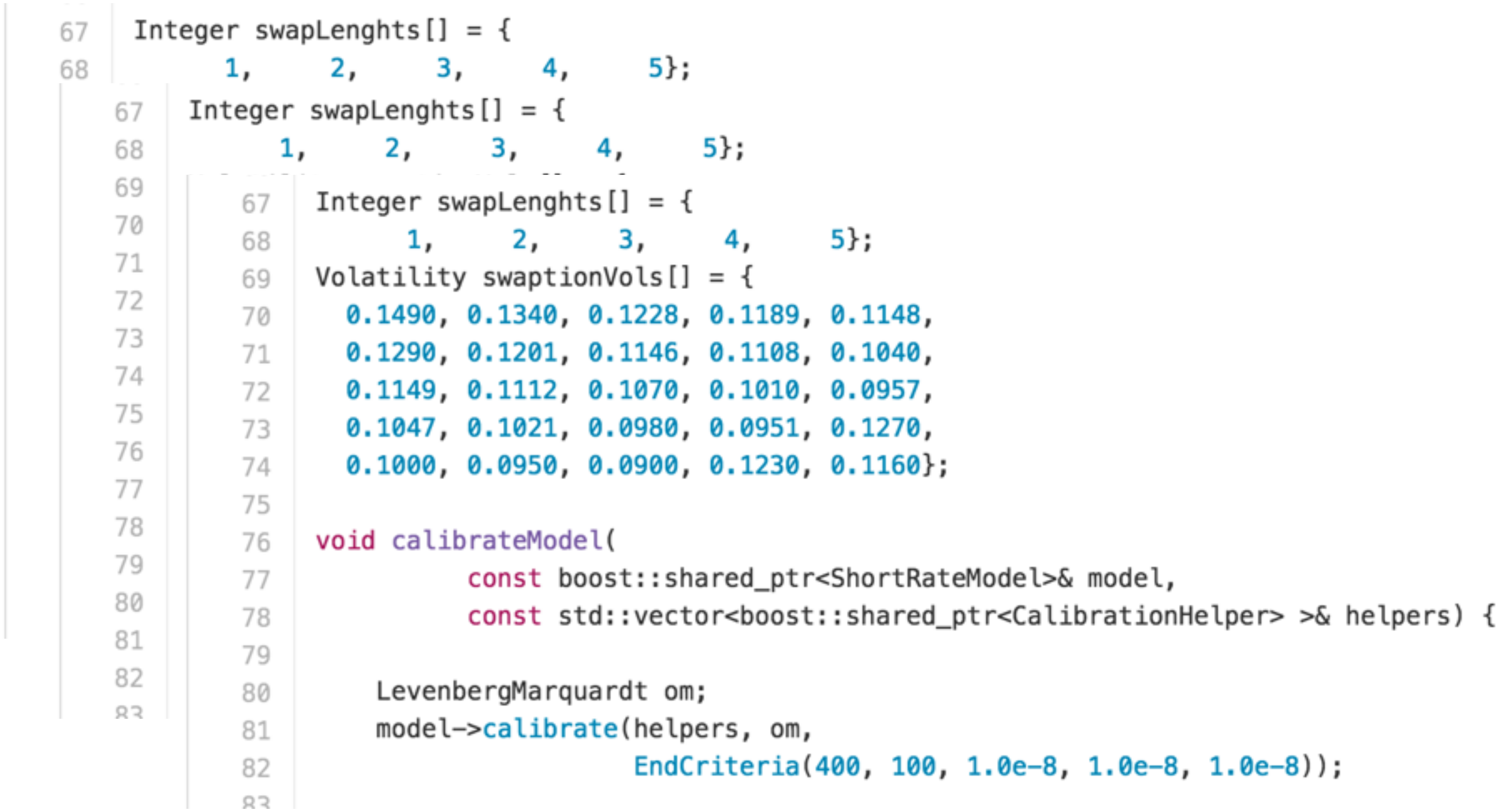
Strategic Code Deletion

Michael Feathers
R7K Research & Conveyance

(everyone wants to delete code)

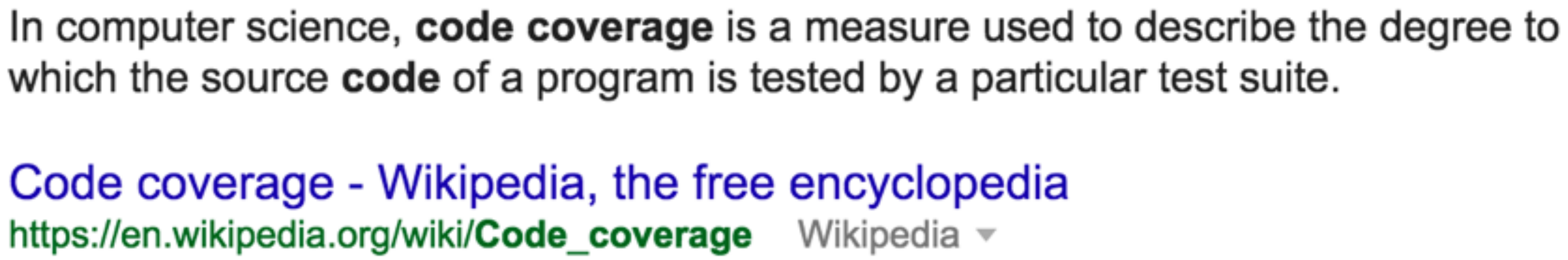
When Can We Delete Code?



```
Integer swapLenghts[] = {
      1,     2,     3,     4,     5};
```

```
Integer swapLenghts[] = {
      1,     2,     3,     4,     5};
```

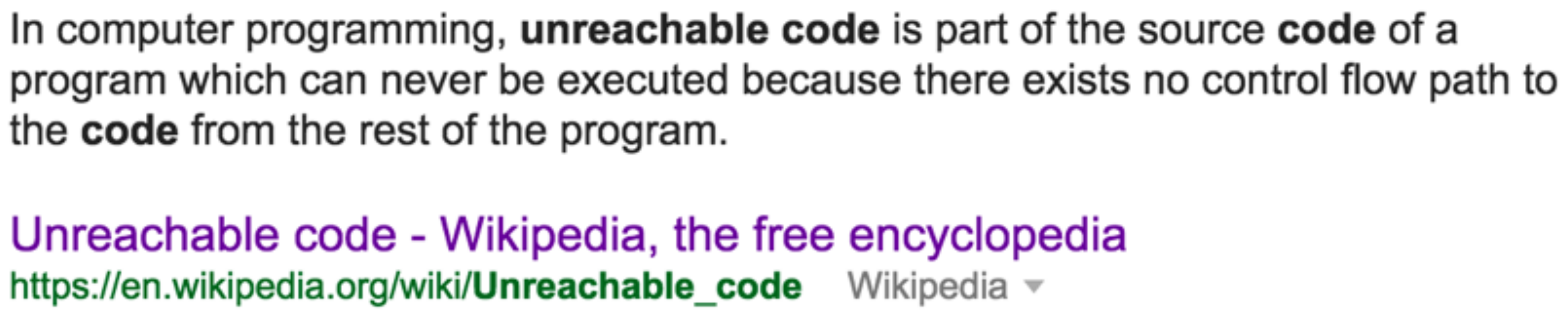
```
Integer swapLenghts[] = {
      1,     2,     3,     4,     5};
Volatility swaptionVols[] = {
  0.1490, 0.1340, 0.1228, 0.1189, 0.1148,
  0.1290, 0.1201, 0.1146, 0.1108, 0.1040,
  0.1149, 0.1112, 0.1070, 0.1010, 0.0957,
  0.1047, 0.1021, 0.0980, 0.0951, 0.1270,
  0.1000, 0.0950, 0.0900, 0.1230, 0.1160};

void calibrateModel(
          const boost::shared_ptr<ShortRateModel>& model,
          const std::vector<boost::shared_ptr<CalibrationHelper> >& helpers) {

    LevenbergMarquardt om;
    model->calibrate(helpers, om,
                     EndCriteria(400, 100, 1.0e-8, 1.0e-8, 1.0e-8));
```



In computer science, **code coverage** is a measure used to describe the degree to which the source **code** of a program is tested by a particular test suite.

Code coverage - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Code_coverage** Wikipedia ▾

Varieties of Useless Code



In computer programming, **unreachable code** is part of the source **code** of a program which can never be executed because there exists no control flow path to the **code** from the rest of the program.

Unreachable code - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Unreachable_code** Wikipedia

In computer programming, **dead code** is a section in the source **code** of a program which is executed but whose result is never used in any other computation. The execution of **dead code** wastes computation time and memory.

