CLEAN CODE

Why You Should Care

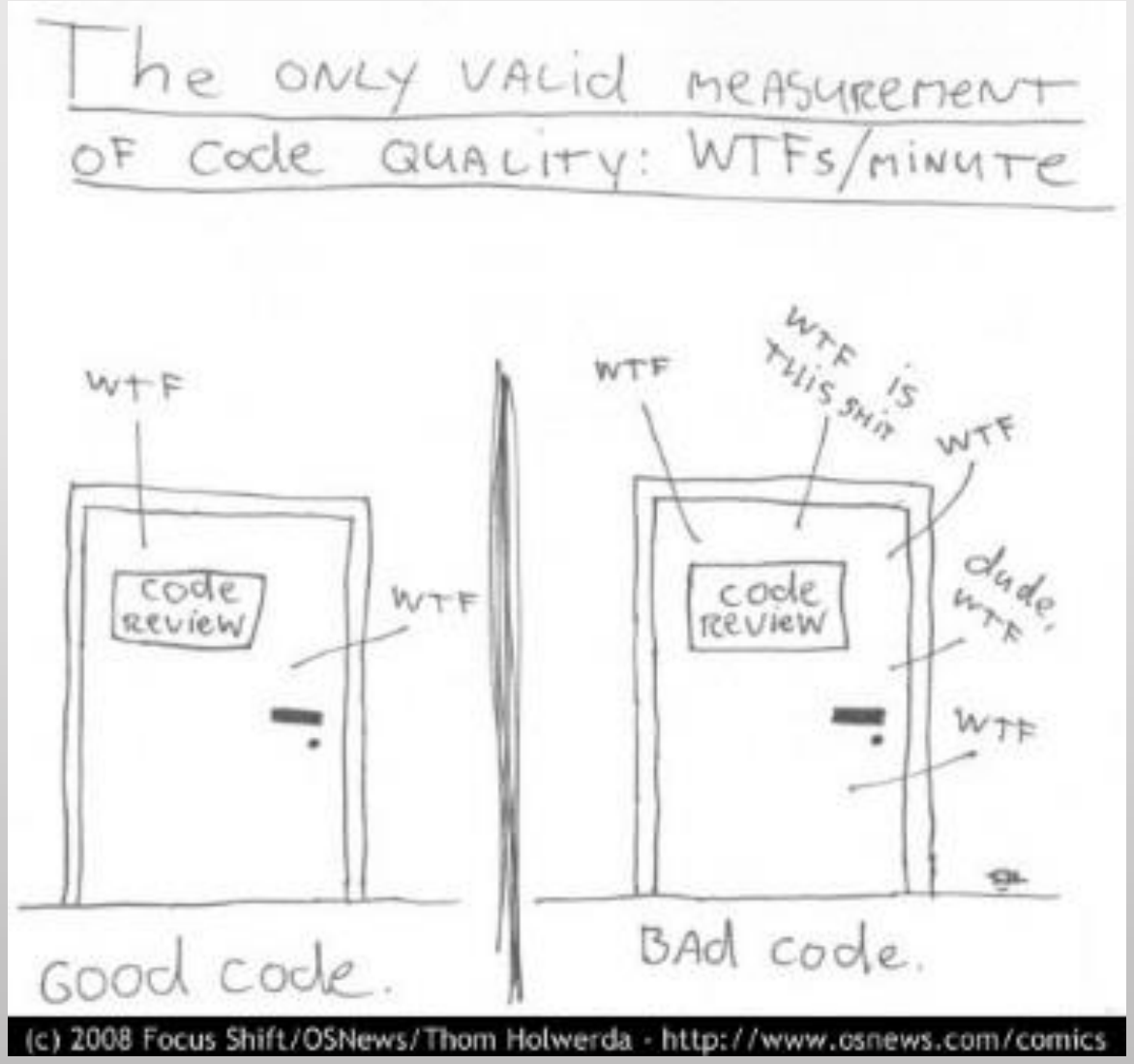
CERN Spring Campus 2014

Gijón, Spain - 15th April 2014

Benjamin Wolff (CERN / GS-AIS)

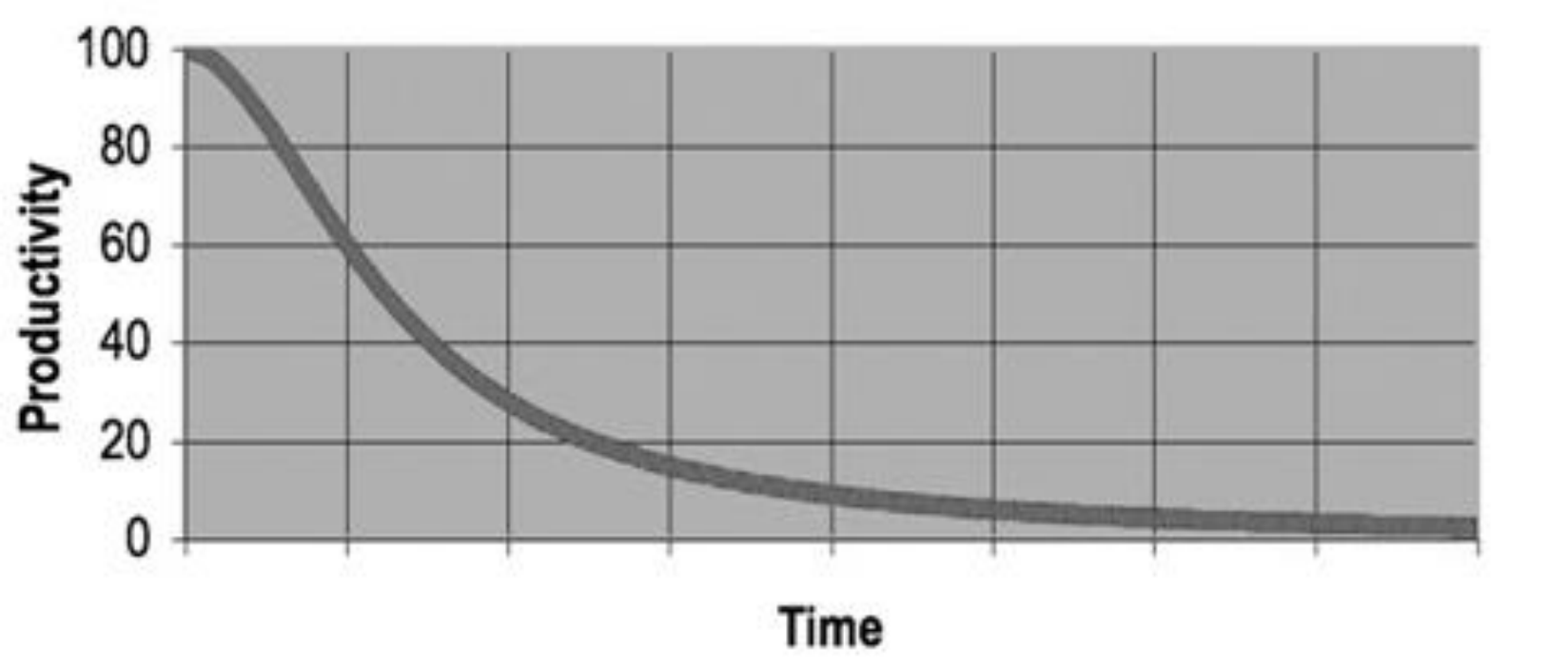
WHAT IS BAD CODE?

