUnit + Eclipse demo

Further reading

JUnit provides many more tools for writing sophisticated tests!

You can read more at junit.org.

Outline

Administrivia

Testing

JUnit

JUnit practice

Sorting

```
public interface ISorter {  
    public int [] sort(int [] arr);  
}
```

- ▶ should return a new array with the exact same contents as the input array, but sorted from smallest to biggest;
- ▶ should not modify the original array;
- ▶ should never throw an exception.

Sorting

```
public interface ISorter {  
    public int [] sort(int [] arr);  
}
```

- ▶ should return a new array with the exact same contents as the input array, but sorted from smallest to biggest;
- ▶ should not modify the original array;
- ▶ should never throw an exception.

Your job: write a set of JUnit tests to identify correct ISorter implementations!

Suggested setup

```
private ISorter sorter;  
  
public void setUp() {  
    sorter = new Sorter();  
}  
  
... sorter.sort(...) ...
```

Testing your tests

Nine different ISorter implementations can be found at

`www.cis.upenn.edu/~cis120/labs/07-junit/*****.zip`

Only one is correct. How many do your tests catch?