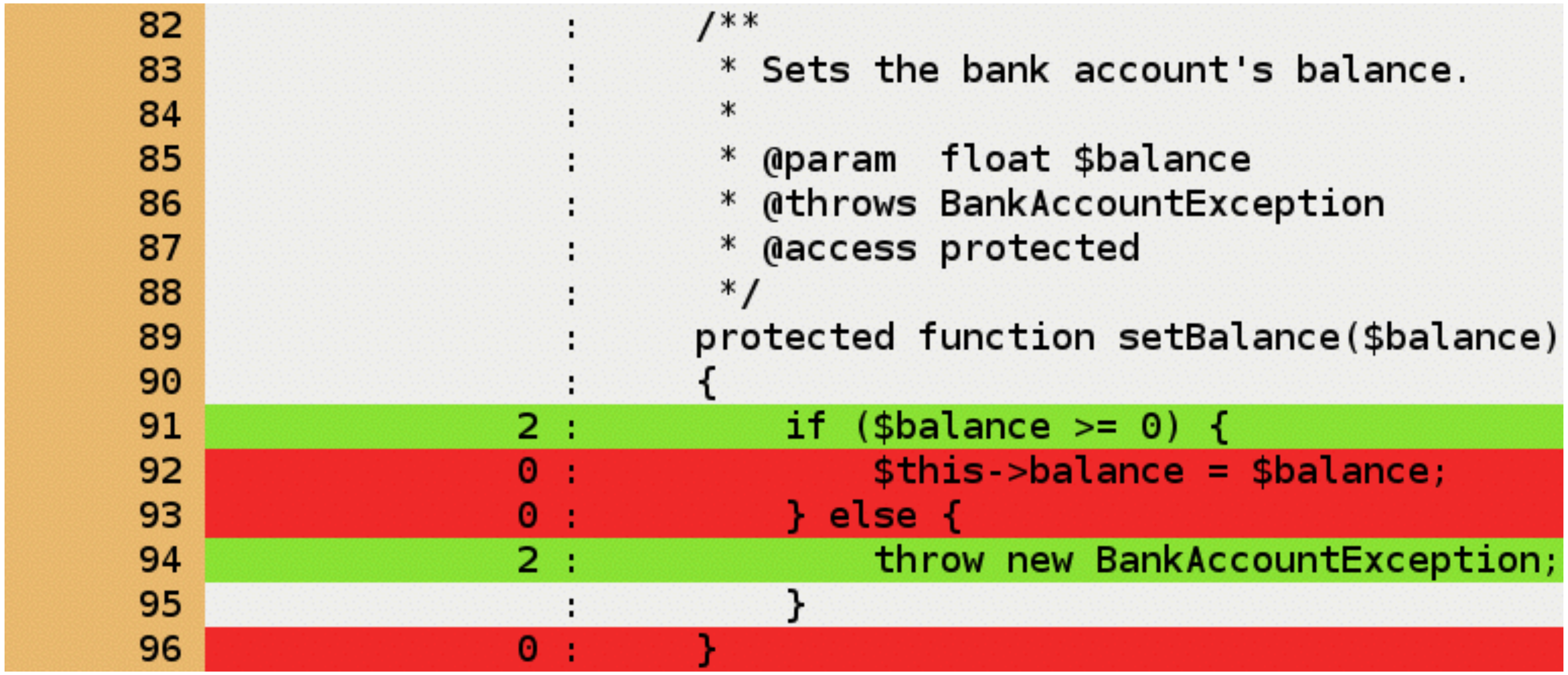
Dead code - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Dead_code** Wikipedia ▾

And another one..

Low Value code

No wikipedia entry (yet)   :-)

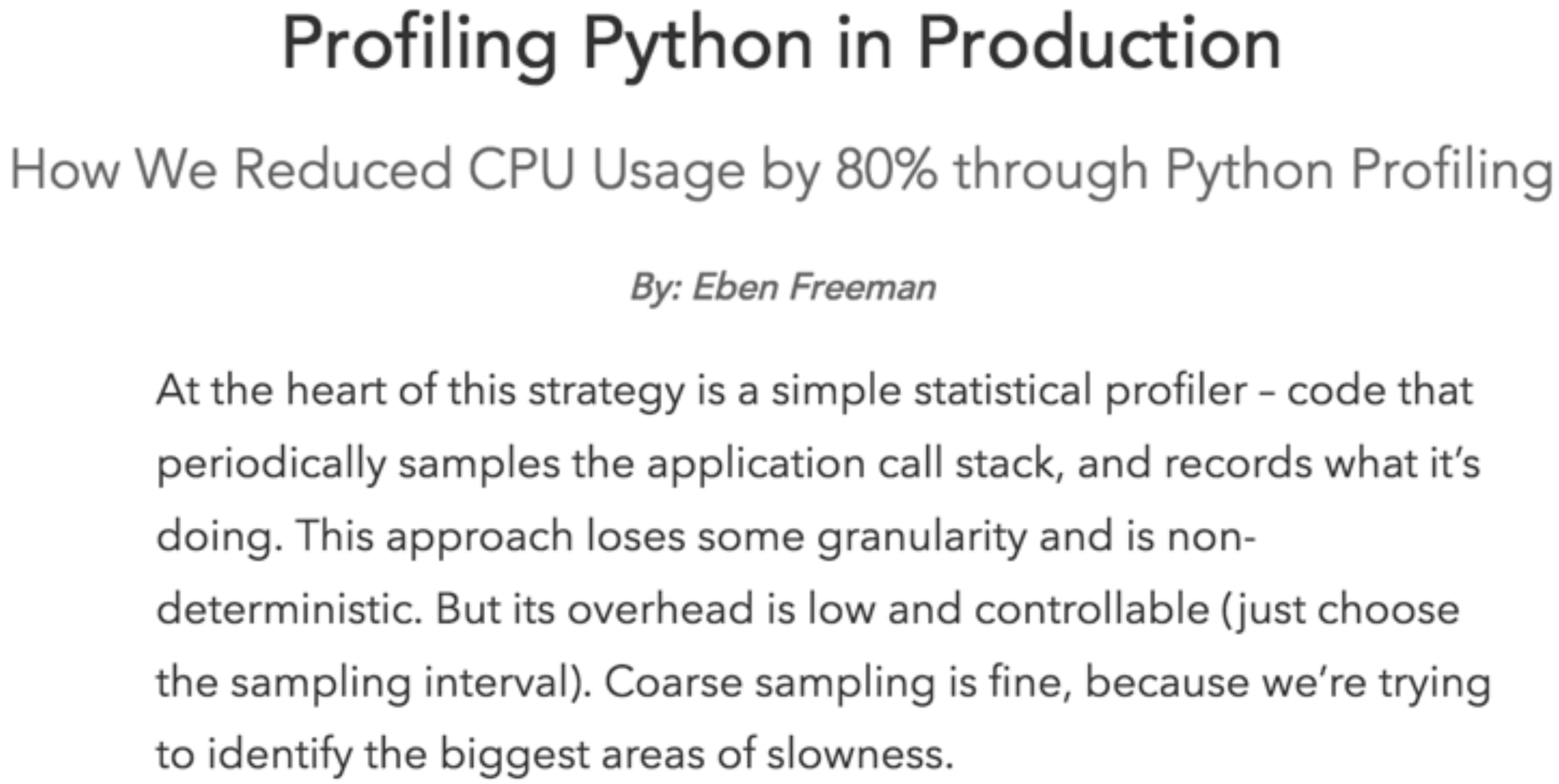
Coverage



```
  :     /**
  :      * Sets the bank account's balance.
  :      *
  :      * @param  float $balance
  :      * @throws BankAccountException
  :      * @access protected
  :      */
  :     protected function setBalance($balance)
  :     {
2 :         if ($balance >= 0) {
0 :             $this->balance = $balance;
0 :         } else {
2 :             throw new BankAccountException;
  :         }
0 :     }
```

gprof

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 33.34      0.02     0.02     7208     0.00     0.00  open
 16.67      0.03     0.01      244     0.04     0.12  offtime
 16.67      0.04     0.01        8     1.25     1.25  memccpy
 16.67      0.05     0.01        7     1.43     1.43  write
 16.67      0.06     0.01                             mcount
  0.00      0.06     0.00      236     0.00     0.00  tzset
  0.00      0.06     0.00      192     0.00     0.00  tolower
  0.00      0.06     0.00       47     0.00     0.00  strlen
  0.00      0.06     0.00       45     0.00     0.00  strchr
  0.00      0.06     0.00        1     0.00    50.00  main
  0.00      0.06     0.00        1     0.00     0.00  memcpy
  0.00      0.06     0.00        1     0.00    10.11  print
  0.00      0.06     0.00        1     0.00     0.00  profil
  0.00      0.06     0.00        1     0.00    50.00  report
...
```



Profiling Python in Production

How We Reduced CPU Usage by 80% through Python Profiling

By: Eben Freeman

At the heart of this strategy is a simple statistical profiler – code that periodically samples the application call stack, and records what it's doing. This approach loses some granularity and is non-deterministic. But its overhead is low and controllable (just choose the sampling interval). Coarse sampling is fine, because we're trying to identify the biggest areas of slowness.

https://nylas.com/blog/performance/

Mutation Testing

```
src2/java/main/org/jaxen/BaseXPath.java
 - changed source on line 691 (char index=24848) from 0 to 1


