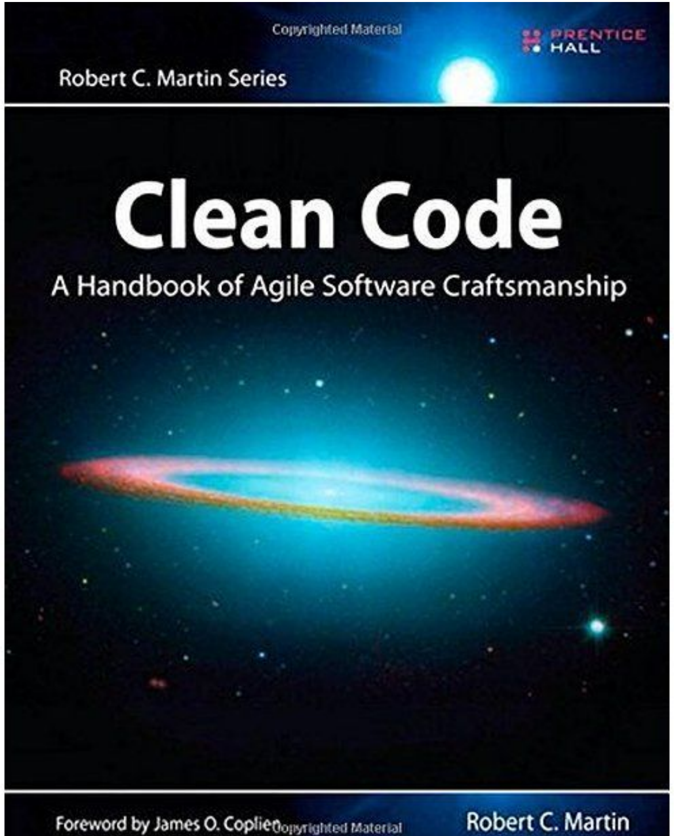
Clean Code

Mateusz Staniak
Wrocław, 28 XI 2019

Based on


Copyrighted Material
PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Copyrighted Material
Robert C. Martin

Code clean: what & why

Clean code

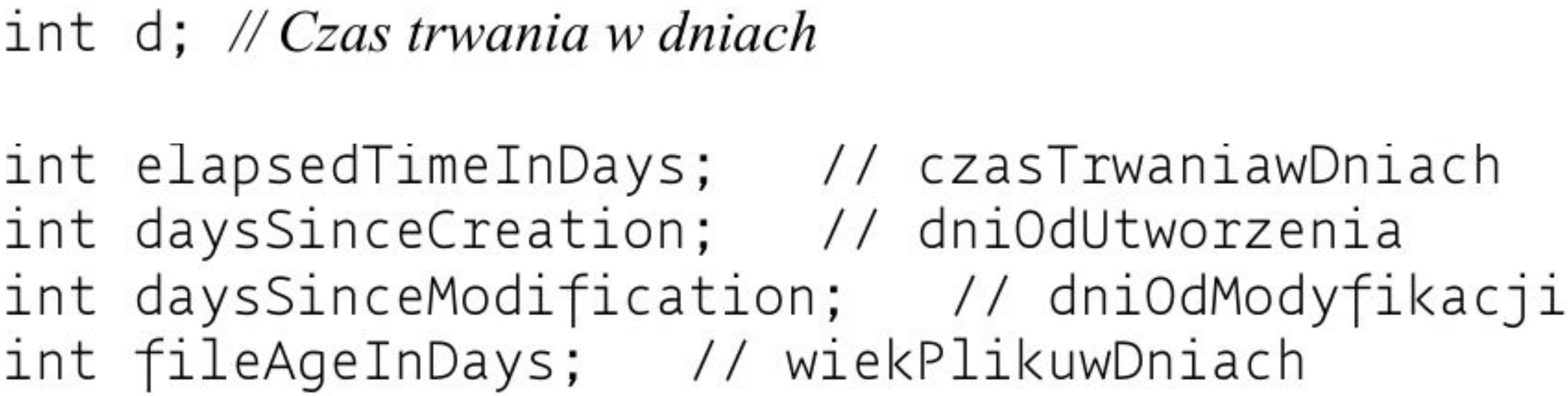
1. Easy to understand
    a. readable
    b. functions and classes have clear responsibilities
    c. relationships between objects, functions, etc are comprehensible
2. Easy to change
    a. extendable
    b. unit tests

