CLEAN CODE

Why You Should Care

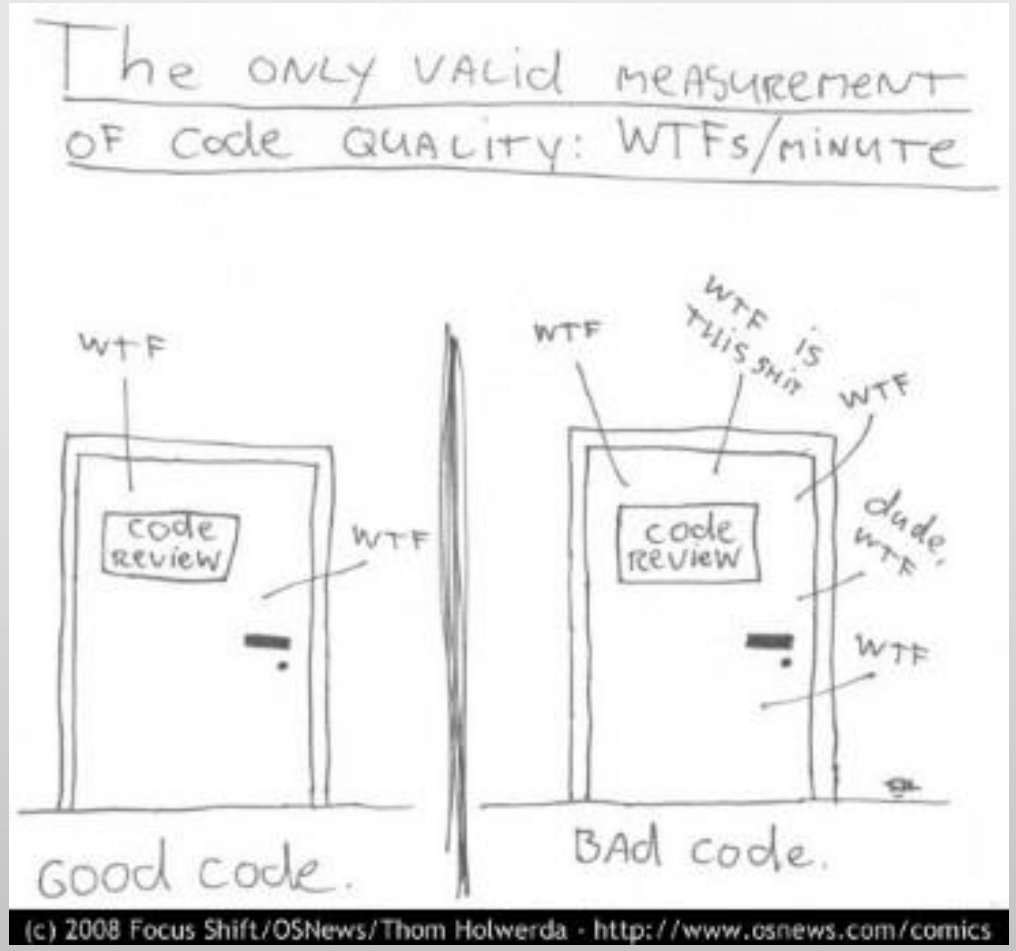
1st Developers@CERN Forum

29th September 2015

Benjamin Wolff (GS-AIS-GDI)

