På Google testar man på toaletten

Peter Karlsson Zetterberg

Sitt nära, några av exemplen har liten text!

Vem är Peter?

AMRA
PRECISION FOR DECISION

Life Science, Big Data, Image Processing…

Innehåll

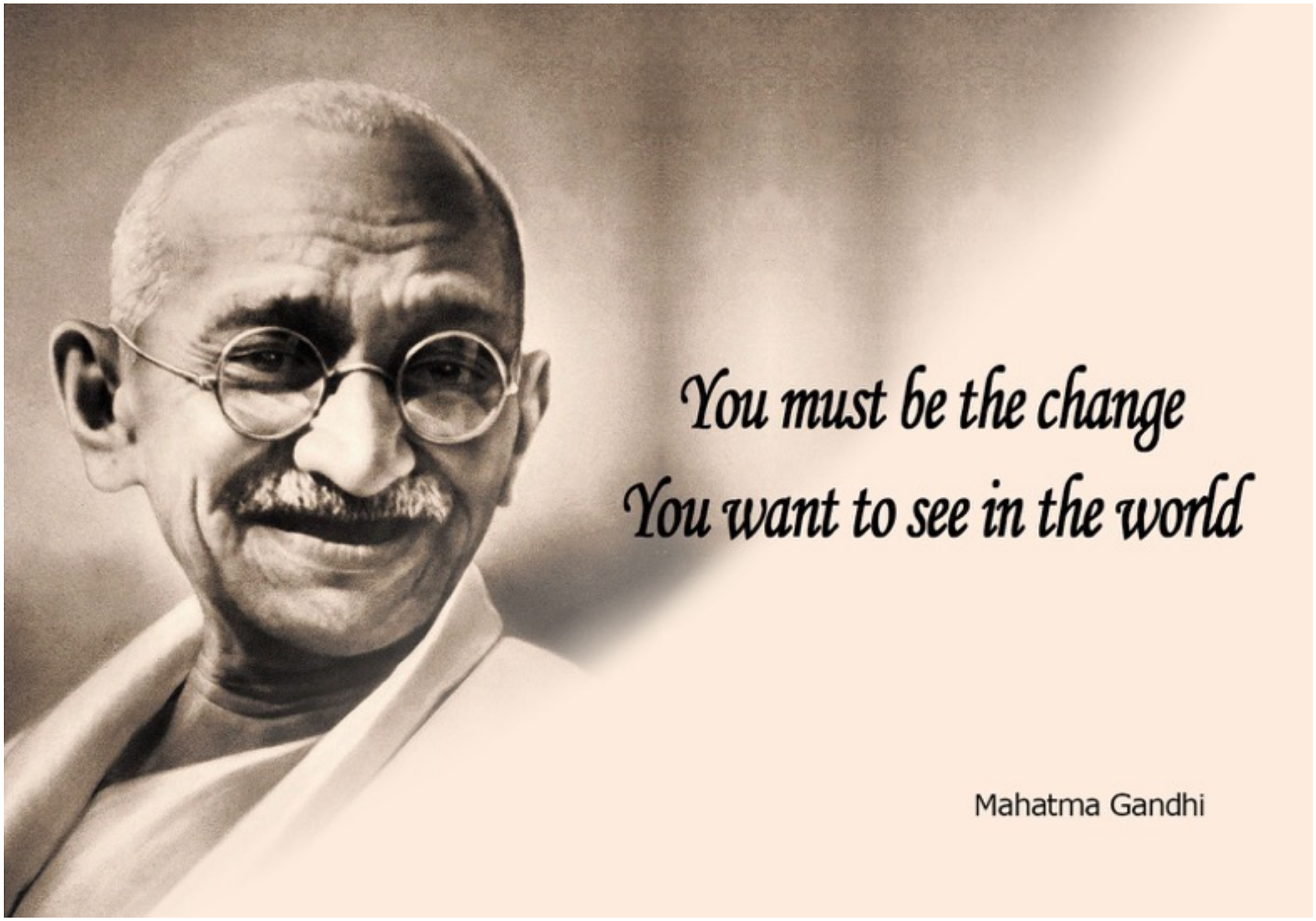
- Hur inspirera?
- Testa på toaletten!
- Råd på vägen

Hur inspirera?

