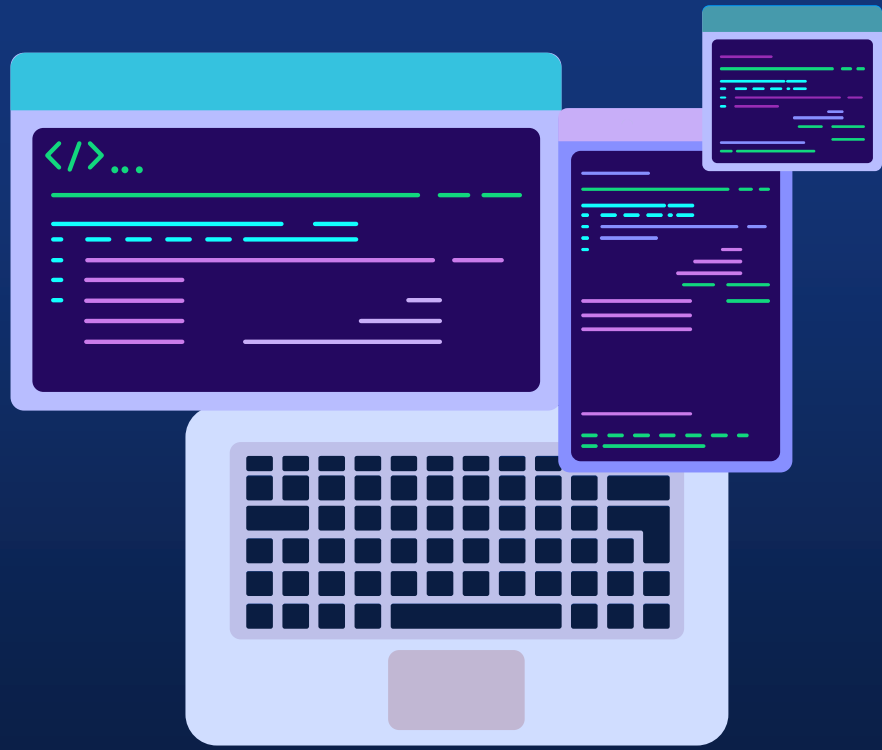
Software Design and Important concepts



Mentor: Einar Rocha

CONTENT

01
OOP Pillars

Inheritance, Polymorphism
Encapsulation, Abstraction

02
Clean Code

Meaningful Names,
Functions, Unit test
Code Smells...

03
SOLID

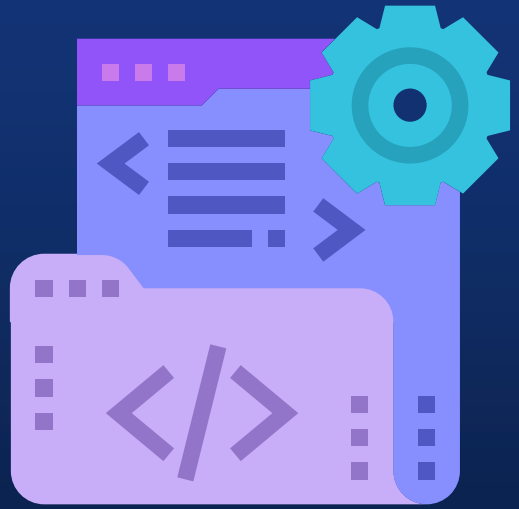
Single Responsiblity
Open closed
Liskov Substitution
Interface Segregation
Dependency Inversion

04
Design patterns

Singleton, Factory Method
Strategy, Observer
Builder...