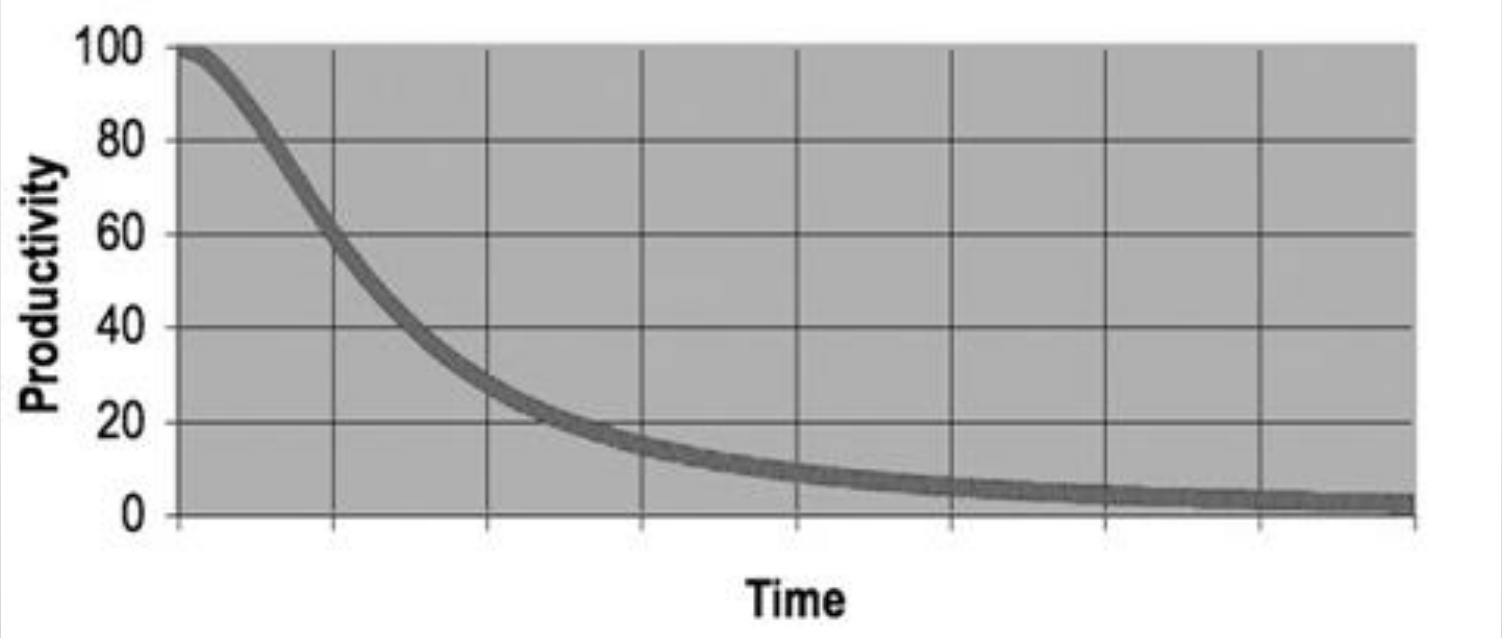
WHAT IS BAD CODE?