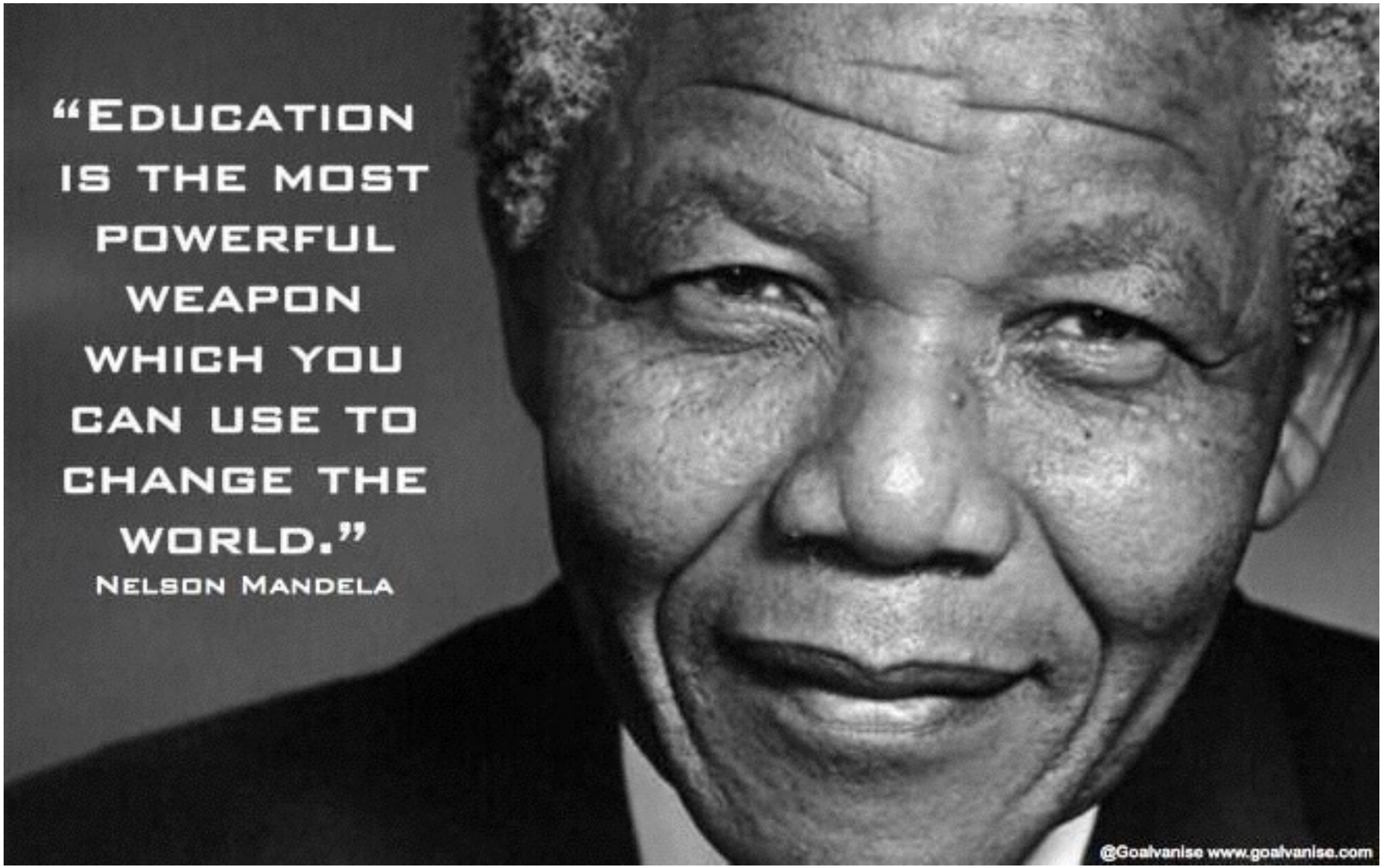
- Hur får jag mina kollegor intresserade av… ?