02

Clean Code



The Goals of Software Design

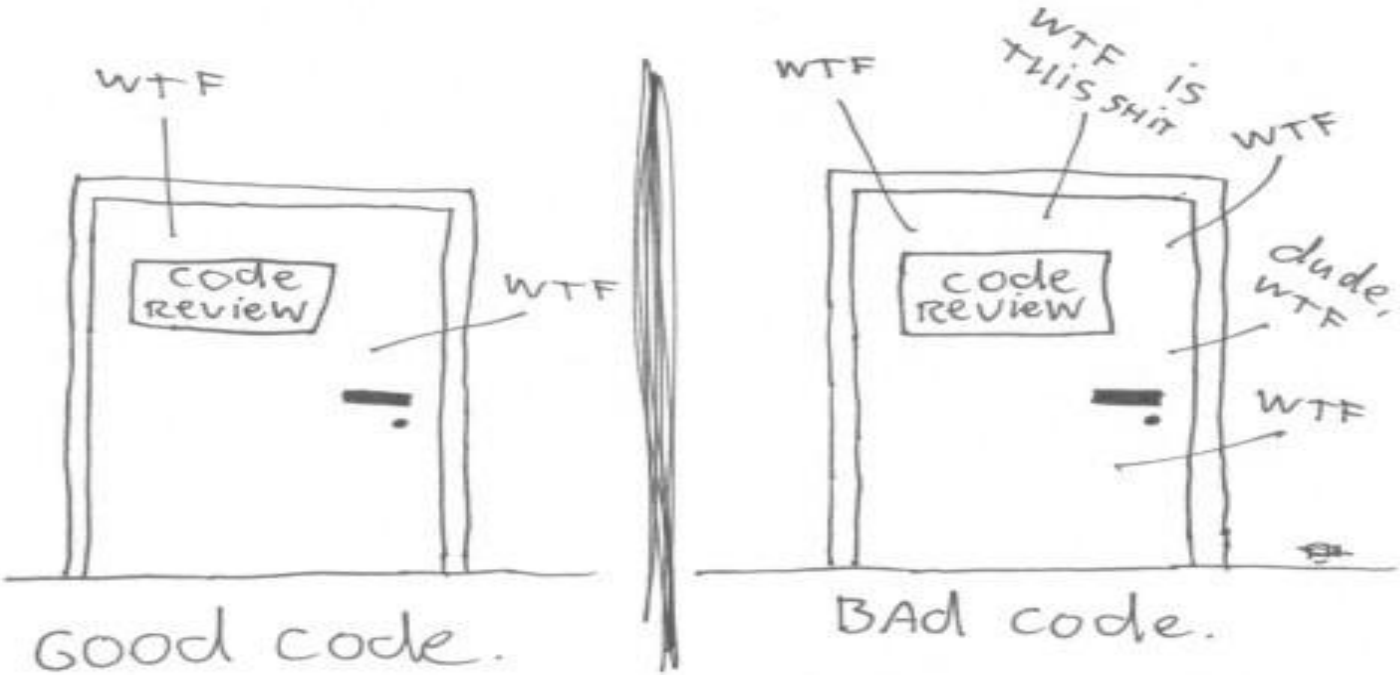
- To allow us to write software that is as helpful as possible.
- To allow our software to continue to be as helpful as possible.
- To design systems that can be created and maintained as easily as possible by their programmers

Agenda

Functions — Small, Do One thing, Descriptive names, arguments, side effects...

Comments — Explain yourself in code, Good comments, Bad comments...