WHAT IS BAD CODE?



CONSEQUENCES OF BAD CODE

CONSEQUENCES OF BAD CODE



The Productivity Trap

Source: Clean Code (2009), Robert C. Martin, p.4

REASONS FOR BAD CODE

REASONS FOR BAD CODE

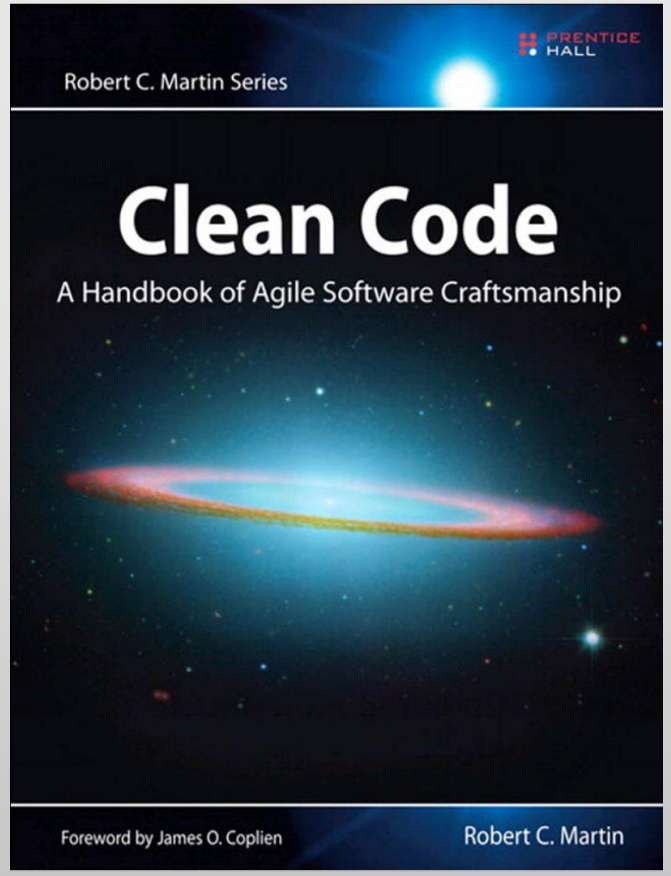
Code Rot

CLEAN CODE HANDBOOK

CLEAN CODE HANDBOOK

A Handbook of Agile Software Craftsmanship

by Robert C. Martin

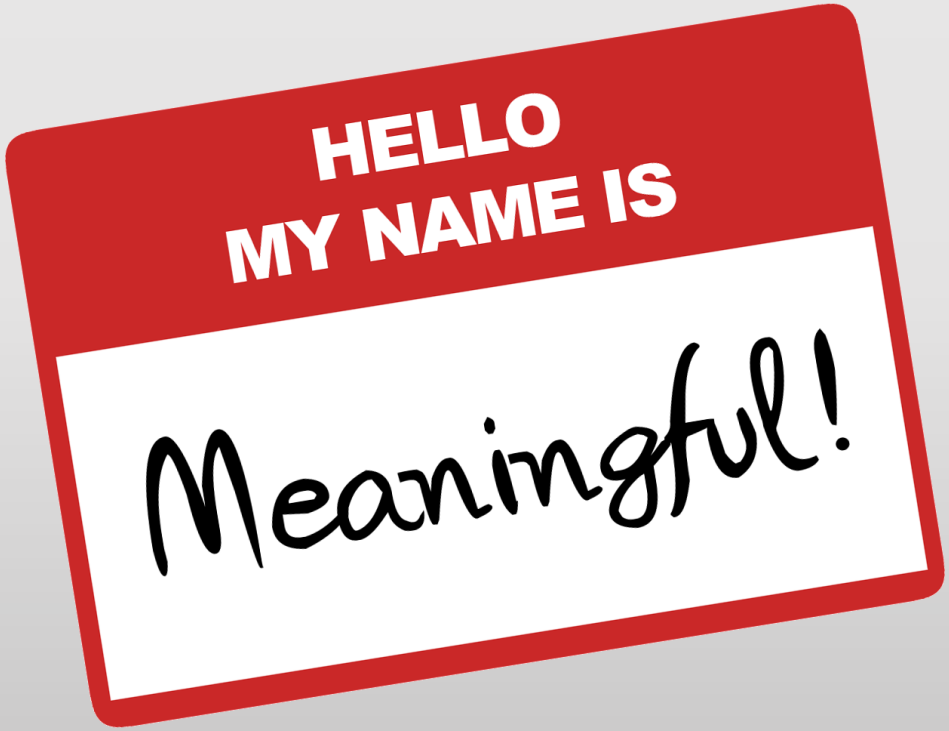


Portrait source: http://www.objectmentor.com/omTeam/martin_r.html

Agenda

1. Meaningful Names
2. Functions
3. Comments
4. Conclusion

MEANINGFUL NAMES



USE INTENTION-REVEALING NAMES

USE INTENTION-REVEALING NAMES

```
int etd;
int dsc;
int dsm;
int fad;
```

USE INTENTION-REVEALING NAMES

```
int etd;
int dsc;
int dsm;
int fad;
```

Better

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

USE INTENTION-REVEALING NAMES

```
int etd;
int dsc;
int dsm;
int fad;
```

Better

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

Avoid **mental mapping**

USE INTENTION-REVEALING NAMES

USE INTENTION-REVEALING NAMES

```
public List<int[]> getThem() {
  List<int[]> list1 = new ArrayList<int[]>();

  for (int[] x : theList) {
    if (x[0] == 4) {
      list1.add(x);
    }
  }

  return list1;
}
```

USE INTENTION-REVEALING NAMES

```
public List<int[]> getFlaggedCells() {
  List<int[]> flaggedCells = new ArrayList<int[]>();

  for (int[] cell : gameBoard) {
    if (cell[STATUS_VALUE] == FLAGGED) {
      flaggedCells.add(cell);
    }
  }

  return flaggedCells;
}
```

Better

USE INTENTION-REVEALING NAMES

```
public List<Cell> getFlaggedCells() {
  List<Cell> flaggedCells = new ArrayList<Cell>();

  for (Cell cell : gameBoard) {
    if (cell.isFlagged()) {
      flaggedCells.add(cell);
    }
  }

  return flaggedCells;
}
```

Even better

USE PRONOUNCABLE NAMES

USE PRONOUNCABLE NAMES

```
class DtaRcrd102 {
  private Date genymdhms;
  private Date modymdhms;
  private final String pszqint = "102";
  /* ... */
}
```

USE PRONOUNCABLE NAMES

```
class DtaRcrd102 {
  private Date genymdhms;
  private Date modymdhms;
  private final String pszqint = "102";
  /* ... */
}
```