Tjat

Förebild


You must be the change
You want to see in the world
Mahatma Gandhi

Utbildning


"EDUCATION
IS THE MOST
POWERFUL
WEAPON
WHICH YOU
CAN USE TO
CHANGE THE
WORLD."
NELSON MANDELA
@Goalvanise www.goalvanise.com

Utmaningen

- Hur får vi Googles anställda mer intresserade av testning?
  - Nyhetsbrev en gång i veckan
  - Interna seminarier
  - Dela ut böcker

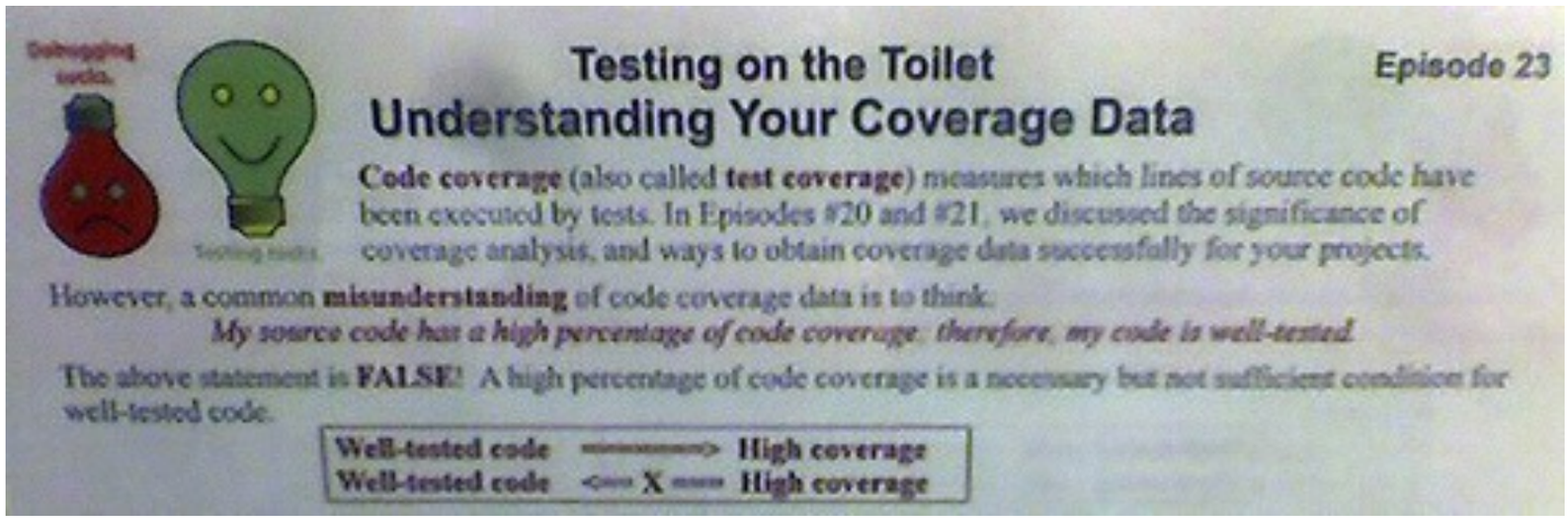
Vad funkade?

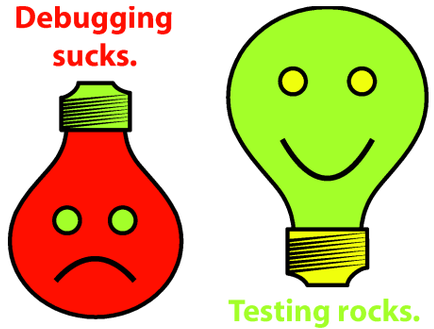
Det här…

Foto: Niklas Lindholm

Testa på toaletten

- Max en A4
- Hur skriva bra enhetstester?
- Hur använder jag ett test-verktyg?
- **Inspiration** istället för pekpinnar





Testing on the Toilet

Naming Unit Tests Responsibly

January 30, 2007

For a class, try having a corresponding set of test methods, where each one describes a responsibility of the object, with the first word implicitly the name of the class under test. For example, in Java:

```
class HtmlLinkRewriterTest ... {
  void testAppendsAdditionalParameterToUrlsInHrefAttributes(){…}
  void testDoesNotRewriteImageOrJavascriptLinks(){…}
  void testThrowsExceptionIfHrefContainsSessionId(){…}
  void testEncodesParameterValue(){…}
}
```

This can be read as:

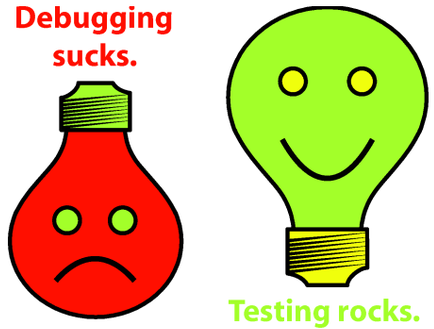
> HtmlLinkRewriter appends additional parameter to URLs in href attributes.
> HtmlLinkRewriter does not rewrite image or JavaScript links.
> HtmlLinkRewriter throws exception if href contains session ID.
> HtmlLinkRewriter encodes parameter value.

Benefits

The tests emphasizes the object's responsibilities (or features) rather than public methods and inputs/output. This makes it easier for future engineers who want to know what it does without having to delve into the code.

These naming conventions can help point out smells. For example, when it's hard to construct a sentence where the first word is the class under test, it suggests the test may be in the wrong place. And classes that are hard to describe in general often need to be broken down into smaller classes with clearer responsibilities.

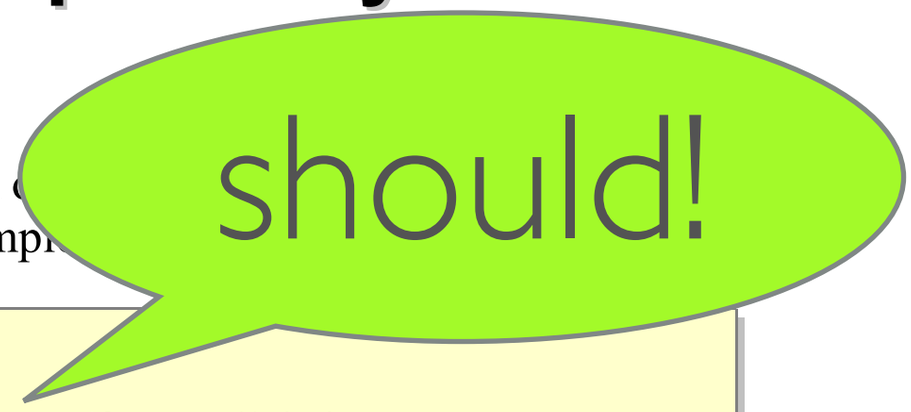
Additionally, tools can be used to help understand code quicker:



Testing on the Toilet

Naming Unit Tests Responsibly

January 30, 2007



For a class, try having a corresponding set of test methods, where each
with the first word implicitly the name of the class under test. For exampl

```
class HtmlLinkRewriterTest ... {
  void testAppendsAdditionalParameterToUrlsInHrefAttributes(){…}
  void testDoesNotRewriteImageOrJavascriptLinks(){…}
  void testThrowsExceptionIfHrefContainsSessionId(){…}
  void testEncodesParameterValue(){…}
}
```

This can be read as:

> HtmlLinkRewriter appends additional parameter to URLs in href attributes.
> HtmlLinkRewriter does not rewrite image or JavaScript links.
> HtmlLinkRewriter throws exception if href contains session ID.
> HtmlLinkRewriter encodes parameter value.

Benefits

The tests emphasizes the object's responsibilities (or features) rather than public methods and inputs/output. This makes it easier for future engineers who want to know what it does without having to delve into the code.

These naming conventions can help point out smells. For example, when it's hard to construct a sentence where the first word is the class under test, it suggests the test may be in the wrong place. And classes that are hard to describe in general often need to be broken down into smaller classes with clearer responsibilities.

Additionally, tools can be used to help understand code quicker:

Hur testa det här?

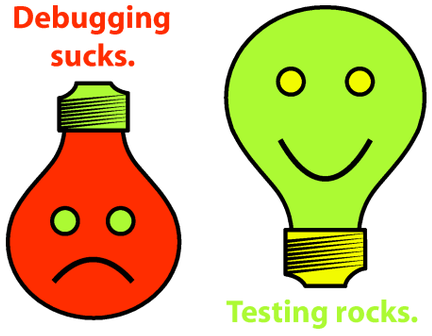
```
/** Return a Date object representing the start of the next
      minute from now */


public Date nextMinuteFromNow() {
   long nowAsMillis = System.currentTimeMillis();
   Date then = new Date(nowAsMillis + 60000);
   then.setSeconds(0);
   then.setMilliseconds(0);
   return then;
}
```

Problem

- Hur testar man de ovanliga fallen?
  - Byte av månad, byte av år, skottår, osv…
  - Hur håller man reda på hur snabbt själva testfallet exekverar?