        return results.get( ?0 );
    }
}
```

Feature Probe

Feature Probe

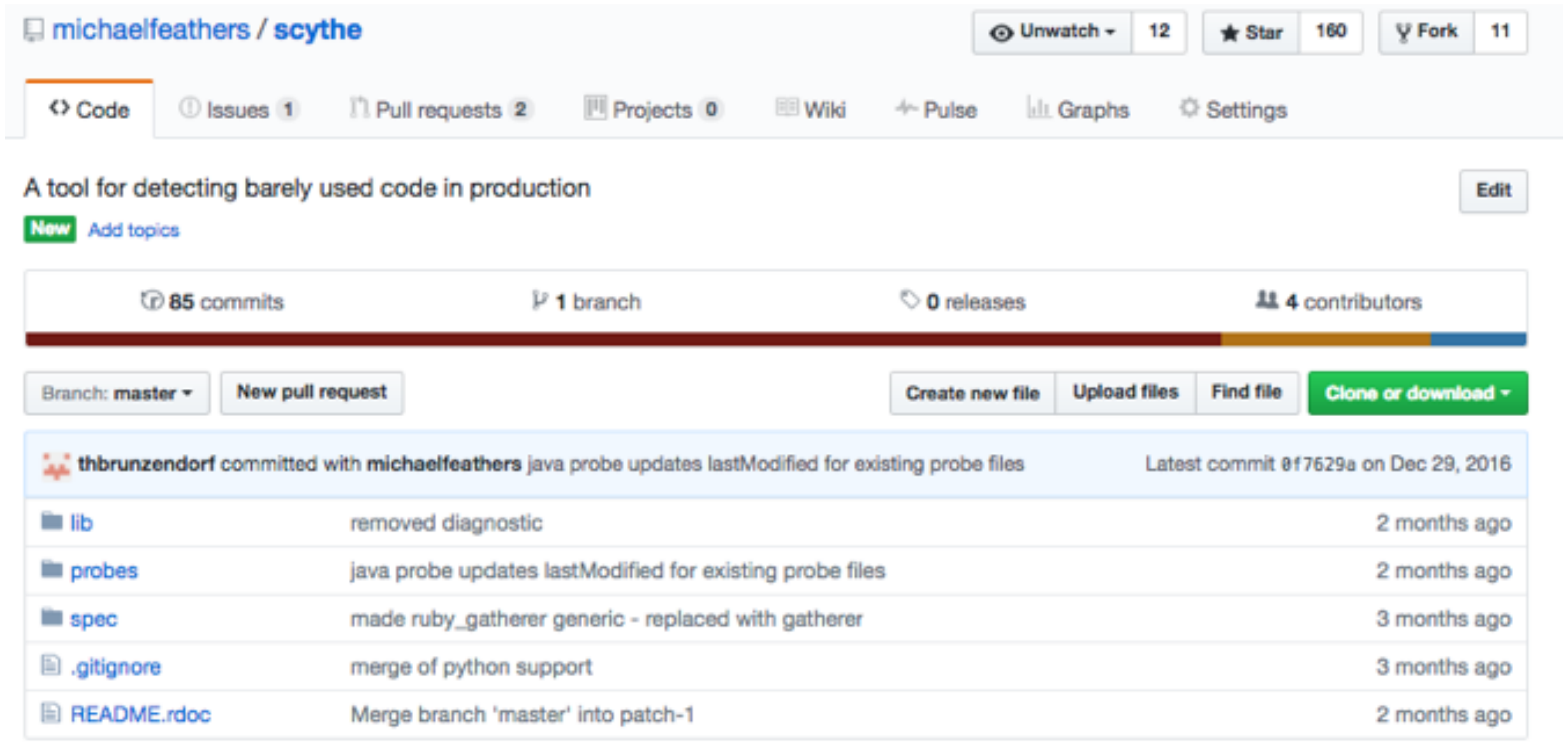
```
scythe_probe(“marker”);
```

Feature Probe