Better

```
class Customer {
  private Date generationTimestamp;
  private Date modificationTimestamp;
  private final String recordId = "102";
  /* ... */
}
```

MAKE MEANINGFUL DISTINCTIONS

MAKE MEANINGFUL DISTINCTIONS

```
public static void copyChars(char a1[], char a2[]) {
  /* ... */
}
```

MAKE MEANINGFUL DISTINCTIONS

```
public static void copyChars(char a1[], char a2[]) {
  /* ... */
}
```

Better

```
public static void copyChars(char source[], char target[]) {
  /* ... */
}
```

AVOID DISINFORMATION

AVOID DISINFORMATION

```
XYZControllerForEfficientHandlingOfStrings
XYZControllerForEfficientStorageOfStrings
```

AVOID DISINFORMATION

```
XYZControllerForEfficientHandlingOfStrings
XYZControllerForEfficientStorageOfStrings
```

Beware of **small variations**

AVOID DISINFORMATION

```
XYZControllerForEfficientHandlingOfStrings
XYZControllerForEfficientStorageOfStrings
```

Beware of **small variations**

```
int a = l;

if (O == 1) {
  a = O1;
} else {
  l = 01;
}
```

AVOID DISINFORMATION

```
XYZControllerForEfficientHandlingOfStrings
XYZControllerForEfficientStorageOfStrings
```

Beware of **small variations**

```
int a = l;

if (O == 1) {
  a = O1;
} else {
  l = 01;
}
```

Only **obscures** the intent

AVOID ENCODINGS

AVOID ENCODINGS

Don‘t use **member prefixes**

```
public class Person {
  String m_firstName;
  String m_lastName;
  String m_description;
  /* ... */
}
```

AVOID ENCODINGS

Don‘t use **member prefixes**

```
public class Person {
  String m_firstName;
  String m_lastName;
  String m_description;
  /* ... */
}
```

Better

```
public class Person {
  String firstName;
  String lastName;
  String description;
  /* ... */
}
```

AVOID ENCODINGS

AVOID ENCODINGS

Don‘t use **Hungarian Notation**

```
String phoneNumberString;
List<String> phoneNumberList;
```

AVOID ENCODINGS

Don‘t use **Hungarian Notation**

```
String phoneNumberString;
List<String> phoneNumberList;
```

Names **don‘t change** when **types change**

```
PhoneNumber phoneNumberString;
Set<PhoneNumber> phoneNumberList;
```

AVOID ENCODINGS

Don‘t use **Hungarian Notation**

```
String phoneNumberString;
List<String> phoneNumberList;
```

Names **don‘t change** when **types change**

```
PhoneNumber phoneNumberString;
Set<PhoneNumber> phoneNumberList;
```

Better

```
PhoneNumber phoneNumber;
List<PhoneNumber> phoneNumbers;
```

AVOID MAGIC NUMBERS

AVOID MAGIC NUMBERS

```
if (data.length < 1048576) {
  /* ... */
}

if (status == 2) {
  /* ... */
}
```

AVOID MAGIC NUMBERS

```
if (data.length < 1048576) {
  /* ... */
}

if (status == 2) {
  /* ... */
}
```

Better

```
if (data.length < MAX_FILE_SIZE_KILOBYTES) {
  /* ... */
}

if (status == STATUS_ACTIVE) {
  /* ... */
}
```

AVOID STRING LITERALS

AVOID STRING LITERALS

```
String username = attributes.get("username");

if (username == null) {
  attributes.set("usernsme", "ben");
}

/* ... */

if (attributes.get("username") == null) {
  throw new AccessDeniedException("Missing username");
}
```

AVOID STRING LITERALS

```
String username = attributes.get("username");

if (username == null) {
  attributes.set("usernsme", "ben");
}

/* ... */

if (attributes.get("username") == null) {
  throw new AccessDeniedException("Missing username");
}
```

AVOID STRING LITERALS

```
static final String USERNAME_KEY = "username";

String username = attributes.get(USERNAME_KEY);

if (username == null) {
  attributes.set(USERNAME_KEY, "ben");
}

/* ... */

if (attributes.get(USERNAME_KEY) == null) {
  throw new AccessDeniedException("Missing username");
}
```

Better

FUNCTIONS

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Source: http://xkcd.com/221/

SMALL!

SMALL!

- 1. Rule: Functions should be **small**

SMALL!

- 1. Rule: Functions should be **small**
- 2. Rule: Functions should be **smaller** than that

SMALL!

- 1. Rule: Functions should be **small**
- 2. Rule: Functions should be **smaller** than that
- They should **hardly ever be >20 lines**

SMALL!

- 1. Rule: Functions should be **small**
- 2. Rule: Functions should be **smaller** than that
- They should **hardly ever be >20 lines**
- Preferably **less**!

DO ONE THING

DO ONE THING

- Functions should do **one thing**

DO ONE THING

- Functions should do **one thing**
- They should do it **well**

DO ONE THING

- Functions should do **one thing**
- They should do it **well**
- They should do it **only**

DO ONE THING

- Functions should do **one thing**
- They should do it **well**
- They should do it **only**

Source: http://www.sem-deutschland.de/wp-content/uploads/2013/10/eierlegendewollmilchsau-web.jpg

```java
public void renderWebPage() {
  StringBuilder content = getContentBuilder();
  content.append("<html>");

  content.append("<head>");
  for (HeaderElement he : getHeaderElements()) {
    String headerEntry = he.getStartTag() + he.getContent() +
        he.getEndTag();
    content.append(headerEntry);
  }
  content.append("</head>");

  content.append("<body>");
  for (BodyElement be : getBodyElements()) {
    String bodyEntry = /* .. */
    content.append(bodyEntry);
  }
  content.append("</body>");

  content.append("</html>");

  OutputStream output = new OutputStream(response);
  output.write(content.toString().getBytes());
  output.close();
}
```

```java
public void renderWebPage() {
  StringBuilder content = getContentBuilder();
  content.append("<html>");

  content.append("<head>");
  for (HeaderElement he : getHeaderElements()) {
    String headerEntry = he.getStartTag() + he.getContent() +
        he.getEndTag();
    content.append(headerEntry);
  }
  content.append("</head>");

  content.append("<body>");
  for (BodyElement be : getBodyElements()) {
    String bodyEntry = /* .. */
    content.append(bodyEntry);
  }
  content.append("</body>");

  content.append("</html>");

  OutputStream output = new OutputStream(response);
  output.write(content.toString().getBytes());
  output.close();
}
```

```java
public void renderWebPage() {
  StringBuilder content = getContentBuilder();
  content.append("<html>");

  content.append("<head>");
  for (HeaderElement he : getHeaderElements()) {
    String headerEntry = he.getStartTag() + he.getContent() +
        he.getEndTag();
    content.append(headerEntry);
  }
  content.append("</head>");

  content.append("<body>");
  for (BodyElement be : getBodyElements()) {
    String bodyEntry = /* .. */
    content.append(bodyEntry);
  }
  content.append("</body>");

  content.append("</html>");

  OutputStream output = new OutputStream(response);
  output.write(content.toString().getBytes());
  output.close();
}
```

```java
public void renderWebPage() {
  StringBuilder content = getContentBuilder();
  content.append("<html>");

  content.append("<head>");
  for (HeaderElement he : getHeaderElements()) {
    String headerEntry = he.getStartTag() + he.getContent() +
        he.getEndTag();
    content.append(headerEntry);
  }
  content.append("</head>");

  content.append("<body>");
  for (BodyElement be : getBodyElements()) {
    String bodyEntry = /* .. */
    content.append(bodyEntry);
  }
  content.append("</body>");

  content.append("</html>");

  OutputStream output = new OutputStream(response);
  output.write(content.toString().getBytes());
  output.close();
}
```

```java
public void renderWebPage() {
  StringBuilder content = getContentBuilder();
  content.append("<html>");

  content.append("<head>");
  for (HeaderElement he : getHeaderElements()) {
    String headerEntry = he.getStartTag() + he.getContent() +
        he.getEndTag();
    content.append(headerEntry);
  }
  content.append("</head>");

  content.append("<body>");
  for (BodyElement be : getBodyElements()) {
    String bodyEntry = /* .. */
    content.append(bodyEntry);
  }
  content.append("</body>");

  content.append("</html>");

  OutputStream output = new OutputStream(response);
  output.write(content.toString().getBytes());
  output.close();
}
```

```java
public void renderWebPage() {
  StringBuilder content = getContentBuilder();
  content.append("<html>");

  content.append("<head>");
  for (HeaderElement he : getHeaderElements()) {
    String headerEntry = he.getStartTag() + he.getContent() +
        he.getEndTag();
    content.append(headerEntry);
  }
  content.append("</head>");

  content.append("<body>");
  for (BodyElement be : getBodyElements()) {
    String bodyEntry = /* .. */
    content.append(bodyEntry);
  }
  content.append("</body>");

  content.append("</html>");

  OutputStream output = new OutputStream(response);
  output.write(content.toString().getBytes());
  output.close();
}
```

```java
public void renderWebPage() {
  startPage();
  includeHeaderContent();
  includeBodyContent();
  endPage();
  flushPageToResponse();
}
```