The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE



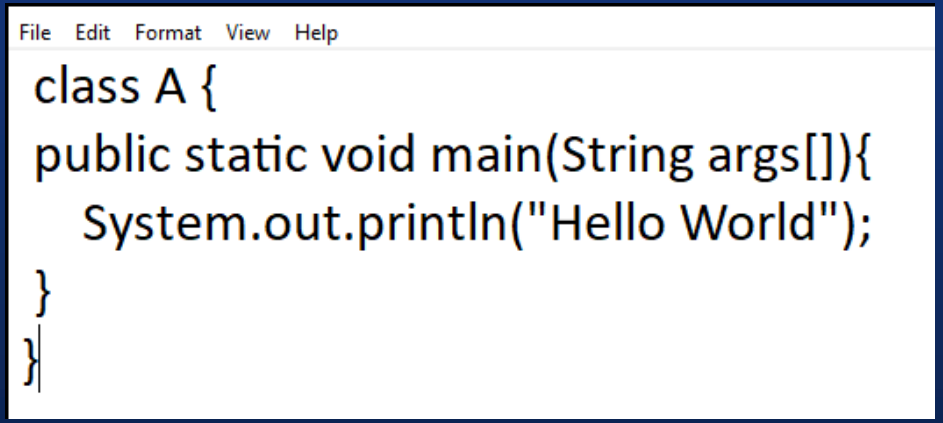
HtmlUtil.java

```
package com.clean;

public class TestHtml {
  public static String testableHtml(
      PageData pageData,
      boolean includeSuiteSetup
  ) throws Exception {
    WikiPage wikiPage = pageData.getWikiPage();
    StringBuffer buffer = new StringBuffer();
    if (pageData.hasAttribute("Test")) {
      if (includeSuiteSetup) {
        WikiPage suiteSetup =
            PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_SETUP_NAME, wikiPage
            );
        if (suiteSetup != null) {
          WikiPagePath pagePath =
              suiteSetup.getPageCrawler().getFullPath(suiteSetup);
          String pagePathName = PathParser.render(pagePath);
          buffer.append("!include -setup .")
              .append(pagePathName)
              .append("\n");
        }
      }
      WikiPage setup =
          PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
      if (setup != null) {
        WikiPagePath setupPath =
            wikiPage.getPageCrawler().getFullPath(setup);
        String setupPathName = PathParser.render(setupPath);
        buffer.append("!include -setup .")
            .append(setupPathName)
            .append("\n");
      }
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
      WikiPage teardown =
          PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
      if (teardown != null) {
        WikiPagePath tearDownPath =
            wikiPage.getPageCrawler().getFullPath(teardown);
        String tearDownPathName = PathParser.render(tearDownPath);
        buffer.append("\n")
            .append("!include -teardown .")
            .append(tearDownPathName)
            .append("\n");
      }
    }
    if (includeSuiteSetup) {
      WikiPage suiteTeardown =
          PageCrawlerImpl.getInheritedPage(
              SuiteResponder.SUITE_TEARDOWN_NAME,
              wikiPage
          );
      if (suiteTeardown != null) {
        WikiPagePath pagePath =
            suiteTeardown.getPageCrawler().getFullPath(suiteTeardown);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -teardown .")
            .append(pagePathName)
            .append("\n");
      }
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
  }

}
```

HtmlUtil.java

```
public static String renderPageWithSetupsAndTeardowns(
        PageData pageData, boolean isSuite
) throws Exception {
    boolean isTestPage = pageData.hasAttribute("Test");
    if (isTestPage) {
        WikiPage testPage = pageData.getWikiPage();
        StringBuffer newPageContent = new StringBuffer();
        includeSetupPages(testPage, newPageContent, isSuite);
        newPageContent.append(pageData.getContent());
        includeTeardownPages(testPage, newPageContent, isSuite);
        pageData.setContent(newPageContent.toString());
    }
    return pageData.getHtml();
}
```

Small


File Edit Format View Help
class A {
public static void main(String args[]){
System.out.println("Hello World");
}
}

HtmlUtil.java

```
public static String renderPageWithSetupsAndTeardowns(
      PageData pageData, boolean isSuite) throws Exception {
   if (isTestPage(pageData))
      includeSetupAndTeardownPages(pageData, isSuite);
   return pageData.getHtml();
}
```

Do One Thing

HtmlUtil.java

```
public static String renderPageWithSetupsAndTeardowns(
        PageData pageData, boolean isSuite) throws Exception {
    if (isTestPage(pageData))1
        includeSetupAndTeardownPages(pageData, isSuite);2
    return pageData.getHtml(); 3
}
```

HtmlUtil.java

```
public static String renderPageWithSetupsAndTeardowns(
        //Initialize
        ..................

        //Do something
        ..................

        //Validate
        ..............
}
```

Use Descriptive Names

```
public static String testHtml(
....
}


//To


public static String renderPageWithSetupsAndTeardowns(
....
}
```

Function Arguments

```
.....
public static String getHtml()


//AVOID
public static String get(String arg1, String, arg2, String arg3, String arg4)
```

Flag Arguments

```
.....
includeSetupAndTeardownPages(pageData, isSuite);


//To

includeSetupAndTeardownPagesForSuite(pageData);

includeSetupAndTeardownPagesForReport(pageData);
```