En lösning

```
public Date minuteAfter(Date now) {
    Date then = new Date(now.getTime() + 60000);
    then.setSeconds(0);
    then.setMilliseconds(0);
    return then;
}
```



Testing on the Toilet

Time is Random

Apr. 3, 2008

How can a method be well tested when it's inputs can't be clearly identified? Consider this method in Java:

```
/** Return a date object representing the start of the next minute from now */
public Date nextMinuteFromNow() {
  long nowAsMillis = System.currentTimeMillis();
  Date then = new Date(nowAsMillis + 60000);
  then.setSeconds(0);
  then.setMilliseconds(0);
  return then;
}
```

There are two barriers to effectively testing this method:

1. There is no easy way to test corner cases; you're at the mercy of the system clock to supply input conditions.
2. When **`nextMinuteFromNow()`** returns, the time has changed. This means the test will not be an assertion, it will be a guess, and may generate low-frequency, hard-to-reproduce failures... **flakiness!** Class loading and garbage collection pauses, for example, can influence this.

Is `System.currentTimeMillis()` starting to look a bit like a random number provider? That's because it is! The current time is yet another source of *non-determinism*; the results of **`nextMinuteFromNow()`** cannot be easily determined from its inputs. Fortunately, this is easy to solve: make the *current time* an input parameter which you can control.

```
public Date minuteAfter(Date now) {
  Date then = new Date(now.getTime() + 60000);
  then.setSeconds(0);
  then.setMilliseconds(0);
  return then;
}

// Retain original functionality
```

Hur testa det här?

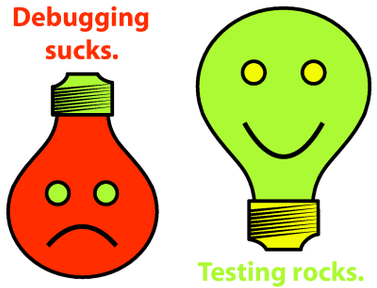
```
public class Client {

    public int process(Params params) {
        Server server = Server.getInstance();
        Data data = server.retrieveData(params);
        ...
    }
}
```

Problem

- Hur testa felhantering?
- Om servern är nere i tre dagar?
- Om server-anropet tar 10 minuter…

En lösning

```
public class Client {
    private final Server server;

    public Client(Server server) {
        this.server = server;
    }

    public int process(Params params){
        Data data = this.server.retrieveData(params);
        ...
    }
}
```



Testing on the Toilet

Using Dependancy Injection to Avoid Singletons

May 15, 2008

It's hard to test code that uses singletons. Typically, the code you want to test is coupled strongly with the singleton instance. You can't control the creation of the singleton object because often it is created in a static initializer or static method. As a result, you also can't mock out the behavior of that Singleton instance.

If changing the implementation of a singleton class is not an option, but changing the **client** of a singleton is, a simple refactoring can make it easier to test. Let's say you had a method that uses a `Server` as a singleton instance:

```
public class Client {
  public int process(Params params) {
    Server server = Server.getInstance();
    Data data = server.retrieveData(params);
    ...
  }
}
```

You can refactor `Client` to use **Dependency Injection** and avoid its use of the singleton pattern altogether. You have not lost any functionality, and have also not lost the requirement that only a singleton instance of `Server` must exist. The only difference is that instead of getting the `Server` instance from the static `getInstance` method, `Client` receives it in its constructor. You have made the class easier to test!

```
public class Client {
  private final Server server;

  public Client(Server server) {
    this.server = server;
  }

  public int process(Params params) {
    Data data = this.server.retrieveData(params);
    ...
  }
}
```

Mycket mer

- Hur tolka siffror för test-täckning?
- Hur många test är lagom många?
- Vad är ett bra enhetstest?
- Skillnaden på mocks, fakes och stubs?
- …

Avslutning

- Tjat, förebilder och utbildning kan funka

- Ibland krävs det något annat…