Better

```java
public void renderWebPage() {
  startPage();
  includeHeaderContent();
  includeBodyContent();
  endPage();
  flushPageToResponse();
}

private void startPage() { /* ... */ }

private void includeHeaderContent() { /* ... */ }

private void includeBodyContent() { /* ... */ }

private void endPage() { /* ... */ }

private void flushPageToResponse() { /* ... */ }
```

Better

THE NEWSPAPER METAPHOR

THE NEWSPAPER METAPHOR

```
public void renderWebPage() {
  startPage();
  includeHeaderContent();
  includeBodyContent();
  endPage();
  flushPageToResponse();
}

private void startPage() { /* ... */ }

private void includeHeaderContent() { /* ... */ }

private void includeBodyContent() { /* ... */ }

private void endPage() { /* ... */ }

private void flushPageToResponse() { /* ... */ }
```

THE NEWSPAPER METAPHOR

```
public void renderWebPage() {
  startPage();
  includeHeaderContent();
  includeBodyContent();
  endPage();
  flushPageToResponse();
}

private void startPage() { /* ... */ }

private void includeHeaderContent() { /* ... */ }

private void includeBodyContent() { /* ... */ }

private void endPage() { /* ... */ }

private void flushPageToResponse() { /* ... */ }
```

Title

THE NEWSPAPER METAPHOR

```
public void renderWebPage() {
  startPage();
  includeHeaderContent();
  includeBodyContent();
  endPage();
  flushPageToResponse();
}

private void startPage() { /* ... */ }

private void includeHeaderContent() { /* ... */ }

private void includeBodyContent() { /* ... */ }

private void endPage() { /* ... */ }

private void flushPageToResponse() { /* ... */ }
```

Title

Summary

THE NEWSPAPER METAPHOR

```
public void renderWebPage() {
  startPage();
  includeHeaderContent();
  includeBodyContent();
  endPage();
  flushPageToResponse();
}
```

Title

Summary

Details

```
private void startPage() { /* ... */ }

private void includeHeaderContent() { /* ... */ }

private void includeBodyContent() { /* ... */ }

private void endPage() { /* ... */ }

private void flushPageToResponse() { /* ... */ }
```

USE DESCRIPTIVE NAMES

USE DESCRIPTIVE NAMES

USE DESCRIPTIVE NAMES

```
article.calculate();
```

USE DESCRIPTIVE NAMES

```
article.calculate();
article.calculatePrice();
```

USE DESCRIPTIVE NAMES

```
article.calculate();
article.calculatePrice();
article.calculatePriceWithDiscount();
```

USE DESCRIPTIVE NAMES

```
article.calculate();
article.calculatePrice();
article.calculatePriceWithDiscount();
article.calculatePriceNetOfTax();
```

USE DESCRIPTIVE NAMES

```
article.calculate();
article.calculatePrice();
article.calculatePriceWithDiscount();
article.calculatePriceNetOfTax();
```

"You know you are working on clean code when each routine you read turns out to be pretty much ***what you expected****."*

— Ward Cunningham

Portrait source: https://en.wikipedia.org/wiki/File:Cunningham2.jpg

FUNCTION ARGUMENTS

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)
- Three (*triadic*) - should be **avoided**

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)
- Three (*triadic*) - should be **avoided**
- More (*polyadic*) - requires **special justification**

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)
- Three (*triadic*) - should be **avoided**
- More (*polyadic*) - requires **special justification**

```
pageRenderer.includeHeaderContent();
```

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)
- Three (*triadic*) - should be **avoided**
- More (*polyadic*) - requires **special justification**

```
pageRenderer.includeHeaderContent();
pageRenderer.writeTo(response);
```

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)
- Three (*triadic*) - should be **avoided**
- More (*polyadic*) - requires **special justification**

```
pageRenderer.includeHeaderContent();
pageRenderer.writeTo(response);
pageRenderer.writeTo(response, "UTF-8");
```

FUNCTION ARGUMENTS

- Zero (*niladic*) - **ideal**
- One (*monoadic*)
- Two (*dyadic*)
- Three (*triadic*) - should be **avoided**
- More (*polyadic*) - requires **special justification**

```
pageRenderer.includeHeaderContent();
pageRenderer.writeTo(response);
pageRenderer.writeTo(response, "UTF-8");

calendar.setWeekDate(2014, 3, 4);
```

FUNCTION ARGUMENTS

FUNCTION ARGUMENTS

```
public List<Article> searchArticles(String articleName,
    String articleNumber, String articleCategory, int pageSize,
    int pageOffset, int order) {
  /* ... */
}
```

FUNCTION ARGUMENTS

```
public List<Article> searchArticles(String articleName,
    String articleNumber, String articleCategory, int pageSize,
    int pageOffset, int order) {
  /* ... */
}
```

Better

```
public List<Article> searchArticles(ArticleSearchCriteria criteria) {
  /* ... */
}
```

FUNCTION ARGUMENTS

```
public List<Article> searchArticles(String articleName,
    String articleNumber, String articleCategory, int pageSize,
    int pageOffset, int order) {
  /* ... */
}
```