```
scythe_probe(“marker”);
```

- Record marker
- Periodically check for absence of marker

Feature Probe

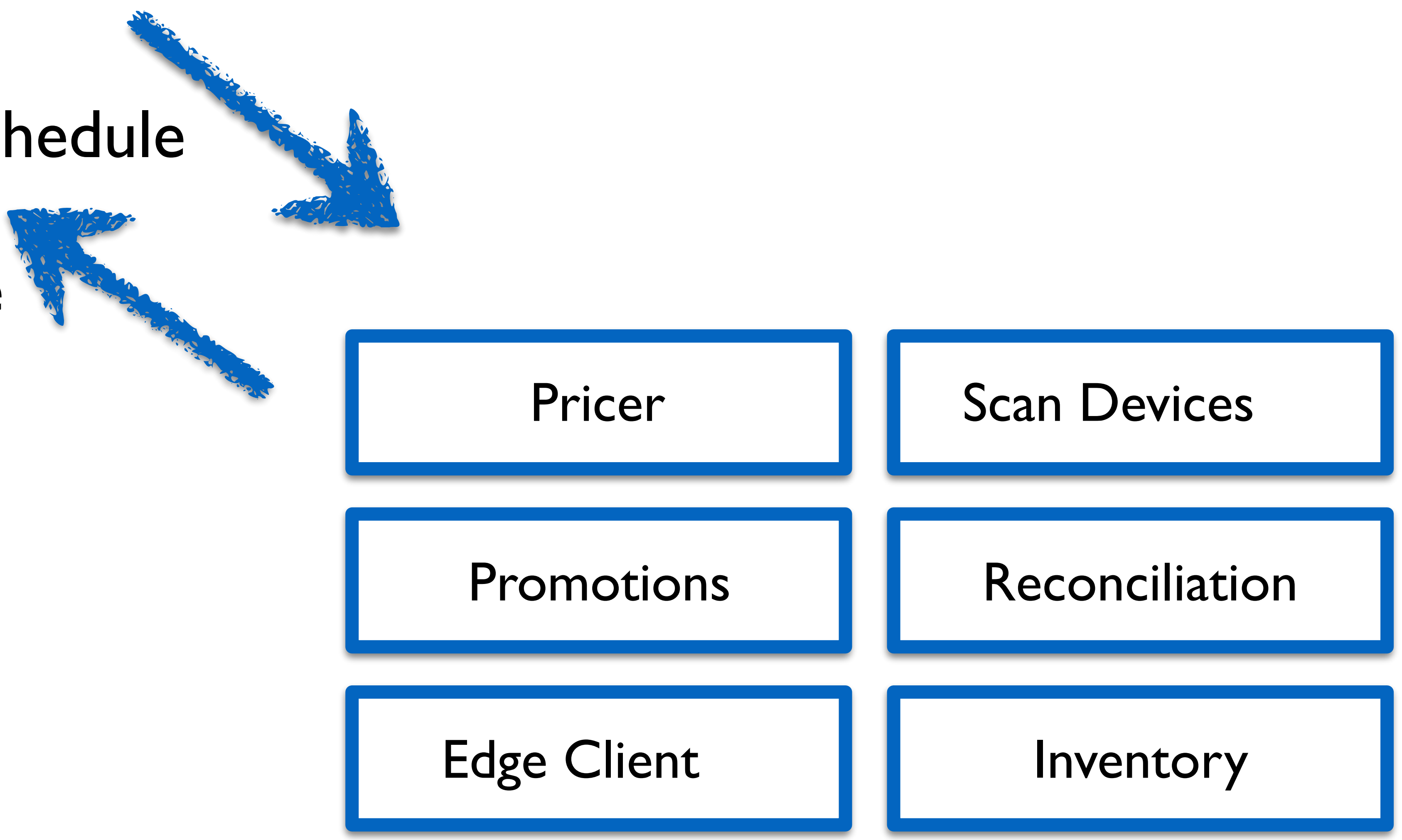

michaelfeathers / scythe

A tool for detecting barely used code in production

thbrunzendorf committed with michaelfeathers java probe updates lastModified for existing probe files — Latest commit 0f7629a on Dec 29, 2016

lib	removed diagnostic	2 months ago
probes	java probe updates lastModified for existing probe files	2 months ago
spec	made ruby_gatherer generic - replaced with gatherer	3 months ago
.gitignore	merge of python support	3 months ago
README.rdoc	Merge branch 'master' into patch-1	2 months ago

- Accept gift card
- Split charge (2 customers)
- Select home delivery
- Produce/Meat by weight
- Daily specials
- Weekly specials
- Restock perishable schedule
- Record invalid scans
- Change till procedure
- Cashier login
- Register reconcile



The Cost Case

Unreachable Code	Dead Code	Low Value Code
Compiler Errors / Warnings	Mutation Testing	Feature Probes / Stack Sampling / Coverage
Delete on Detect	Delete on Detect	Strategize

Strategy

- Delete *Unreachable* and *Dead Code* on Detection
- Disable *Low Value Code* at entry points
  - Verify in sample

Human in the Loop for Internal Pruning

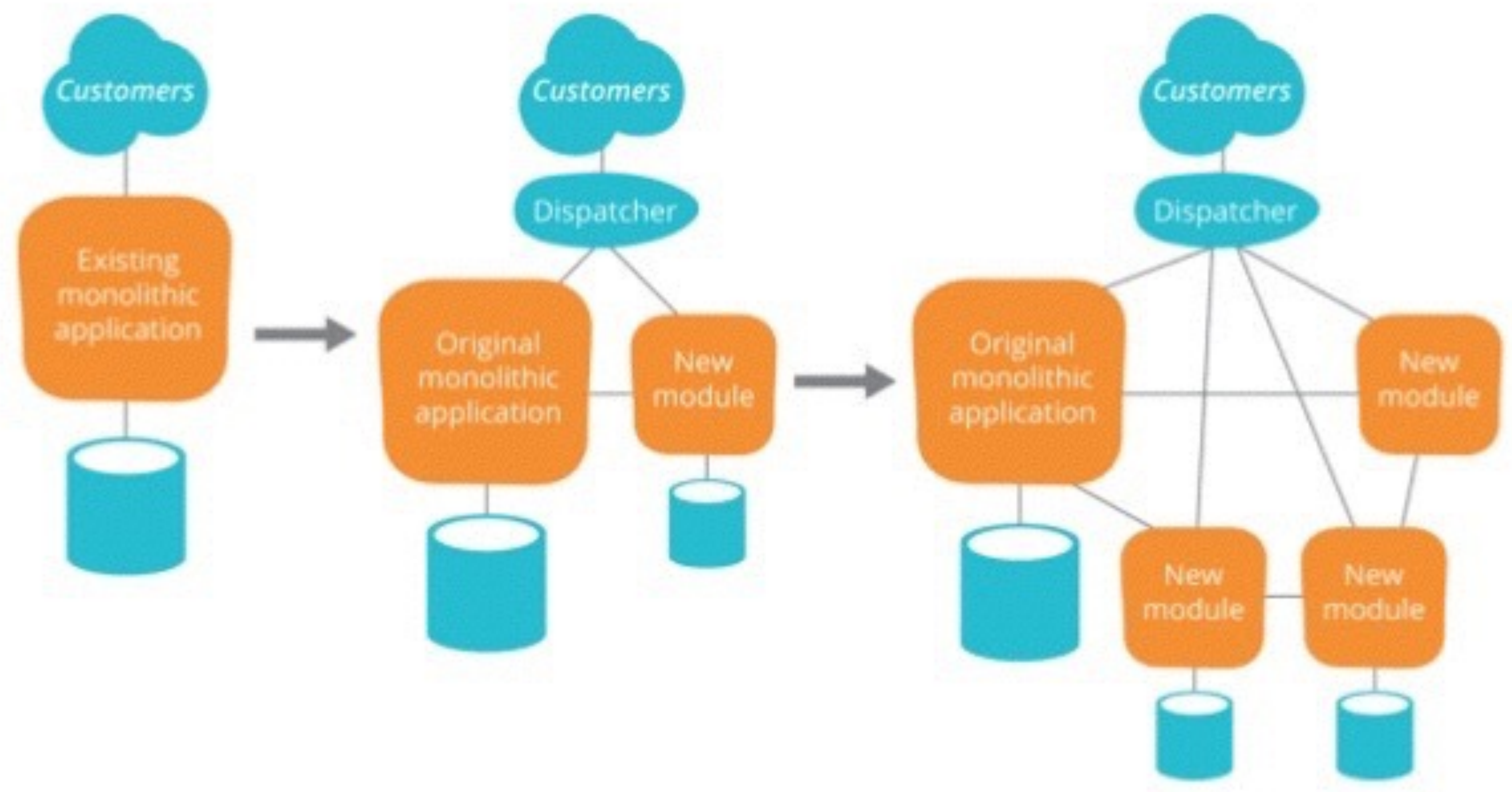
Systematic Rewrite

Most rewrites are motivated by technology or architectural change

Most rewrites are motivated by technology or architectural change

“Cleaning rewrites” should be focused and selective

Strangler Application

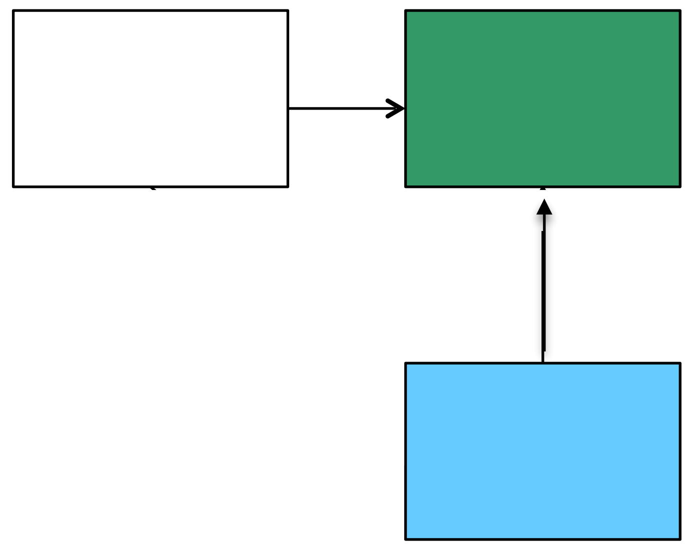

Customers
Existing monolithic application
Customers
Dispatcher
Original monolithic application
New module
Customers
Dispatcher
Original monolithic application
New module
New module
New module

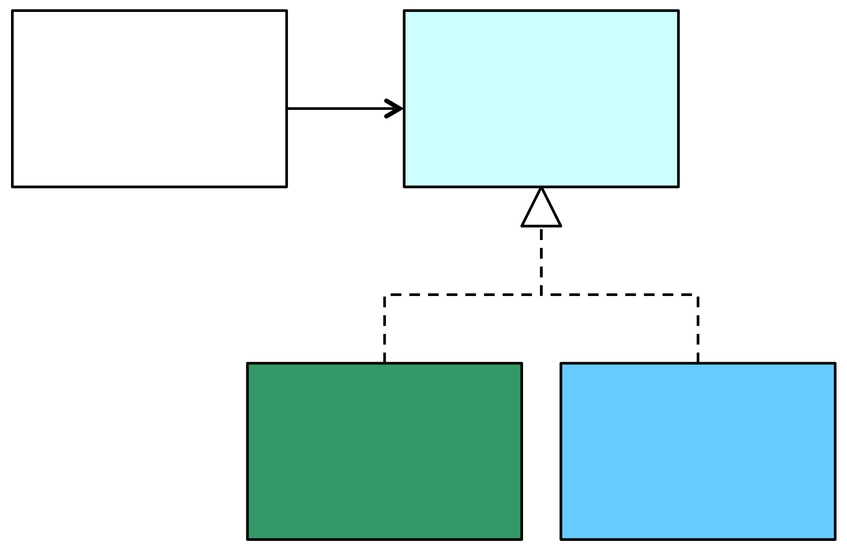
(from Jez Humble)

Key Actions for Rewrite

- Can you find all inputs and outputs?
- Are you reducing conditionality?
- Are you scoped for *Characterization Testing?*
- Can you run redundant?

Searching for Pinch Points





Characterization Testing

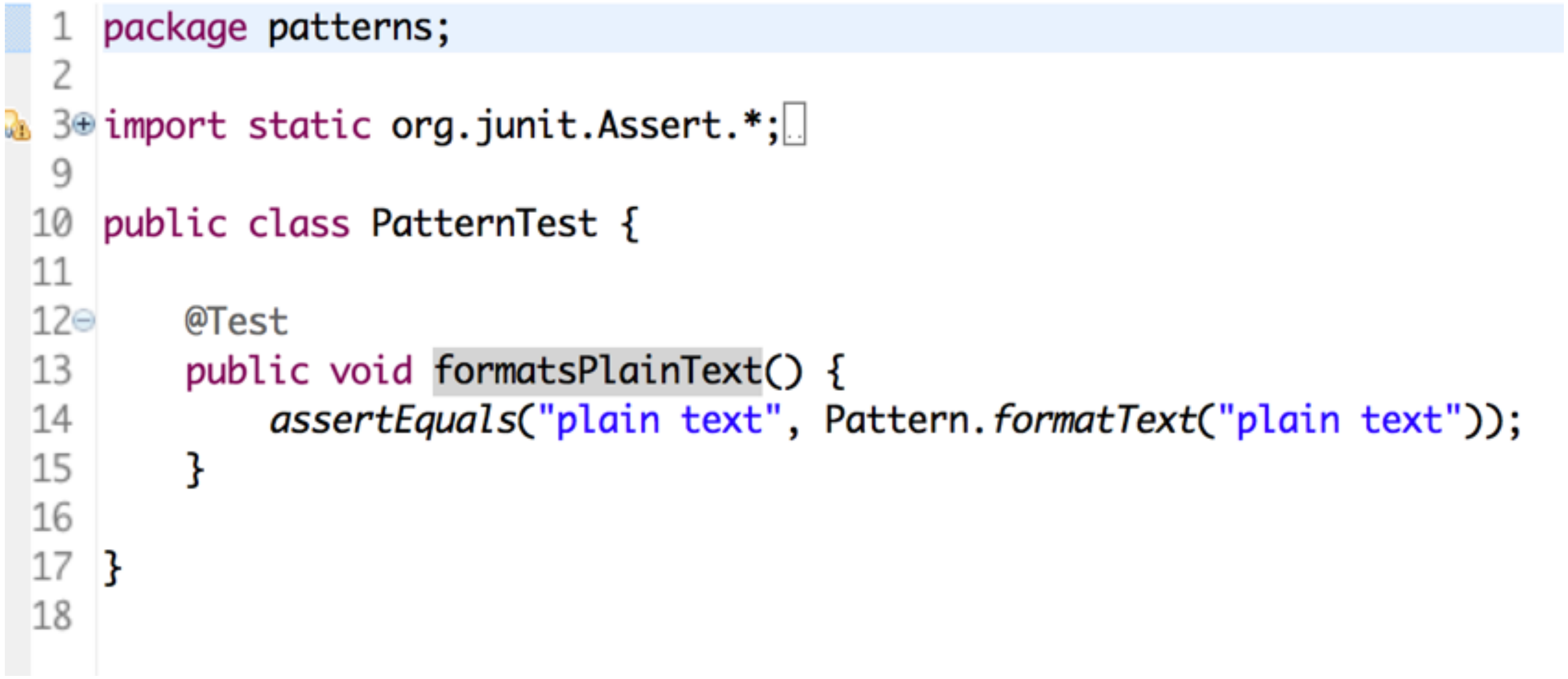
Tests you write to describe the current behavior of your system

Characterization Testing

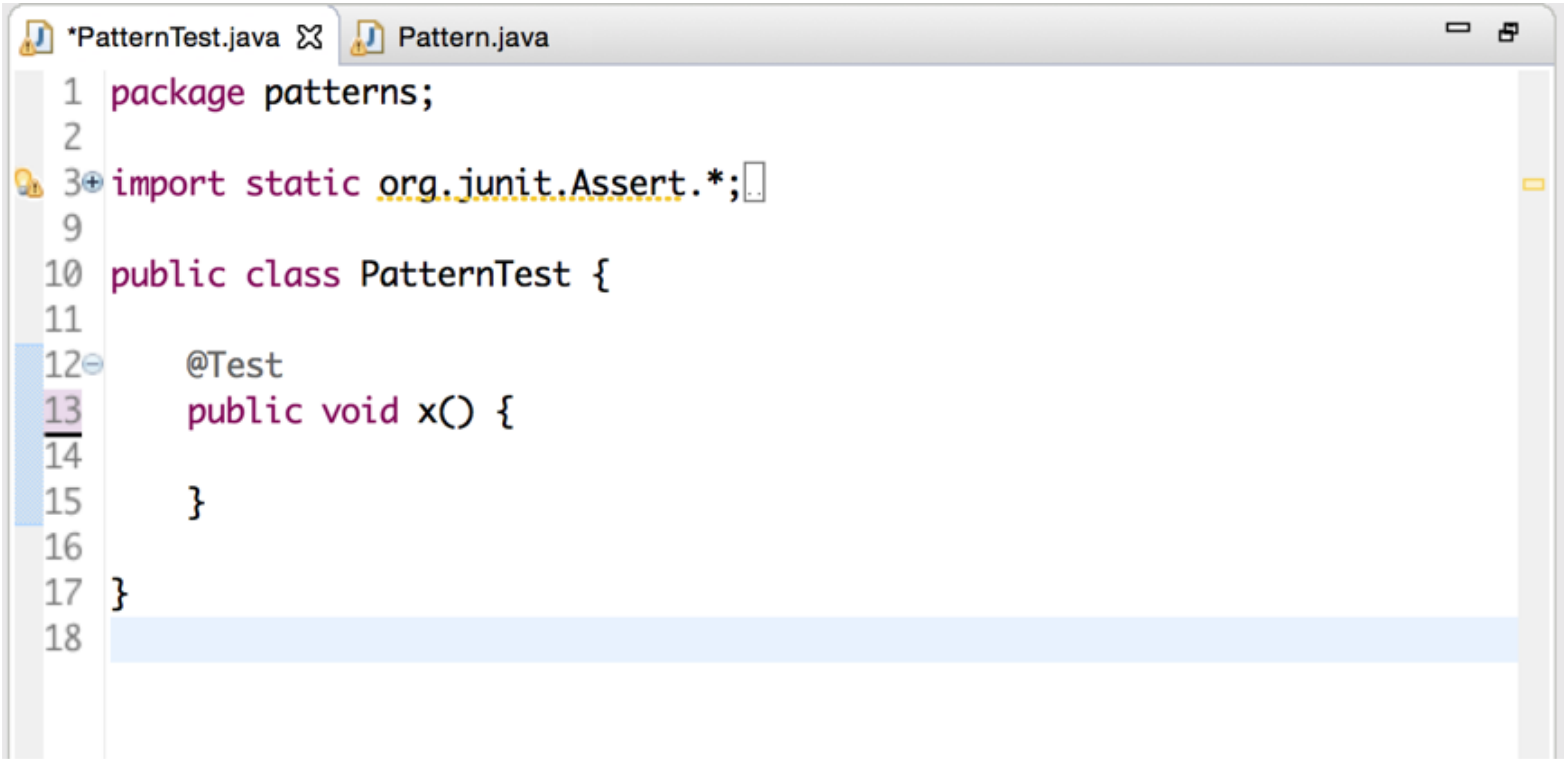
Tests you write to describe the current behavior of your system

How is this different from other testing?

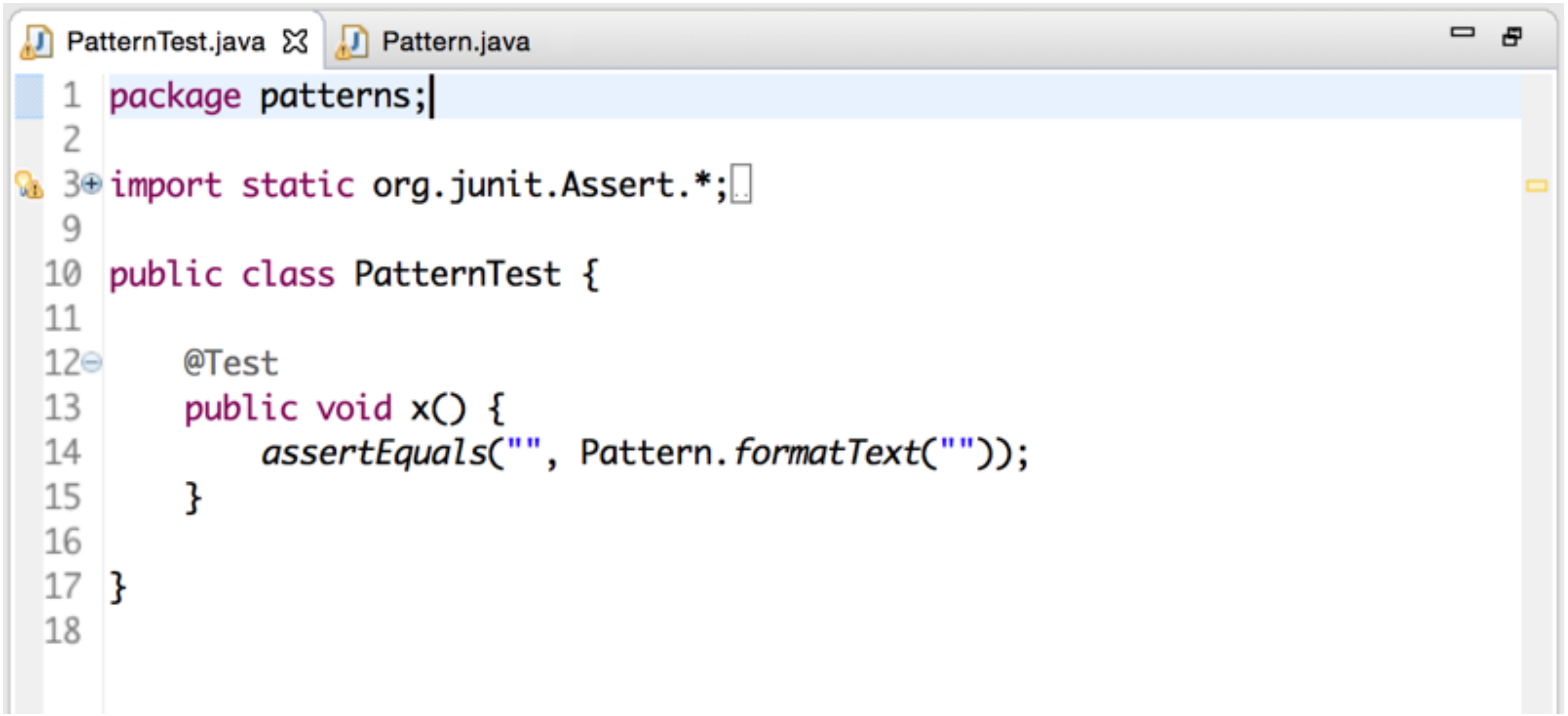
```
public static String formatText(String text)
{
    StringBuffer result = new StringBuffer();
    for (int n = 0; n < text.length(); ++n) {
        int c = text.charAt(n);
        if (c == '<') {
            while(n < text.length() && text.charAt(n) != '/' && text.charAt(n) != '>')
                n++;
            if (n < text.length() && text.charAt(n) == '/')
                n+=4;
            else
                n++;
        }
        if (n < text.length())
            result.append(text.charAt(n));
    }
    return new String(result);
}
```



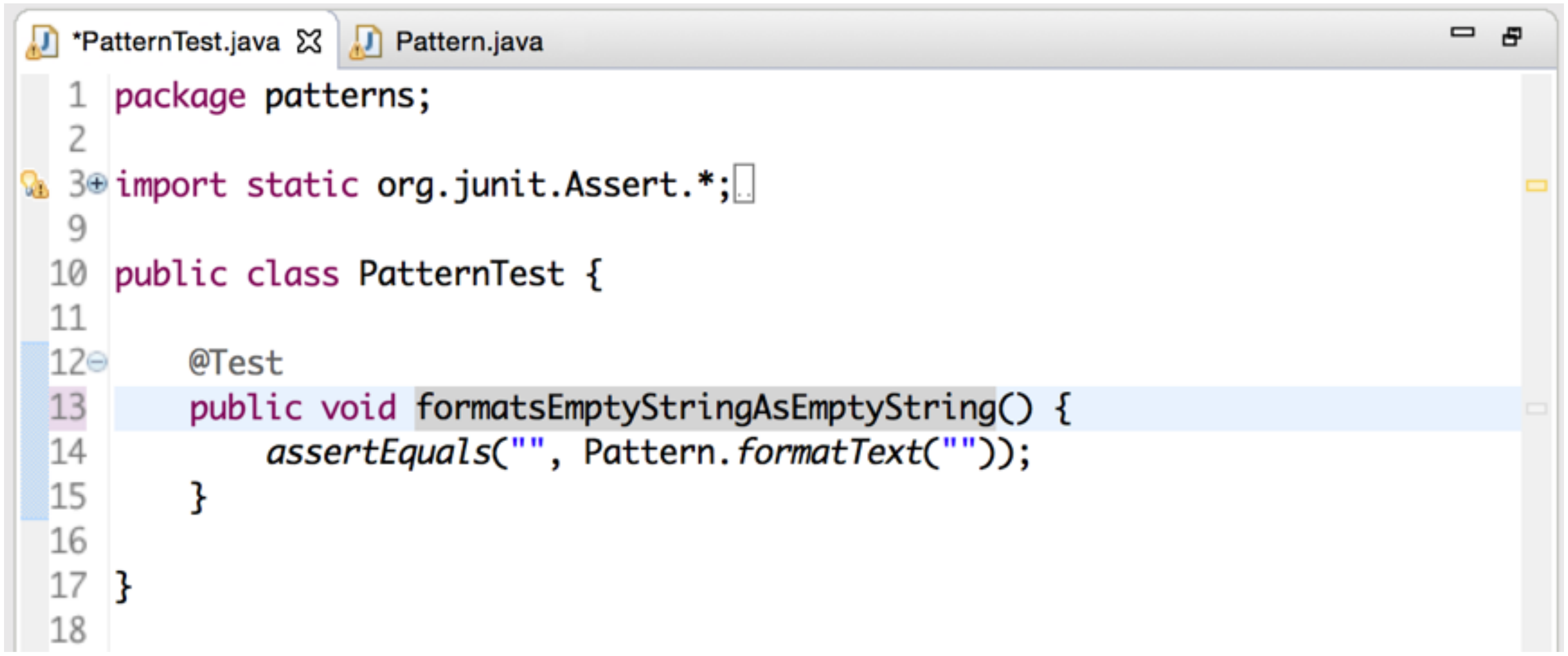
```
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void formatsPlainText() {
        assertEquals("plain text", Pattern.formatText("plain text"));
    }