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

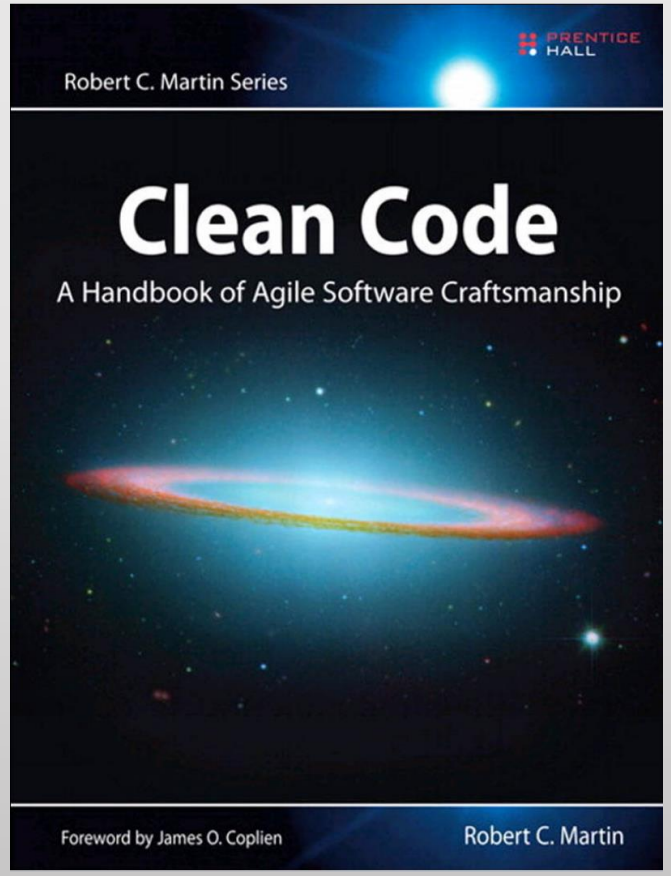
REASONS FOR BAD CODE

Code Rot

CLEAN CODE HANDBOOK

A Handbook of Agile Software Craftsmanship

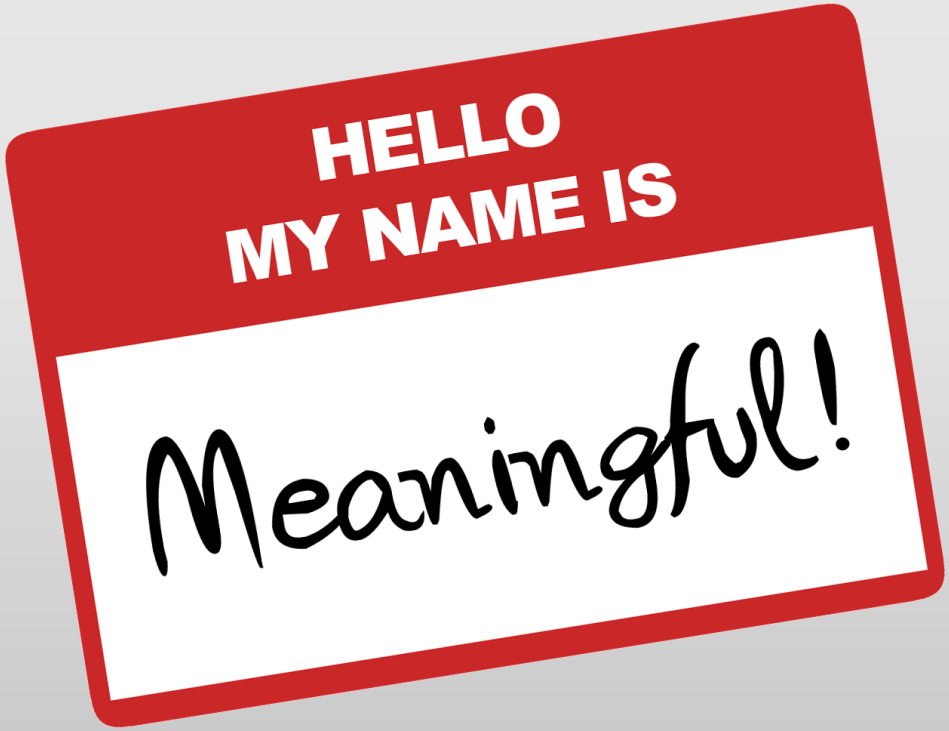
by Robert C. Martin



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;
Date genymdhms;
```

USE INTENTION-REVEALING NAMES

```
int etd;
int dsc;
int dsm;
Date genymdhms;
```

Better

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
Date generationTimestamp;
```

USE INTENTION-REVEALING NAMES

```
int etd;
int dsc;
int dsm;
Date genymdhms;
```

Better

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
Date generationTimestamp;
```

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

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_BYTES) {
  /* ... */
}

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

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();
  writePageToResponse();
}
```

Better

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

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

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

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

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

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

Better

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

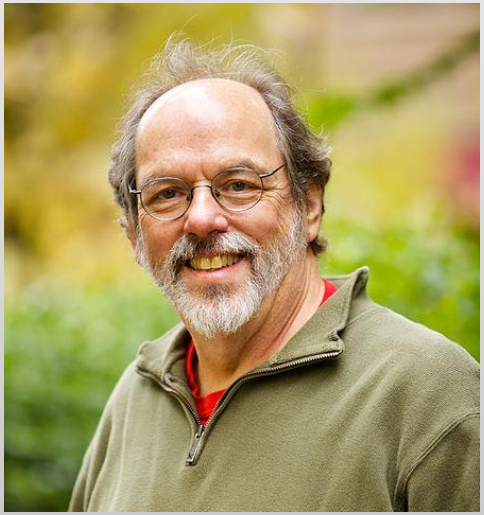
```
article.calculate();
article.calculatePrice();
article.calculatePriceWithDiscount();
article.calculatePriceNetOfTax();
```