Better

```
public List<Article> searchArticles(ArticleSearchCriteria criteria) {
  /* ... */
}
```

```
critera = new ArticleSearchCriteria()
  .withArticleName("clean code")
  .withArticleNumber("abc123")
  .withArticleCategory("software engineering")
  .withPageSize(10);
```

AVOID FLAG ARGUMENTS

AVOID FLAG ARGUMENTS

```
rotate(90.0, true);
```

AVOID FLAG ARGUMENTS

```
rotate(90.0, true);
```

Function declaration tells us:

```
public void rotate(double degrees, boolean left) /* ... */
```

AVOID FLAG ARGUMENTS

```
rotate(90.0, true);
```

Function declaration tells us:

```
public void rotate(double degrees, boolean left) /* ... */
```

Better

```
public void rotateLeft(double degrees) /* ... */
public void rotateRight(double degrees) /* ... */

rotateLeft(90.0);
```

NO SIDE-EFFECTS

NO SIDE-EFFECTS

```
public boolean checkPassword(String username, String password) {
  User user = userDao.findByName(username);

  if (user != null) {
    String encodedPhrase = user.getPhraseEncodedByPassword();
    String decodedPhrase = cryptographer.decrypt(encodedPhrase,
        password);

    if ("valid passphrase".equals(decodedPhrase)) {
      Session.initialize();
      return true;
    }
  }

  return false;
}
```

NO SIDE-EFFECTS

```
public boolean checkPassword(String username, String password) {
  User user = userDao.findByName(username);

  if (user != null) {
    String encodedPhrase = user.getPhraseEncodedByPassword();
    String decodedPhrase = cryptographer.decrypt(encodedPhrase,
        password);

    if ("valid passphrase".equals(decodedPhrase)) {
      Session.initialize();
      return true;
    }
  }

  return false;
}
```

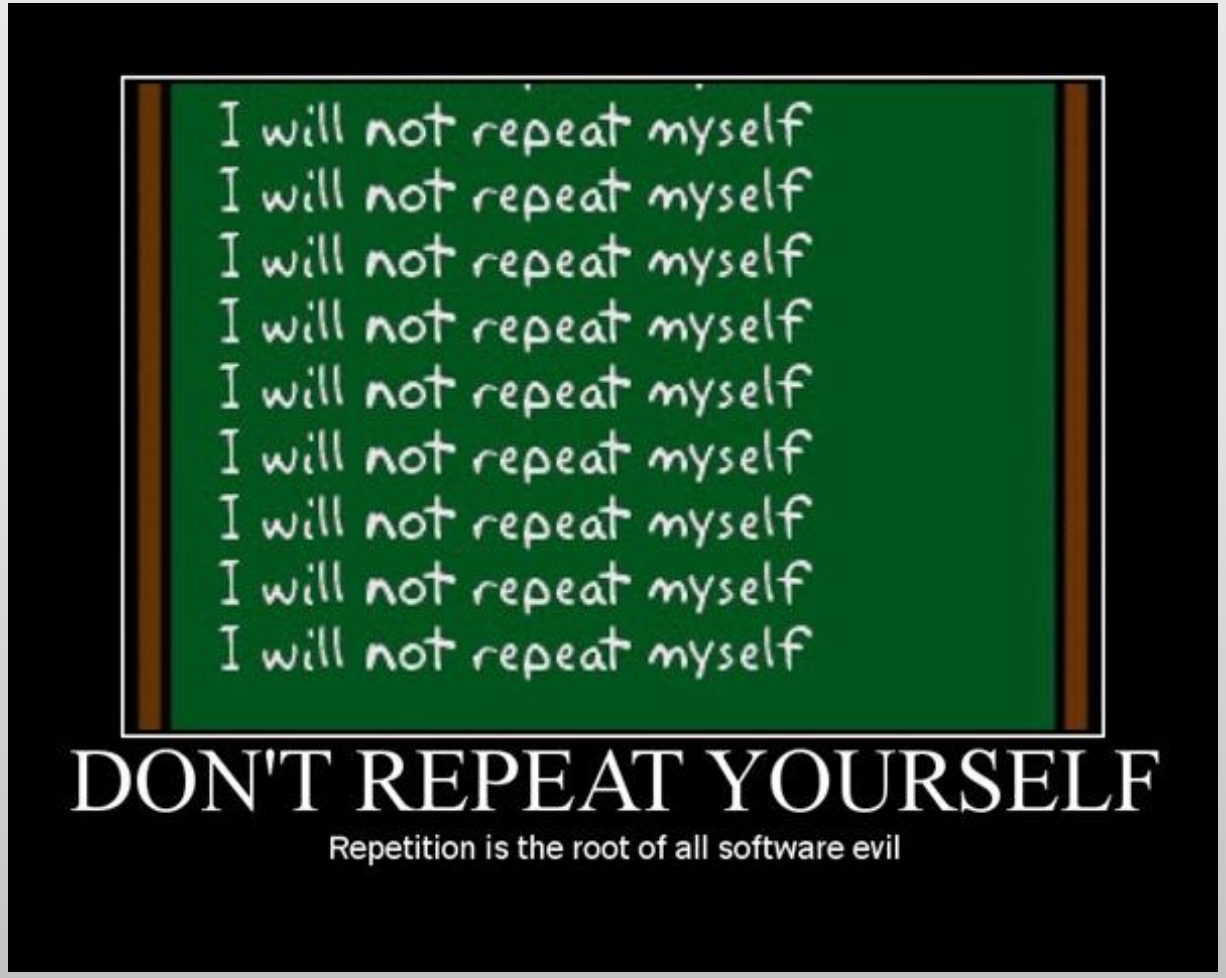
NO SIDE-EFFECTS

```
public boolean checkPassword(String username, String password) {
  User user = userDao.findByName(username);

  if (user != null) {
    String encodedPhrase = user.getPhraseEncodedByPassword();
    String decodedPhrase = cryptographer.decrypt(encodedPhrase,
        password);

    if ("valid passphrase".equals(decodedPhrase)) {
      Session.initialize();
      return true;
    }
  }

  return false;
}
```

Side effects are **lies**!