}
```



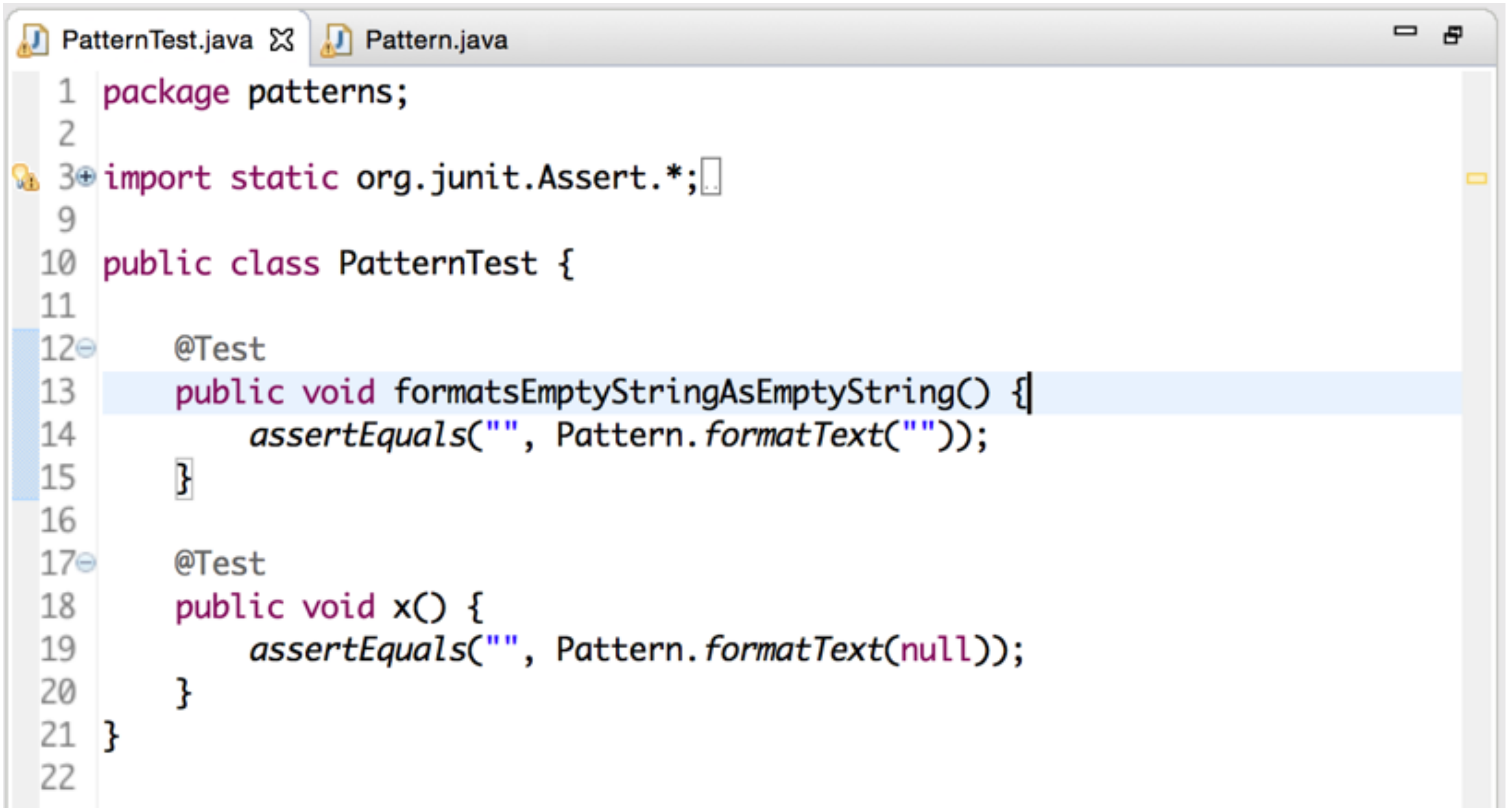
```
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void x() {