Dyadic Functions

```
.....
writeField(output-Stream, name);
```

```
writeField(name);
Point p = new Point(0,0);
assertEquals(expected, actual);
```

Triads Functions

```
.....
assertEquals(message, expected, actual)
```

Triads Functions

```
Circle makeCircle(double x, double y, double radius);
Circle makeCircle(Point center, double radius);
```

Side effects

```
public boolean checkPassword(String userName, String password) {
    User user = UserGateway.findByName(userName);
    if (user != User.NULL) {
        String codedPhrase = user.getPhraseEncodedByPassword();
        String phrase = cryptographer.decrypt(codedPhrase, password);
        if ("Valid Password".equals(phrase)) {
            Session.initialize();
            return true;
        }
    }
    return false;
}
```

Command Query Separation

```
public boolean set(String attribute, String value);

.....
if (set("username", "unclebob"))
```

```
if (attributeExists("username")) {
      setAttribute("username", "unclebob");
      ...
   }
```

Prefer Exceptions to Returning Error Codes

```
if (deletePage(page) == E_OK) {
    if (registry.deleteReference(page.name) == E_OK) {
        if (configKeys.deleteKey(page.name.makeKey()) == E_OK){
            logger.log("page deleted");
        } else {
            logger.log("configKey not deleted");
        }
    } else {
        logger.log("deleteReference from registry failed");
    }
} else {
    logger.log("delete failed");
    return E_ERROR;
}
```

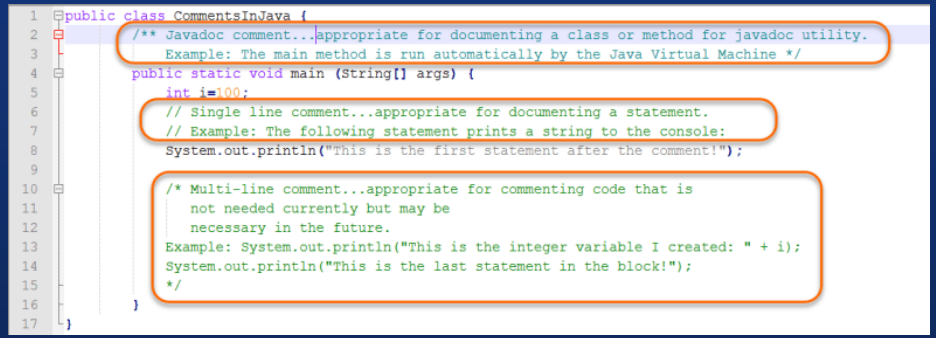
Prefer Exceptions to Returning Error Codes

```
try {
    deletePage(page);
    registry.deleteReference(page.name);
    configKeys.deleteKey(page.name.makeKey());
}
catch (Exception e) {
    logger.log(e.getMessage());
}
```

Extract Try/Catch Blocks

```
public void delete(Page page) {
    try {
        deletePageAndAllReferences(page);
    }
    catch (Exception e) {
        logError(e);
    }
}
private void deletePageAndAllReferences(Page page) throws Exception {
    deletePage(page);
    registry.deleteReference(page.name);
    configKeys.deleteKey(page.name.makeKey());
}
```

Comments



```
1  public class CommentsInJava {
2      /** Javadoc comment...appropriate for documenting a class or method for javadoc utility.
3          Example: The main method is run automatically by the Java Virtual Machine */
4      public static void main (String[] args) {
5          int i=100;
6          // Single line comment...appropriate for documenting a statement.
7          // Example: The following statement prints a string to the console:
8          System.out.println("This is the first statement after the comment!");
9
10         /* Multi-line comment...appropriate for commenting code that is
11            not needed currently but may be
12            necessary in the future.
13         Example: System.out.println("This is the integer variable I created: " + i);
14         System.out.println("This is the last statement in the block!");
15         */
16     }
17 }
```