DON'T REPEAT YOURSELF

DON'T REPEAT YOURSELF



Source: http://tck.io/posts/sharing_models.html

DON'T REPEAT YOURSELF

DON'T REPEAT YOURSELF

```
if (account.id > 0 && account.name != null) {
  /* ... */
}

/* ... */

if (account.id > 0 && account.name != null) {
  /* ... */
}
```

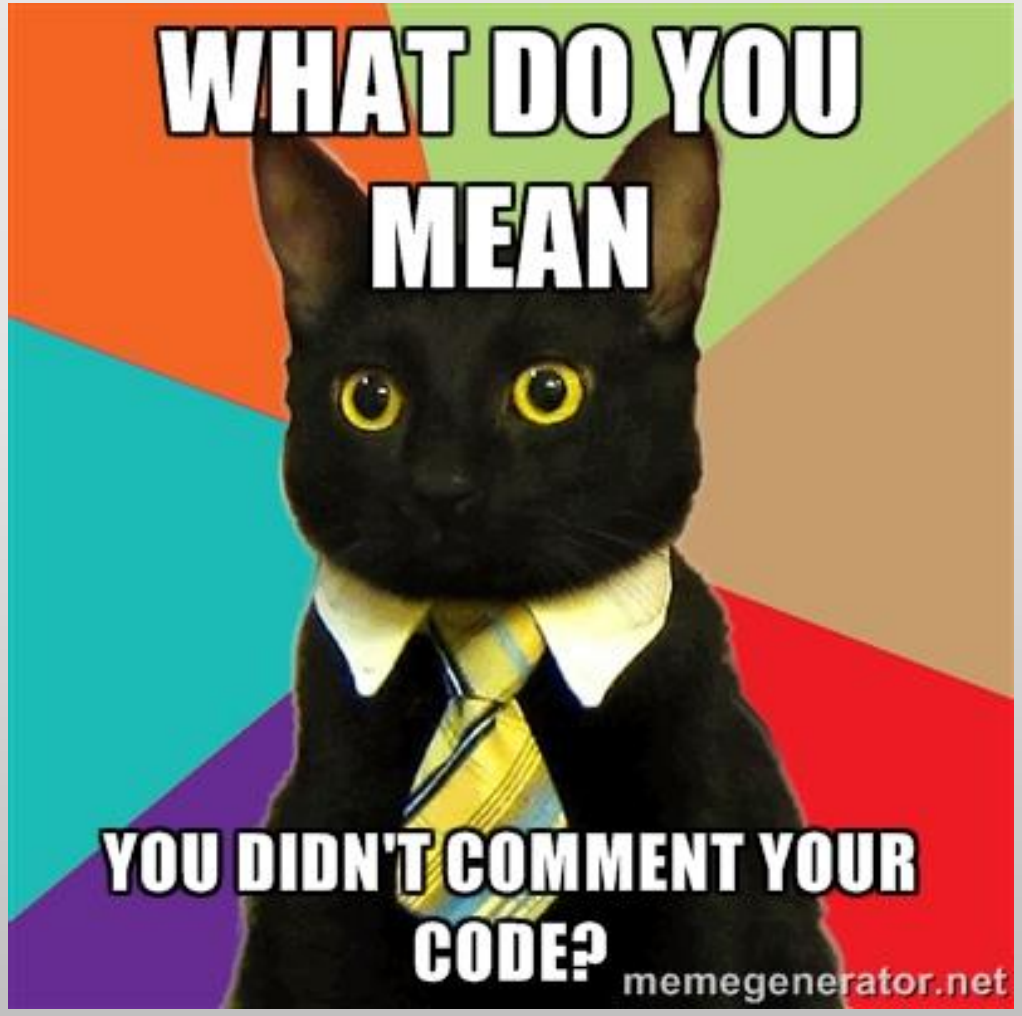
DON'T REPEAT YOURSELF

```
if (account.isValid()) {
  /* ... */
}

/* ... */

if (account.isValid()) {
  /* ... */
}
```

Better

COMMENTS

COMMENTS


WHAT DO YOU MEAN
YOU DIDN'T COMMENT YOUR CODE?
memegenerator.net

BAD COMMENTS

DON‘T COMMENT BAD CODE

DON'T COMMENT BAD CODE

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) != 0 && (employee.age > 65)) {
  /* ... */
}

String r; // lowercase URL
```

DON'T COMMENT BAD CODE

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) != 0 && (employee.age > 65)) {
  /* ... */
}

String r; // lowercase URL
```

Rewrite it!

```
if (employee.isEligibleForFullBenefits()) {
  /* ... */
}

String lowerCaseUrl;
```

DON'T COMMENT BAD CODE

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) != 0 && (employee.age > 65)) {
  /* ... */
}

String r; // lowercase URL
```

Rewrite it!

```
if (employee.isEligibleForFullBenefits()) {
  /* ... */
}

String lowerCaseUrl;
```

Express yourself **in code**

REDUNDANT COMMENTS

REDUNDANT COMMENTS

```
// Check if members have been initialized. If not, do it!
if (members == null) {
  members = new ArrayList();
}
```

REDUNDANT COMMENTS

```
// Check if members have been initialized. If not, do it!
if (members == null) {
  members = new ArrayList();
}
```

Don‘t be ***Captain Obvious***

Source: https://www.flickr.com/photos/g4r37h/2389099529/in/set-72157622028121764

MANDATED COMMENTS

MANDATED COMMENTS

Every function **must have** a JavaDoc comment

MANDATED COMMENTS

Every function **must have** a JavaDoc comment

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

NOISE COMMENTS

NOISE COMMENTS

```
public class Person {
  /**
   * The ID of the Person
   */
  private long id;

  /**
   * Default constructor.
   */
  public Person() {}

  /**
   * Returns the ID.
   * @return the ID
   */
  public long getId() {
    return id;
  }
```

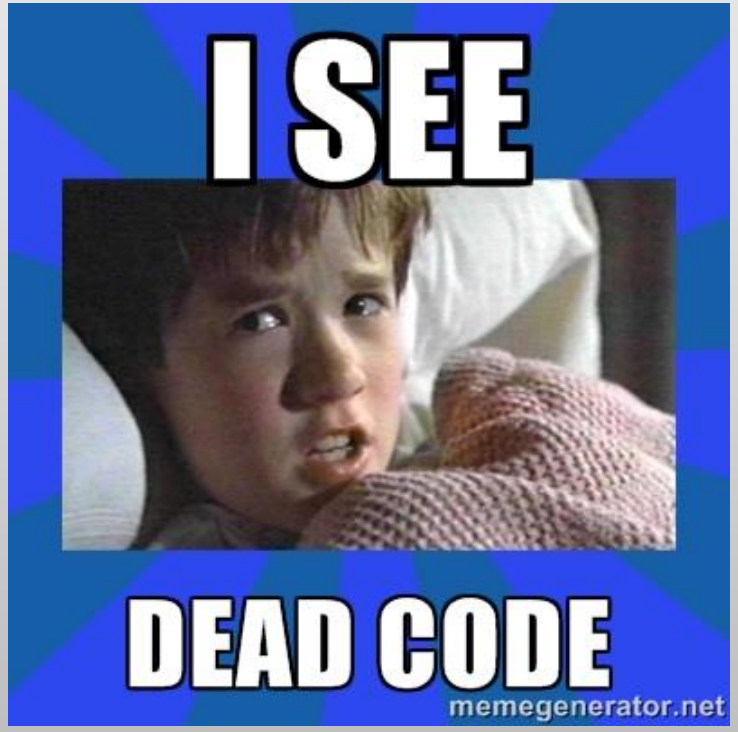
COMMENTED-OUT CODE

COMMENTED-OUT CODE

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

COMMENTED-OUT CODE

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



MISLEADING COMMENTS

MISLEADING COMMENTS

```
/**
 * Always returns true.
 */
public boolean isAvailable() {
  return false;
}
```

MISLEADING COMMENTS

```
/**
 * Always returns true.
 */
public boolean isAvailable() {
  return false;
}
```

Comments are **lies** waiting to happen

MISLEADING COMMENTS