    }

}
```



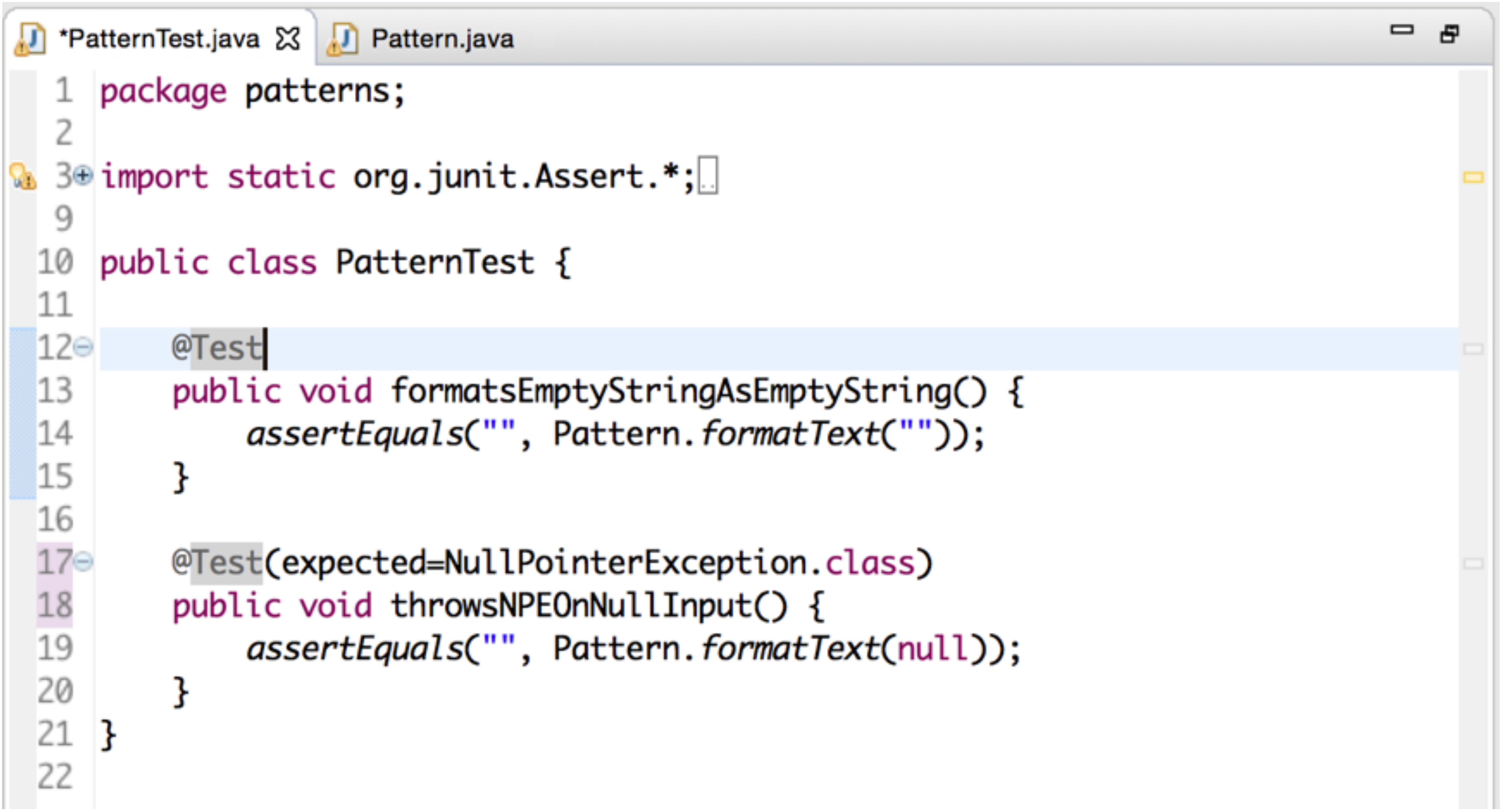
```
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void x() {
        assertEquals("", Pattern.formatText(""));
    }