Old Comments

```
MockRequest request;
// Example: "Tue, 02 Apr 2003 22:18:49 GMT"
private Response response;
private FitNesseContext context;
private FileResponder responder;
private Locale saveLocale;
private final String HTTP_DATE_REGEXP =
      "[SMTWF][a-z]{2}\\,\\s[0-9]{2}\\s[JFMASOND][a-z]{2}\\s"+
           "[0-9]{4}\\s[0-9]{2}\\:[0-9]{2}\\:[0-9]{2}\\sGMT";
```

Extract Try/Catch Blocks

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) &&
        (employee.age > 65))

// To
if (employee.isEligibleForFullBenefits())
```

Good Comments - Legal

```
// Copyright (C) 2003,2004,2005 by Object Mentor, Inc. All rights reserved.
// Released under the terms of the GNU General Public License version 2 or later.
package com.clean;
```

Good Comments - Informative

```
// Returns an instance of the Responder being tested.
protected abstract Responder responderInstance();

//maybe?


protected abstract Responder responderBeingTested();
```

Good Comments - Explanation of Intent

```
public int compareTo(Object o)
{
   if(o instanceof WikiPagePath)
   {
      WikiPagePath p = (WikiPagePath) o;
      String compressedName = StringUtil.join(names, "");
      String compressedArgumentName = StringUtil.join(p.names, "");
      return compressedName.compareTo(compressedArgumentName);
   }
   return 1; // we are greater because we are the right type.
}
```

Good Comments - Clarification

```
public void testCompareTo() throws Exception
{
   WikiPagePath a = PathParser.parse("PageA");
   WikiPagePath ab = PathParser.parse("PageA.PageB");
   WikiPagePath b = PathParser.parse("PageB");
   WikiPagePath aa = PathParser.parse("PageA.PageA");
   assertTrue(a.compareTo(a) == 0); // a == a
   assertTrue(a.compareTo(b) != 0); // a != b
   assertTrue(ab.compareTo(ab) == 0); // ab == ab
   assertTrue(a.compareTo(b) == -1); // a < b
   assertTrue(aa.compareTo(ab) == -1); // aa < ab
   assertTrue(b.compareTo(a) == 1); // b > a
   assertTrue(ab.compareTo(aa) == 1); // ab > aa
}
```

Good Comments - Warning of Consequences

```
// Don't run unless you
// have some time to kill.
public void _testWithReallyBigFile()
{
    writeLinesToFile(10000000);
    response.setBody(testFile);
    response.readyToSend(this);
    String responseString = output.toString();
    assertSubString("Content-Length: 1000000000", responseString);
    assertTrue(bytesSent > 1000000000);
}
```

Good Comments - ToDo

```
// TODO-MdM these are not needed
// We expect this to go away when we do the checkout model
protected versionInfo makeVersion() throws Exception
{
   return null;
}
```

Good Comments - Amplification

```
String listItemContent = match.group(3).trim();
// the trim is real important. It removes the starting
// spaces that could cause the item to be recognized
// as another list.
new ListItemWidget(this, listItemContent, this.level + 1);
return buildList(text.substring(match.end()));
```

Good Comments - Javadocs in Public APIs

```java
/**
 * <p>This is a simple description of the method. . .
 * <a href="http://www.supermanisthegreatest.com">Superman!</a>
 * </p>
 * @param incomingDamage the amount of incoming damage
 * @return the amount of health hero has after attack
 * @see <a href="http://www.link_to_jira/HERO-402">HERO-402</a>
 * @since 1.0
 */
public int successfullyAttacked(int incomingDamage) {
   // do things
   return 0;
}
```

Bad Comments - Mumbling

```
public void loadProperties()
{
   try
   {
      String propertiesPath = propertiesLocation + "/" + PROPERTIES_FILE;
      FileInputStream propertiesStream = new FileInputStream(propertiesPath);
      loadedProperties.load(propertiesStream);
   }
   catch(IOException e)
   {
      // No properties files means all defaults are loaded
   }
}
```

Bad Comments - Redundant Comments

```
// Utility method that returns when this.closed is true. Throws an exception
// if the timeout is reached.
public synchronized void waitForClose(final long timeoutMillis)
        throws Exception
{
    if(!closed)
    {
        wait(timeoutMillis);
        if(!closed)
            throw new Exception("MockResponseSender could not be closed");
    }
}
```

Bad Comments - Redundant Comments