```
/**
 * Always returns true.
 */
public boolean isAvailable() {
  return false;
}
```

Comments are **lies** waiting to happen

But the code **never lies**!

GOOD COMMENTS

INFORMATIVE COMMENTS

INFORMATIVE COMMENTS

```
// Matches hh:mm:ss EEE, MMM dd, yyyy
Pattern timeMatcher = Pattern.compile(
  "\\d{2}:\\d{2}:\\d{2} \\w{3}, \\w{3} \\d{2}, \\d{4}"
);
```

EXPLANATION OF INTEND

EXPLANATION OF INTEND

```
// We need to remove duplicates from the names because
// a person cannot more the same name more than once.
Set<String> uniqueNames = new HashSet<String>(names);
```

EXPLANATION OF INTEND

```
// We need to remove duplicates from the names because
// a person cannot more the same name more than once.
Set<String> uniqueNames = new HashSet<String>(names);
```

Describe **why** you do something - **not how**

AMPLIFICATION

AMPLIFICATION

```
// The trim is real important. It removes the starting
// spaces that could cause the item to be recognized
// as another list.
String listItemContent = match.group(3).trim();
```

WARNING OF CONSEQUENCES

WARNING OF CONSEQUENCES

```
public static SimpleDateFormat createStandardHttpDateFormat() {
  // SimpleDateFormat is not thread safe, so we need to
  // create each instance independently.
  return new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z");
}
```

Public API JavaDocs

Public API JavaDocs

```
@XmlTransient
public WeightedRandomSet<String> options = new WeightedRandomSet<String>();
```

toxi.util.datatypes.WeightedRandomSet<String>

This class provides a generic random-weight distribution of arbitary objects. Add elements with their weight to the set and then use the `getRandom()` method to retrieve objects. The frequency of returned elements is based on their relative weight. This makes it easy to provide biased preferences.
http://www.electricmonk.nl/log/2009/12/23/weighted-random-distribution/

Press 'F2' for focus

Source: http://toxiclibs.org/2011/09/the-road-ahead/

TODO COMMENTS

TODO COMMENTS

```
// TODO AISFWK-718: This will be refactored when we
// migrate to the new binding process
public ViewModel bindToViewModel(Model domainModel) {
  /* ... */
}
```

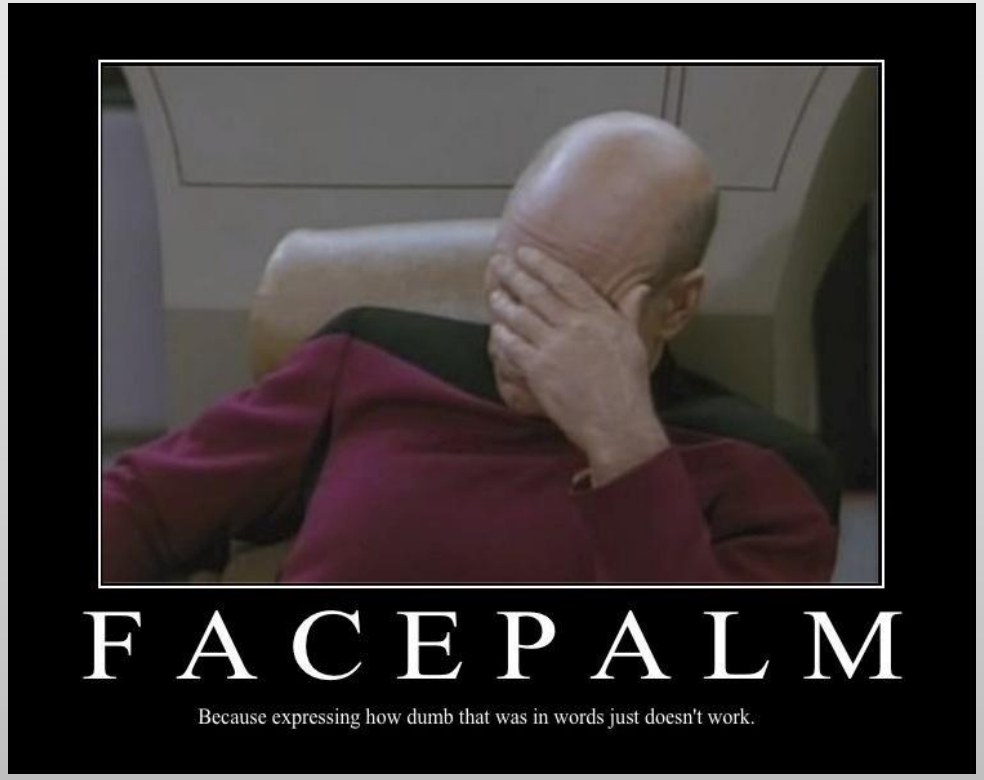
TODO COMMENTS

TODO COMMENTS

```
boolean hasAccessRights(User user) {
  // TODO Implement proper access checks
  return true;
}
```

TODO COMMENTS