}

```



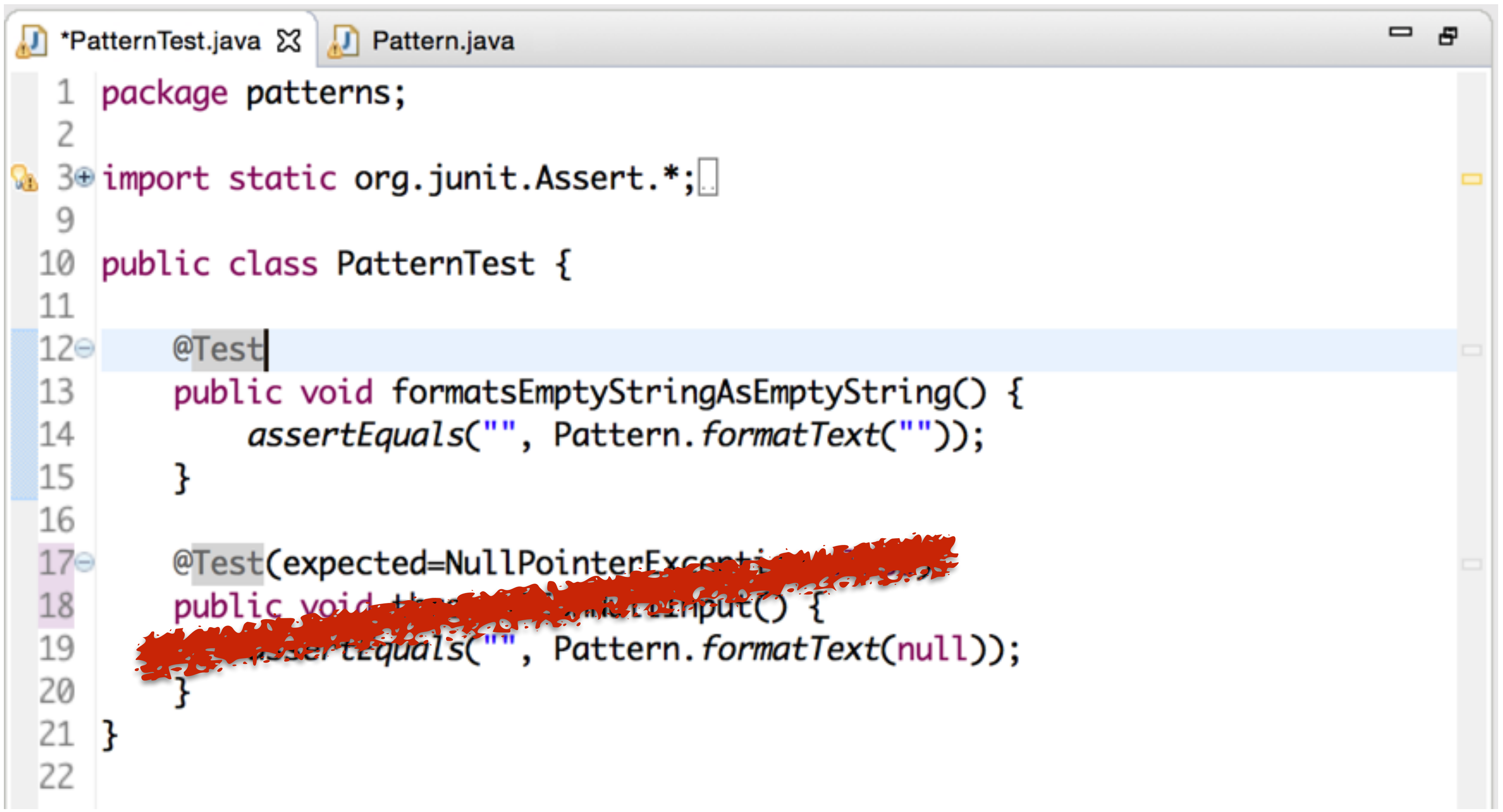
```
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void formatsEmptyStringAsEmptyString() {
        assertEquals("", Pattern.formatText(""));
    }

}
```



```java
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void formatsEmptyStringAsEmptyString() {
        assertEquals("", Pattern.formatText(""));
    }

    @Test
    public void x() {
        assertEquals("", Pattern.formatText(null));
    }
}
```


*PatternTest.java   Pattern.java

```java
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void formatsEmptyStringAsEmptyString() {
        assertEquals("", Pattern.formatText(""));
    }

    @Test(expected=NullPointerException.class)
    public void throwsNPEOnNullInput() {
        assertEquals("", Pattern.formatText(null));
    }
}

```

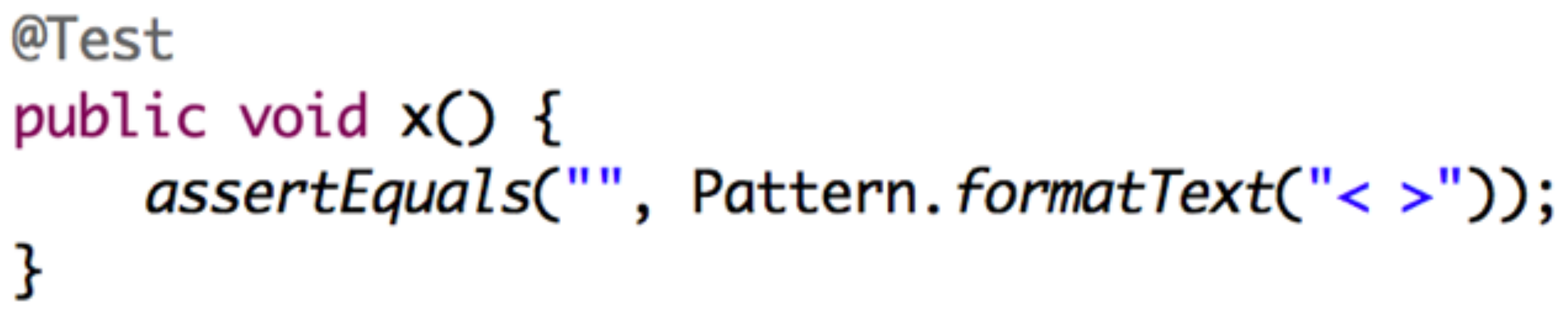


```
package patterns;

import static org.junit.Assert.*;

public class PatternTest {

    @Test
    public void formatsEmptyStringAsEmptyString() {
        assertEquals("", Pattern.formatText(""));
    }

    ~~@Test(expected=NullPointerExcept[illegible]~~
    ~~public void th[illegible]Input() {~~
        ~~[illegible]ertEquals("", Pattern.formatText(null));~~
    }
}

```

```java
@Test
public void formatsUndelimitedTextAsUnchanged() {
    assertEquals("plain text", Pattern.formatText("plain text"));
}
```

```java
public static String formatText(String text)
{
    StringBuffer result = new StringBuffer();
    for (int n = 0; n < text.length(); ++n) {
        int c = text.charAt(n);
        if (c == '<') {
            while(n < text.length() && text.charAt(n) != '/' && text.charAt(n) != '>')
                n++;
            if (n < text.length() && text.charAt(n) == '/')
                n+=4;
            else
                n++;
        }
        if (n < text.length())
            result.append(text.charAt(n));
    }
    return new String(result);
}
```

```
@Test
public void removesDelimiterUnderFormatting() {
    assertEquals("", Pattern.formatText("<>"));
}
```



```java
@Test
public void x() {
    assertEquals("", Pattern.formatText("< >"));
}
```

Is this a bug?

Simple Case Heuristics

Simple Case Heuristics

- Start with “x”

Simple Case Heuristics

- Start with “x”
- Use expressive (long) names

Simple Case Heuristics

- Start with “x”
- Use expressive (long) names
- Rename to tell a story

Simple Case Heuristics

- Start with "x"
- Use expressive (long) names
- Rename to tell a story
- Make the call on bugs

Simple Case Heuristics

- Start with “x”
- Use expressive (long) names
- Rename to tell a story
- Make the call on bugs
- Be curious!

```java
public static String formatText(String text)
{
    StringBuffer result = new StringBuffer();
    for (int n = 0; n < text.length(); ++n) {
        int c = text.charAt(n);
        if (c == '<') {
            while(n < text.length() && text.charAt(n) != '/' && text.charAt(n) != '>')
                n++;
            if (n < text.length() && text.charAt(n) == '/')
                n+=4;
            else
                n++;
        }
        if (n < text.length())
            result.append(text.charAt(n));
    }
    return new String(result);
}
```

What was our goal?

```
public class ResidentialAccount {

    public void charge(int gallons, LocalDate readingDate) {
        if (billingPlan.isMonthly()) {
            if (gallons < RESIDENTIAL_MIN) {
                balance += RESIDENTIAL_BASE;
            } else {
                balance += 1.2 * priceForGallons(gallons);
            }
            billingPlan.postReading(readingDate, gallons);
        }
    }
```

```
public class ResidentialAccount {

    public void charge(int gallons, LocalDate readingDate) {
        if (billingPlan.isMonthly()) {
            if (gallons < RESIDENTIAL_MIN) {
                balance += RESIDENTIAL_BASE;
            } else {
                balance += 1.2 * priceForGallons(gallons);
            }
            billingPlan.postReading(readingDate, gallons);
        }
    }
```

```java
public class ResidentialAccount {

    public void charge(int gallons, LocalDate readingDate) {
        if (billingPlan.isMonthly()) {
            if (gallons < RESIDENTIAL_MIN) {
                balance += RESIDENTIAL_BASE;
            } else {
                balance += 1.2 * priceForGallons(gallons);
            }
            billingPlan.postReading(readingDate, gallons);
        }
    }
```

✔ balance
✔ gallons
✘ RESIDENTIAL_MIN
✘ RESIDENTIAL_BASE

Tooling Varies

```java
public class ResidentialAccount {

    public double sensingBalance = 0;