```
public abstract class ContainerBase
      implements Container, Lifecycle, Pipeline,
      MBeanRegistration, Serializable {
   /**
    * The processor delay for this component.
    */
   protected int backgroundProcessorDelay = -1;
   /**
    * The lifecycle event support for this component.
    */
   protected LifecycleSupport lifecycle =
         new LifecycleSupport(this);
   /**
    * The container event listeners for this Container.
    */
   protected ArrayList listeners = new ArrayList();
   /**
    * The Loader implementation with which this Container is
    * associated.
    */
   protected Loader loader = null;
   /**
    * The Logger implementation with which this Container is
    * associated.
    */
   protected Log logger = null;
   /**
    * Associated logger name.
    */
   protected String logName = null;
.....................
```

Bad Comments - Misleading Comments

```
// Utility method that returns when this.closed is true. Throws an exception
// if the timeout is reached.
public synchronized void waitForClose(final long timeoutMillis)
        throws Exception
{
    if(!closed)
    {
        wait(timeoutMillis);
        if(!closed)
            throw new Exception("MockResponseSender could not be closed");
    }
}
```

Bad Comments - Mandated Comments

```
/**
 *
 * @param title The title of the CD
 * @param author The author of the CD
 * @param tracks The number of tracks on the CD
 * @param durationInMinutes The duration of the CD in minutes
 */
public void addCD(String title, String author,
                  int tracks, int durationInMinutes) {
    CD cd = new CD();
    cd.title = title;
    cd.author = author;
    cd.tracks = tracks;
    cd.duration = duration;
    cdList.add(cd);
}
```

Bad Comments - Journal Comments

```
/**
 * Changes (from 11-Oct-2001)
 * --------------------------
 * 11-Oct-2001 : Re-organised the class and moved it to new package
 * com.jrefinery.date (DG);
 * 05-Nov-2001 : Added a getDescription() method, and eliminated NotableDate
 * class (DG);
 * 12-Nov-2001 : IBD requires setDescription() method, now that NotableDate
 * class is gone (DG); Changed getPreviousDayOfWeek(),
 * getFollowingDayOfWeek() and getNearestDayOfWeek() to correct
 * bugs (DG);
 * 05-Dec-2001 : Fixed bug in SpreadsheetDate class (DG);
 * 29-May-2002 : Moved the month constants into a separate interface
 * (MonthConstants) (DG);
 * 27-Aug-2002 : Fixed bug in addMonths() method, thanks to N???levka Petr (DG);
 * 03-Oct-2002 : Fixed errors reported by Checkstyle (DG);
 * 13-Mar-2003 : Implemented Serializable (DG);
 * 29-May-2003 : Fixed bug in addMonths method (DG);
 * 04-Sep-2003 : Implemented Comparable. Updated the isInRange javadocs (DG);
 * 05-Jan-2005 : Fixed bug in addYears() method (1096282) (DG);
 */
```

Bad Comments - Noise Comments

```java
/**
 * Returns the day of the month.
 *
 * @return the day of the month.
 */
public int getDayOfMonth() {
   return dayOfMonth;
}
```

Bad Comments - Scary Noise

```
/** The name. */
private String name;
/** The version. */
private String version;
/** The licenceName. */
private String licenceName;
/** The version. */
private String info;
```

Don’t Use a Comment When You Can Use a Function or a Variable

```
// does the module from the global list <mod> depend on the
// subsystem we are part of?
if (smodule.getDependSubsystems().contains(subSysMod.getSubSystem()))



ArrayList moduleDependees = smodule.getDependSubsystems();
String ourSubSystem = subSysMod.getSubSystem();
if (moduleDependees.contains(ourSubSystem))
```

Bad Comments - Scary Noise

```
// Initializers ////////////////////////////////

.............

// Actions ////////////////////////////////

.............