```
boolean hasAccessRights(User user) {
  // TODO Implement proper access checks
  return true;
}
```



CONCLUSION

Clean Code is about…

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code
- ... authoring for the **readers**

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code
- ... authoring for the **readers**
- ... **refactoring** and **cleaning** existing code

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code
- ... authoring for the **readers**
- ... **refactoring** and **cleaning** existing code
- ... **caring** about your code

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code
- ... authoring for the **readers**
- ... **refactoring** and **cleaning** existing code
- ... **caring** about your code
- ... being **professional**

Clean Code is about...

- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code
- ... authoring for the **readers**
- ... **refactoring** and **cleaning** existing code
- ... **caring** about your code
- ... being **professional**

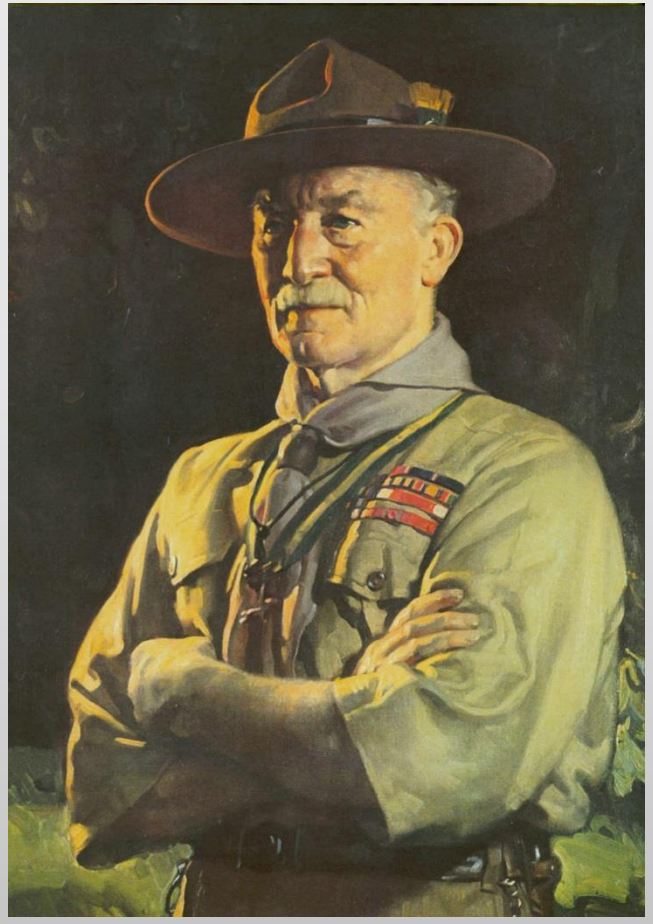
And there is a lot more to it!

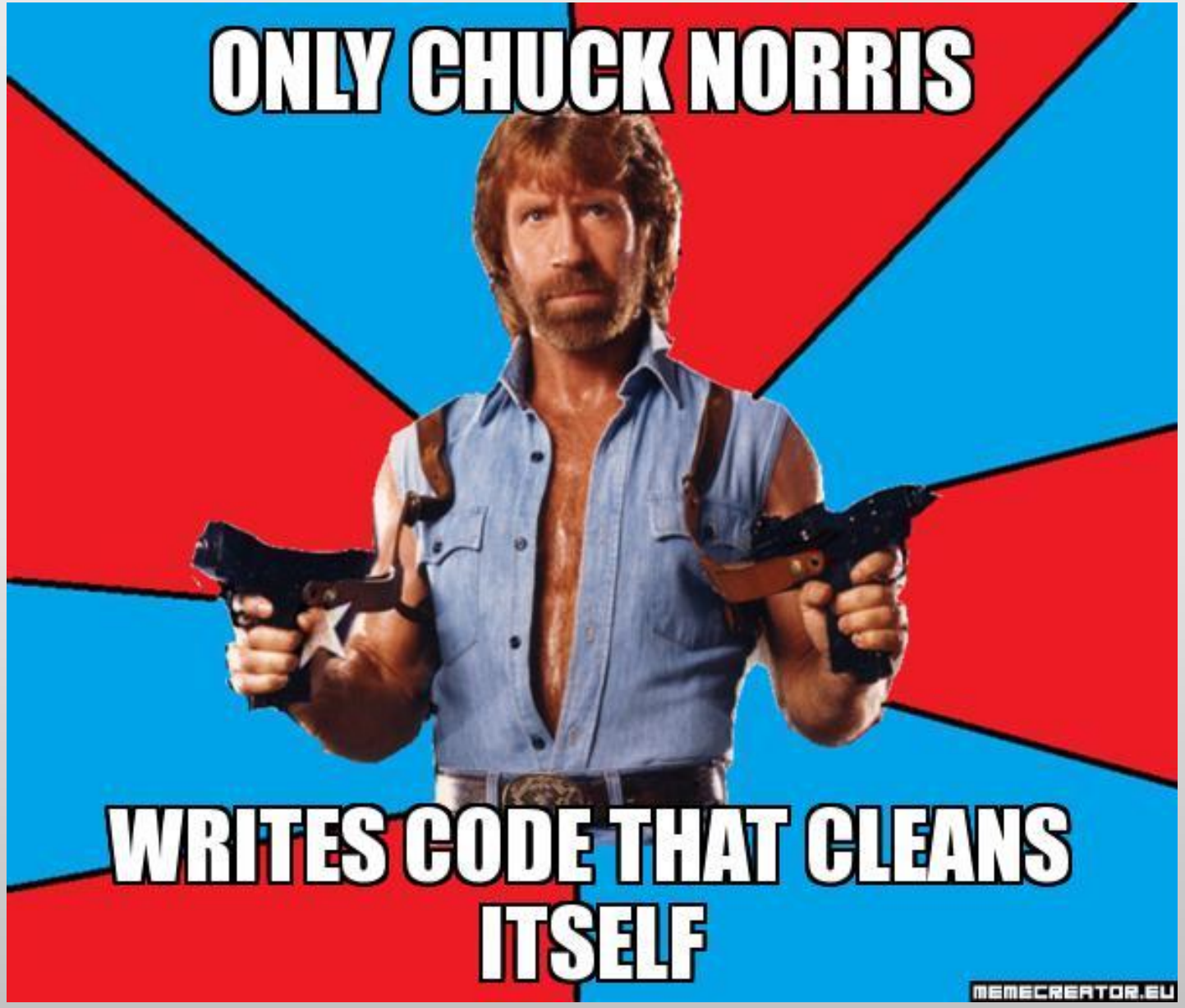
THE BOY SCOUT RULE

THE BOY SCOUT RULE

"Leave the campground cleaner than you found it."

—Inspired by Robert Baden Powell




ONLY CHUCK NORRIS
WRITES CODE THAT CLEANS ITSELF
MEMECREATOR.EU

¡GRACIAS POR VUESTRA ATENCIÓN!

¿PREGUNTAS?

```
void answerAudienceQuestions() {
  while (audience.hasQuestions()) {
    Question question = audience.getNextQuestion();
    speaker.answer(question);
  }
}
```

BACKUP SLIDES

CLASS NAMES

CLASS NAMES

Should be **nouns** or **noun phrases**

```
Customer
WikiPage
Account
AddressParser
```

CLASS NAMES

Should be **nouns** or **noun phrases**

```
Customer
WikiPage
Account
AddressParser
```

But avoid **noninformative words**

```
Manager
Processor
Data
Info
```

FUNCTION NAMES

FUNCTION NAMES

Should be **verbs** or **verbs phrases**

```
save()
postPayment()
deletePage()
```

FUNCTION NAMES

Should be **verbs** or **verbs phrases**

```
save()
postPayment()
deletePage()
```

Use conventions for **accessors/mutators/predicates**

```
String name = employee.getName();

customer.setName("Ben");

if (paycheck.isPosted()) {
  /* ... */
}
```

USE SOLUTION DOMAIN NAMES

USE SOLUTION DOMAIN NAMES

```
AccountVisitor
JobQueue
NodeStack
```

USE SOLUTION DOMAIN NAMES

```
AccountVisitor
JobQueue
NodeStack
```

USE PROBLEM DOMAIN NAMES

```
AbsenceRequest
TravelClaim
MaterialOrder
```

COMMAND QUERY SEPARATION

COMMAND QUERY SEPARATION

Functions should **do** or **answer** something

COMMAND QUERY SEPARATION

Functions should **do** or **answer** something

Not **both**!

COMMAND QUERY SEPARATION

Functions should **do** or **answer** something

Not **both**!

```
if (set("username", "ben")) {
  /* ... */
}
```

COMMAND QUERY SEPARATION

Functions should **do** or **answer** something

Not **both**!

```
if (set("username", "ben")) {
  /* ... */
}
```

Avoid **ambiguity**

```
if (attributeExists("username")) {
  setAttribute("username", "ben");
  /* ... */
}
```

COMMAND QUERY SEPARATION

Functions should **do** or **answer** something

Not **both**!

```
if (set("username", "ben")) {
  /* ... */
}
```

Avoid **ambiguity**

```
static final String USERNAME_KEY = "username";

/* ... */

if (attributeExists(USERNAME_KEY)) {
  setAttribute(USERNAME_KEY, "ben");
  /* ... */
}
```