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: "Ward Cunningham - Commons-1" by Carrigg Photography for the Wikimedia Foundation - Own work. Licensed under CC BY-SA 3.0 via Commons https://commons.wikimedia.org/wiki/File:Ward_Cunningham_-_Commons-1.jpg#/media/File:Ward_Cunningham_-_Commons-1.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.setDate(2014, 3, 4);
```

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.setDate(2014, 3, 4);

find(input, null, "articles", null, "asc", 10, 5, false, true);
```

FUNCTION ARGUMENTS

FUNCTION ARGUMENTS

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

FUNCTION ARGUMENTS

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

Better

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

FUNCTION ARGUMENTS

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

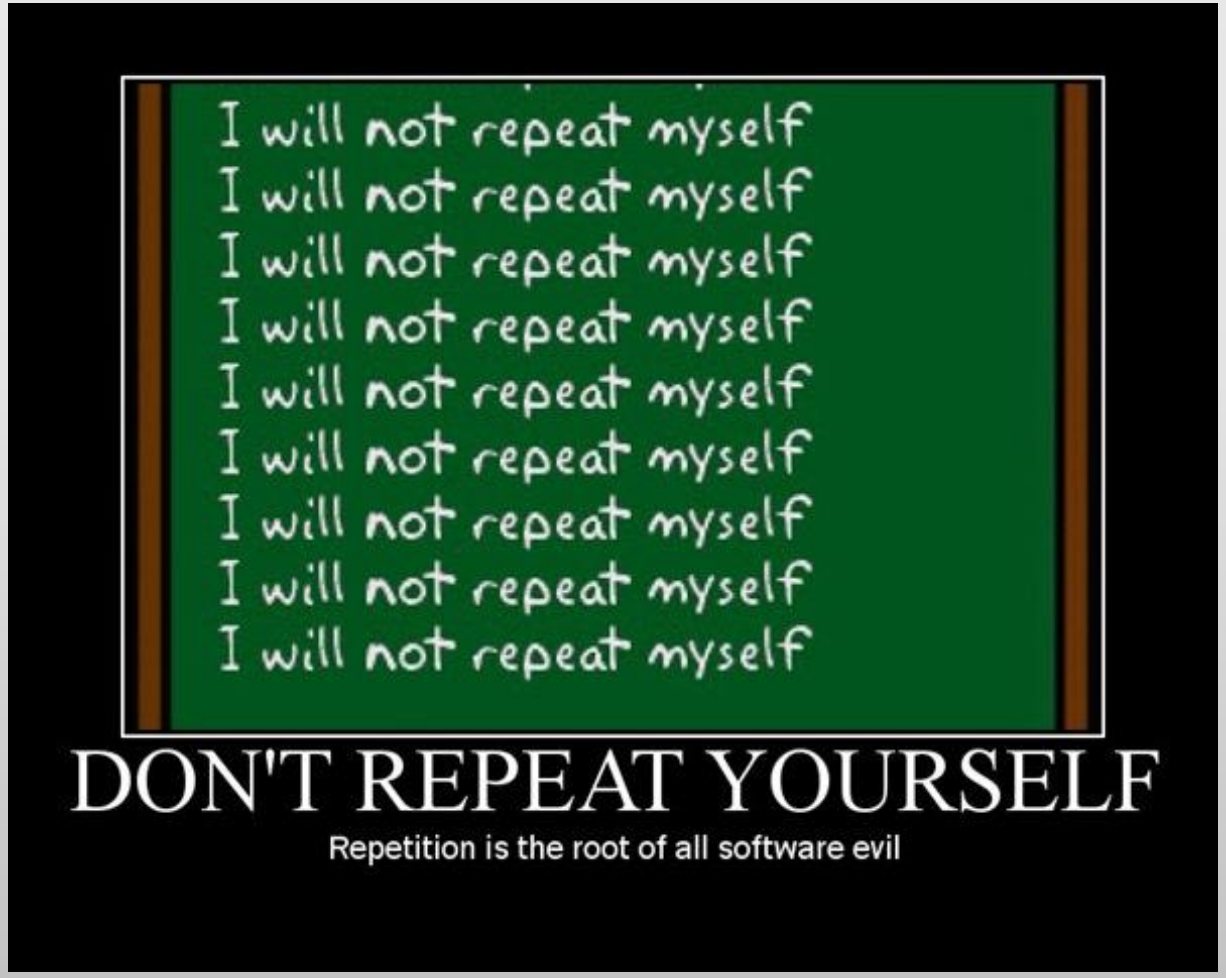
Better

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

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

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

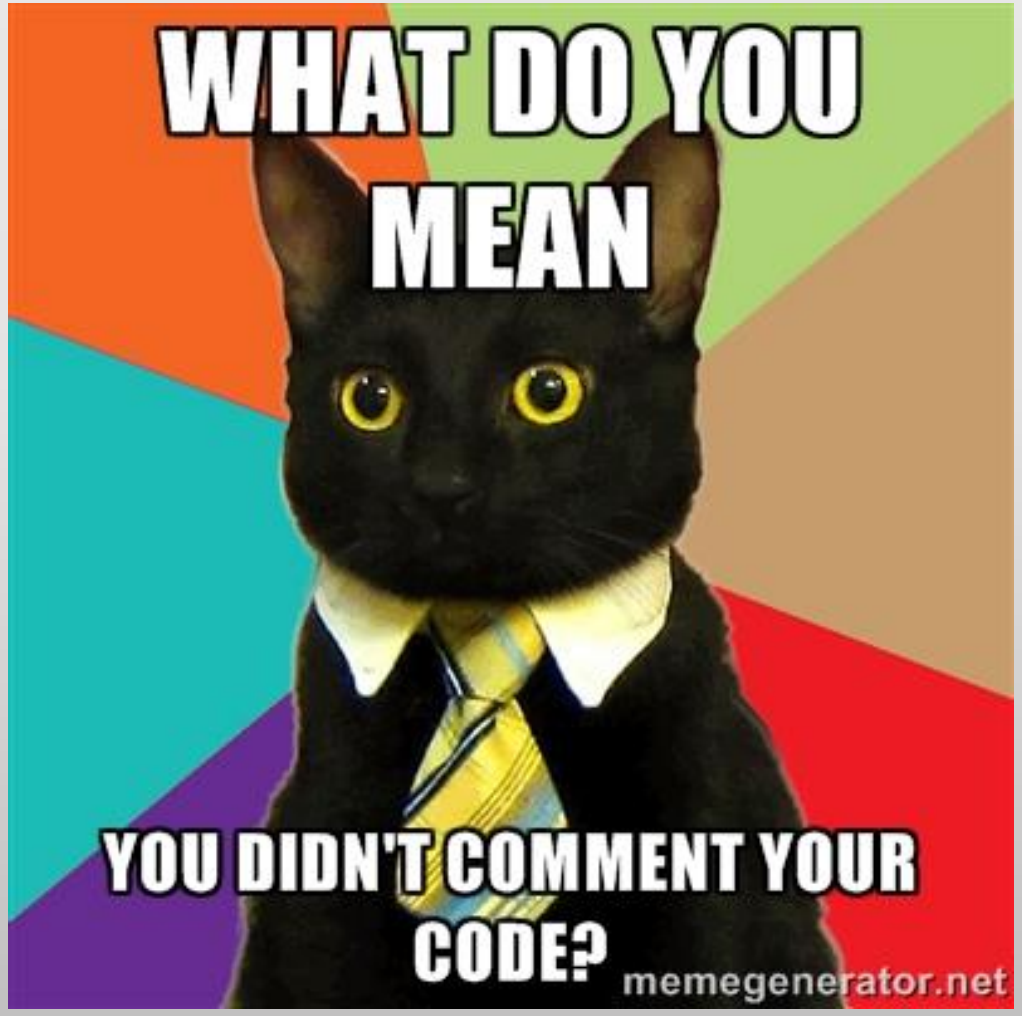
```
if (account.isValid()) {
  /* ... */
}

/* ... */

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

Better

COMMENTS


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<Member>();
}
```