// Functions ////////////////////////////////

.............
```

Bad Comments - Closing Brace Comments

```
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
String line;
int lineCount = 0;
int charCount = 0;
int wordCount = 0;
try {
   while ((line = in.readLine()) != null) {
      lineCount++;
      charCount += line.length();
      String words[] = line.split("\\W");
      wordCount += words.length;
   } //while
   System.out.println("wordCount = " + wordCount);
   System.out.println("lineCount = " + lineCount);
   System.out.println("charCount = " + charCount);
} // try
catch (IOException e) {
   System.err.println("Error:" + e.getMessage());
} //catch
```

Bad Comments - Commented-Out Code

```
InputStreamResponse response = new InputStreamResponse();
response.setBody(formatter.getResultStream(), formatter.getByteCount());
// InputStream resultsStream = formatter.getResultStream();
// StreamReader reader = new StreamReader(resultsStream);
// response.setContent(reader.read(formatter.getByteCount()));
```

Bad Comments - HTML Comments

```
/**
 * Task to run fit tests.
 * This task runs fitnesse tests and publishes the results.
 * <p/>
 * <pre>
 * Usage:
 * &lt;taskdef name=&quot;execute-fitnesse-tests&quot;
 * classname=&quot;fitnesse.ant.ExecuteFitnesseTestsTask&quot;
 * classpathref=&quot;classpath&quot; /&gt;
 * OR
 * &lt;taskdef classpathref=&quot;classpath&quot;
 * resource=&quot;tasks.properties&quot; /&gt;
 * <p/>
 * &lt;execute-fitnesse-tests
 * suitepage=&quot;FitNesse.SuiteAcceptanceTests&quot;
 * fitnesseport=&quot;8082&quot;
 * resultsdir=&quot;${results.dir}&quot;
 * resultshtmlpage=&quot;fit-results.html&quot;
 * classpathref=&quot;classpath&quot; /&gt;
 * </pre>
 */
```

Bad Comments - Nonlocal Information

```
/**
 * Port on which fitnesse would run. Defaults to <b>8082</b>.
 *
 * @param fitnessePort
 */
public void setFitnessePort(int fitnessePort)
{
   this.fitnessePort = fitnessePort;
}
```

Bad Comments - Too Much Information

```
/**
 * RFC 2045 - Multipurpose Internet Mail Extensions (MIME)
 *       Part One: Format of Internet Message Bodies
 *       section 6.8. Base64 Content-Transfer-Encoding
 *       The encoding process represents 24-bit groups of input bits as output
 *       strings of 4 encoded characters. Proceeding from left to right, a
 *       24-bit input group is formed by concatenating 3 8-bit input groups.
 *       These 24 bits are then treated as 4 concatenated 6-bit groups, each
 *       of which is translated into a single digit in the base64 alphabet.
 *       When encoding a bit stream via the base64 encoding, the bit stream
 *       must be presumed to be ordered with the most-significant-bit first.
 *       That is, the first bit in the stream will be the high-order bit in
 *       the first 8-bit byte, and the eighth bit will be the low-order bit in
 *       the first 8-bit byte, and so on.
 */
```

Bad Comments - Inobvious Connection

```
/**
 * start with an array that is big enough to hold all the pixels
 * (plus filter bytes), and an extra 200 bytes for header info
 */
this.pngBytes = new byte[((this.width + 1) * this.height * 3) + 200];
```

Bad Comments - Javadocs in Nonpublic Code

```
/**
 * This class Generates prime numbers up to a user specified
 * maximum. The algorithm used is the Sieve of Eratosthenes.
 * <p>
 * Eratosthenes of Cyrene, b. c. 276 BC, Cyrene, Libya --
 * d. c. 194, Alexandria. The first man to calculate the
 * circumference of the Earth. Also known for working on
 * calendars with leap years and ran the library at Alexandria.
 * <p>
 * The algorithm is quite simple. Given an array of integers
 * starting at 2. Cross out all multiples of 2. Find the next
 * uncrossed integer, and cross out all of its multiples.
 * Repeat untilyou have passed the square root of the maximum
 * value.
 *
 * @author Alphonse
 * @version 13 Feb 2002 atp
 */
import java.util.*;
public class GeneratePrimes
{
  /**
   * @param maxValue is the generation limit.
   */
  public static int[] generatePrimes(int maxValue)
  {
```