    public void charge(int gallons, LocalDate readingDate) {
        if (billingPlan.isMonthly()) {
            if (gallons < RESIDENTIAL_MIN) {
                balance += RESIDENTIAL_BASE;
            } else {
                balance += 1.2 * priceForGallons(gallons);
            }
            sensingBalance = balance;
            billingPlan.postReading(readingDate, gallons);
        }
    }
```

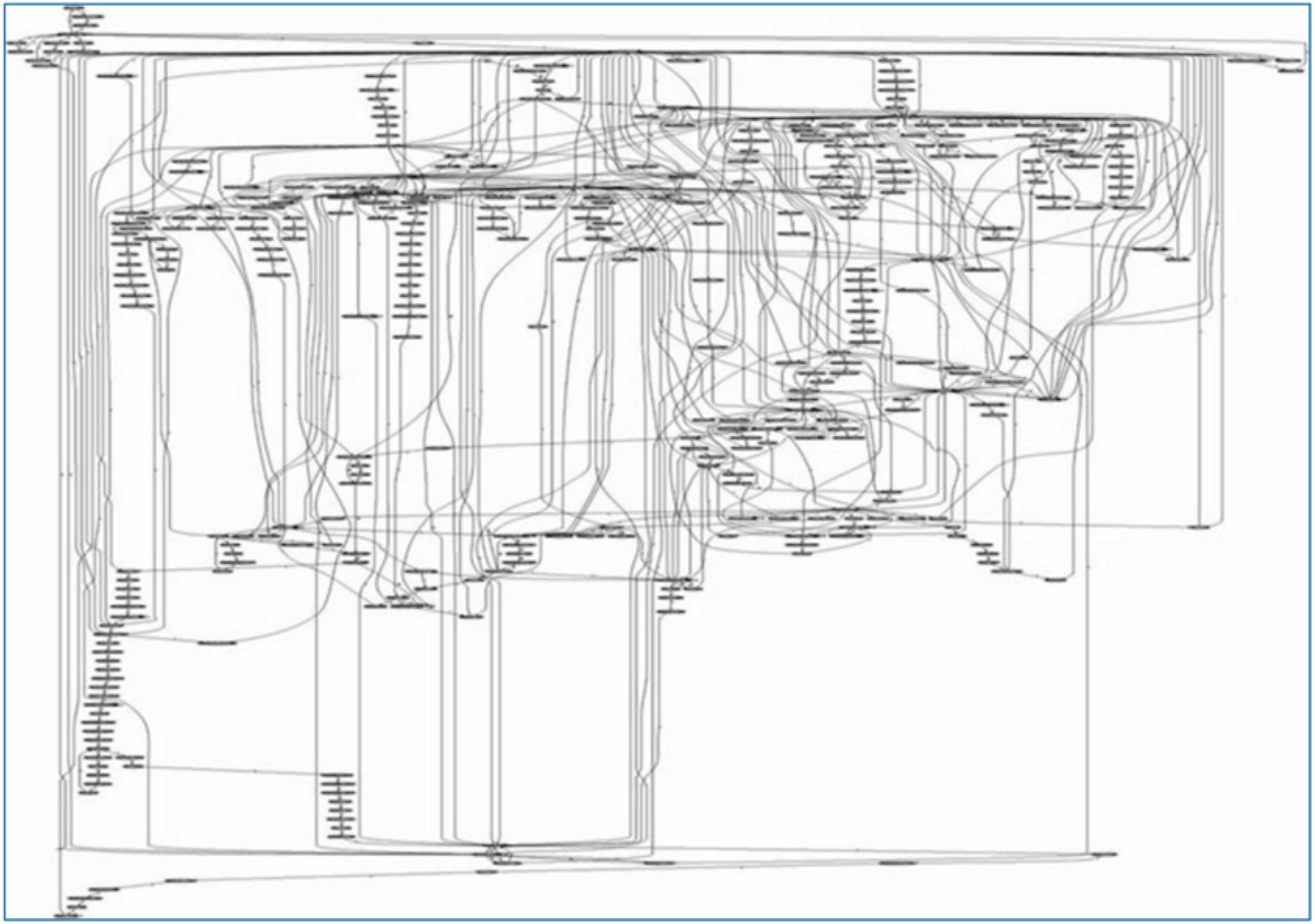
```
public class ResidentialAccount {

    public double sensingBalance = 0;

    public void charge(int gallons, LocalDate readingDate) {
        if (billingPlan.isMonthly()) {
            if (gallons < RESIDENTIAL_MIN) {
                balance += RESIDENTIAL_BASE;
            } else {
                balance += 1.2 * priceForGallons(gallons);
            }
            sensingBalance = balance;
            billingPlan.postReading(readingDate, gallons);
        }
    }
```

```
@Test
public void x() {
    ResidentialAccount account = new ResidentialAccount();
    account.charge(15, LocalDate.ofYearDay(2000, 1));
    assertEquals(account.sensingBalance, 0, 0.1);
}
```

```java
public class ResidentialAccount {

    public void charge(int gallons, LocalDate readingDate) {
        if (billingPlan.isMonthly()) {
            if (gallons < RESIDENTIAL_MIN) {
                balance += RESIDENTIAL_BASE;
            } else {
                balance += 1.2 * priceForGallons(gallons);
            }
            Vise.grip(balance);
            billingPlan.postReading(readingDate, gallons);
        }
    }
```

Reducing Conditionality

Apache



Microsoft IIS

[illegible]

```
if (…) {
    …
}
```

1	0
1	5
1	3
1	2
1	0
2	3
1	1
1	0
3	7

```
0--5--3--2--0-----1--0-----------------0-----------------------0--1--0--------------
---------------3-----------------1--3-----------------------------------3--0--------
------------------------7--2--2-----------2--4--0-----------------------------0--2--
---------------------------------------------------2--0-----------------------------
---------------------------------------------------------1--0-----------------------
------------------------------------------------------------------------------------
```

```ruby
STRING_COUNT = 6

def tab_column string, fret
  ["---"              ] * (string - 1) +
  [fret.ljust(3,'-')] +
  ["---"              ] * (STRING_COUNT - string)
end

puts ARGF.each_line
         .map(&:split)
         .map {|string,fret| tab_column(string.to_i, fret) }
         .transpose
         .map(&:join)
         .join($/)
```

Edge-Free Programming

```ruby
def make_entries text
  @valid_entries, @invalid_entries =
      *text.each_line
           .map(&:split)
           .zip((1..Float::INFINITY))
           .partition {|fields,_| parseable_date?(fields[1]) }

end


def render_errors
  @invalid_entries.each do |fields,line_no|
    @observer.data_error(line_no, "invalid date: #{fields[1]}")
  end
end


def render_summary
  @valid_entries.map(&:first)
                .map {|marker,date_string| [marker.strip, DateTime.parse(date_string)] }
                .group_by {|marker,_| marker }
                .map {|marker,groups| [marker,groups.map(&:second).max] }
                .map {|marker,date| "#{marker} #{days_unused(date)}" }
                .join
end
```

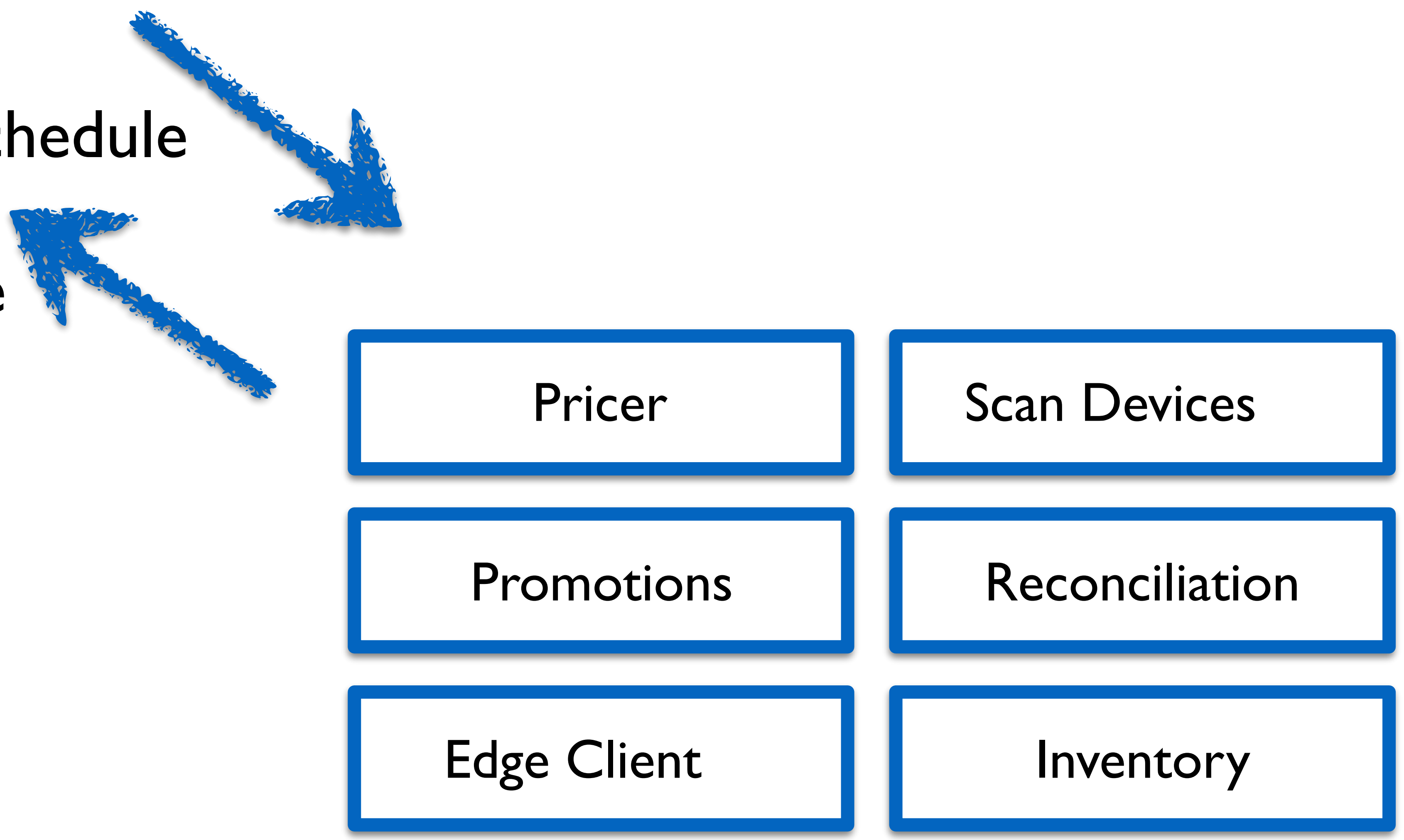
Redund Until Replace

Redund Until Replace

When you have a pure section of code that is a replacement, run it in parallel and log if there are differences

Conclusion

- Accept gift card
- Split charge (2 customers)
- Select home delivery
- Produce/Meat by weight
- Daily specials
- Weekly specials
- Restock perishable schedule
- Record invalid scans
- Change till procedure
- Cashier login
- Register reconcile



r7k