REDUNDANT COMMENTS

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

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;
  }
```

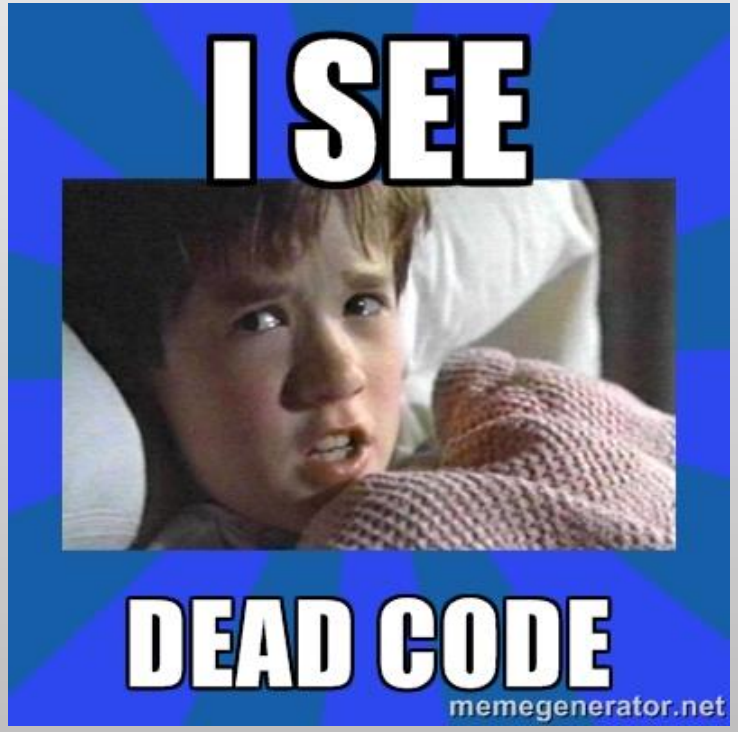
COMMENTED-OUT CODE

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;
}

private static final int TIMEOUT = 28800; // 2 hours in seconds
```

MISLEADING COMMENTS

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

private static final int TIMEOUT = 28800; // 2 hours in seconds
```

Comments are **lies** waiting to happen

MISLEADING COMMENTS

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

private static final int TIMEOUT = 28800; // 2 hours in seconds
```

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 have 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 have the same name more than once.
Set<String> uniqueNames = new HashSet<String>(names);
```

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

Public API JavaDocs

Public API JavaDocs

```
public static void commit() {
    if (isTransactionActive()) {
        getSession().getTransaction().commit();
    }
}

public stat
    if (isT
        get
    }
}

public stat
    try {
        rol
    } catch
    }
}

public stat
    if (star
        commit();
    }
}
```

void org.hibernate.Transaction.commit()

Commit this transaction. This might entail a number of things depending on the context:

- If this transaction is the `initiator`, `Session.flush` the `Session` with which it is associated (unless `Session` is in `FlushMode.MANUAL`).
- If this transaction is the `initiator`, commit the underlying transaction.
- Coordinate various callbacks

Throws:

HibernateException - Indicates a problem committing the transaction.

TODO COMMENTS

TODO COMMENTS

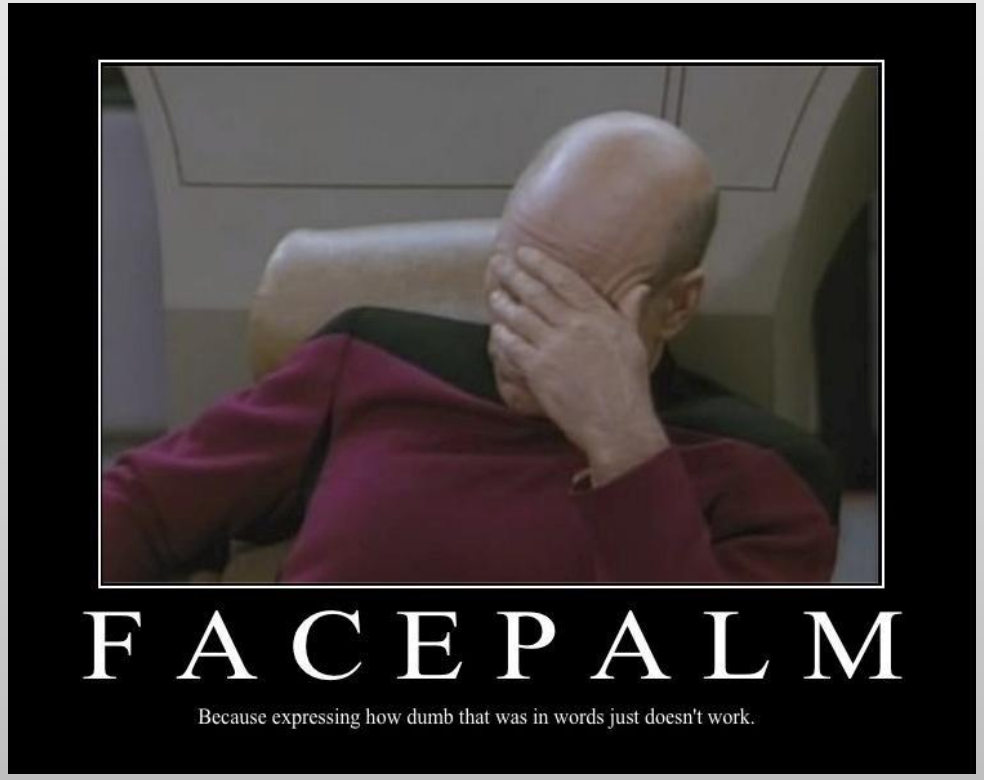
```
// TODO AISFWK-718: This needs to be refactored when we
// migrate to the new binding process
public ViewModel bindToViewModel(Model domainModel) {
  /* ... */
}
```

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...

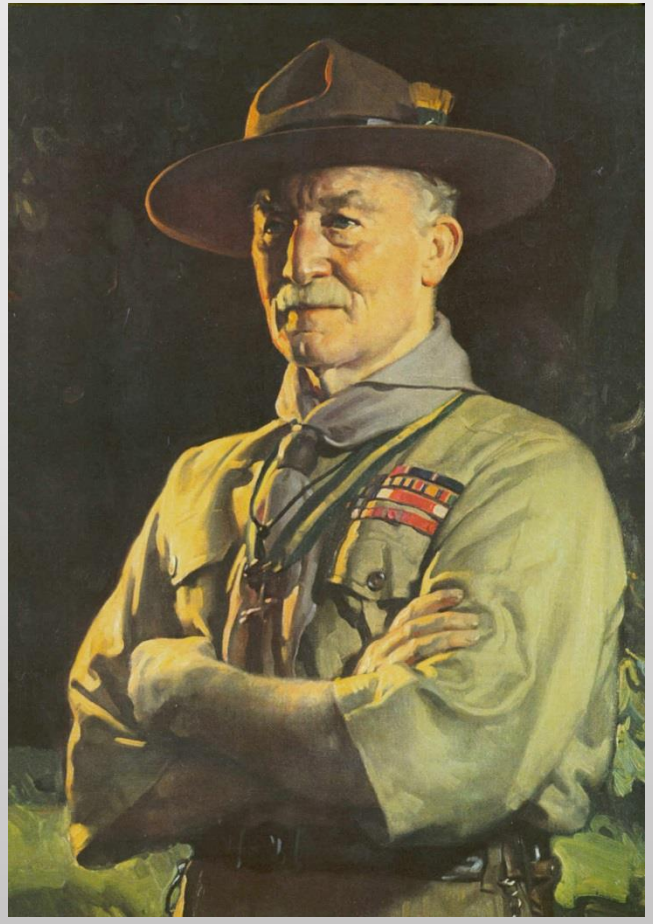
- ... **revealing intent** and **avoiding disinformation**
- ... **expressing** oneself in code
- ... authoring for the **readers**
- ... **refactoring** and **cleaning** existing code
- ... **caring** about your code
- ... being **professional**

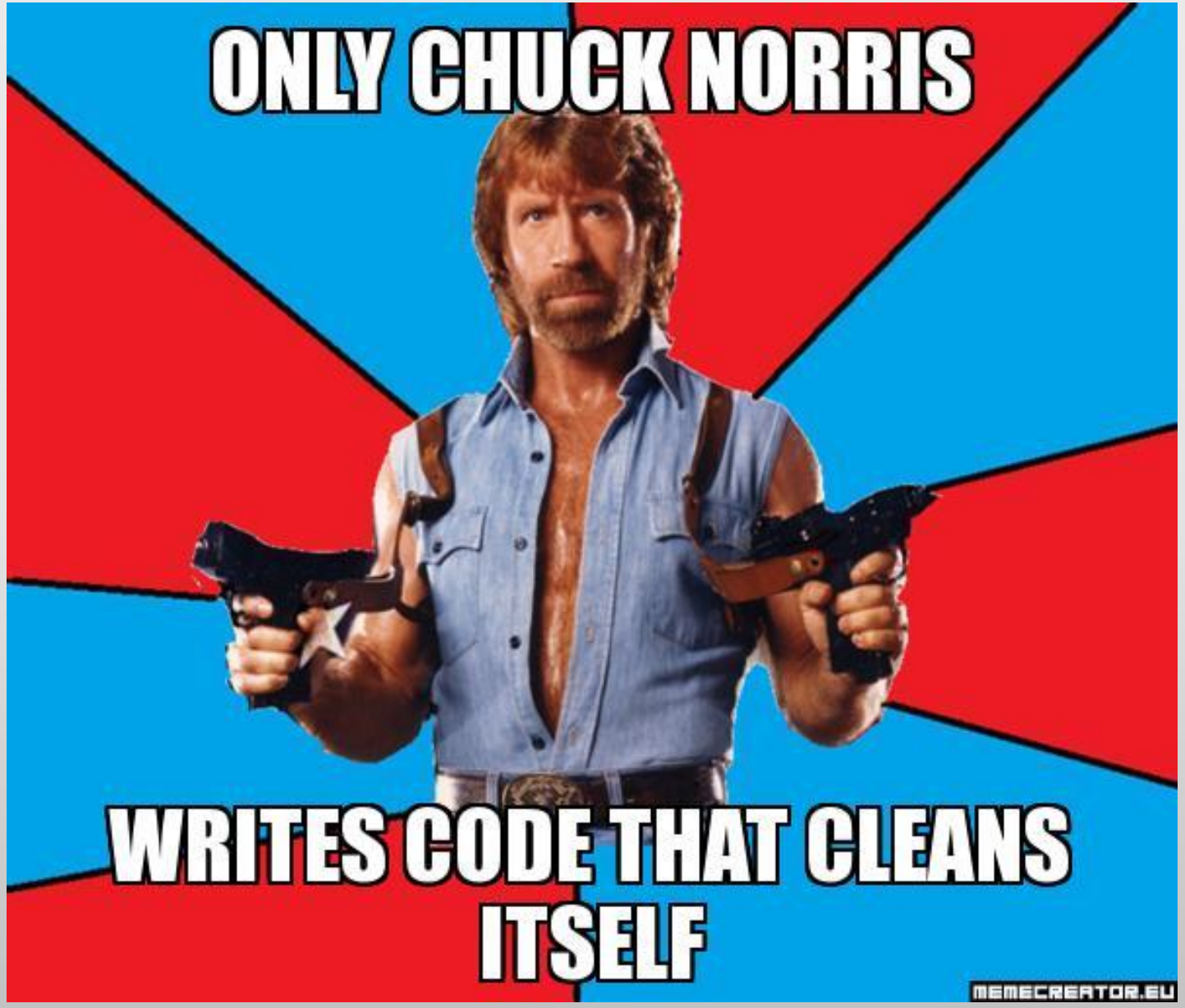
And there is a lot more to it!

THE BOY SCOUT RULE

"Leave the campground cleaner than you found it."

—Inspired by Robert Baden Powell




ONLY CHUCK NORRIS
WRITES CODE THAT CLEANS ITSELF
MEMECREATOR.EU


THANK
YOU

QUESTIONS?

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