The cost of messy code

1. Time wasted on
   a. reading and understanding
   b. refactoring
2. Slow (or no) development
   a. changes require big rewrites
   b. changes may break the existing code
3. The code is prone to bugs

Naming objects

Describe the intentions



```
int d;  // Czas trwania w dniach

int elapsedTimeInDays;    // czasTrwaniawDniach
int daysSinceCreation;    // dniOdUtworzenia
int daysSinceModification;    // dniOdModyfikacji
int fileAgeInDays;    // wiekPlikuwDniach
```

Avoid disinformation

Name	**Problem**
accountsList	(is it really a list?)
XYZControllerForEfficientHandlingOfStrings XYZControllerForEfficientStorageOfStrings	too similar
lOst (1 or l or I? 0 or O?)	confusing letters
a1, a2, a3	not distinguishable enough

More guidelines

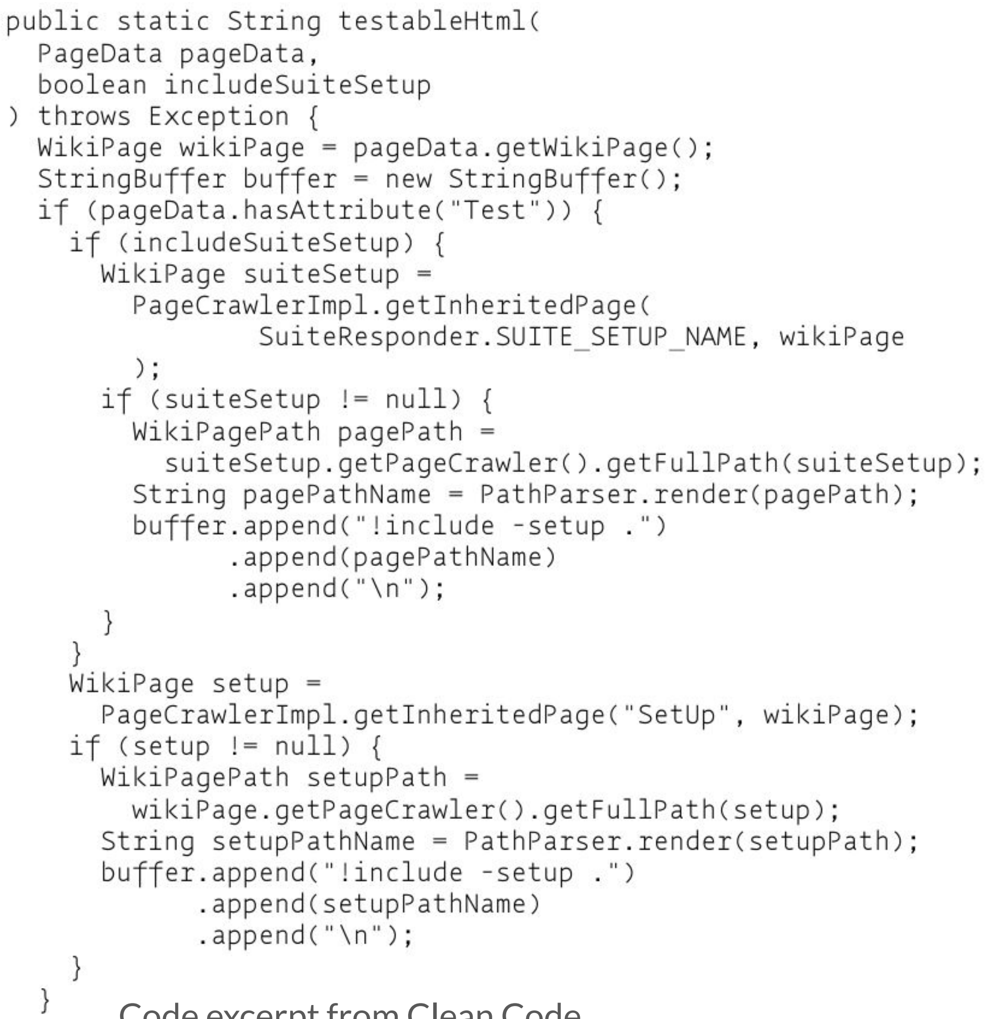
- Name word = one concept
- Easy pronunciation
- Use domain-specific terms
- Don’t try to be funny
- Class names - nouns, function/method names - verbs

Functions

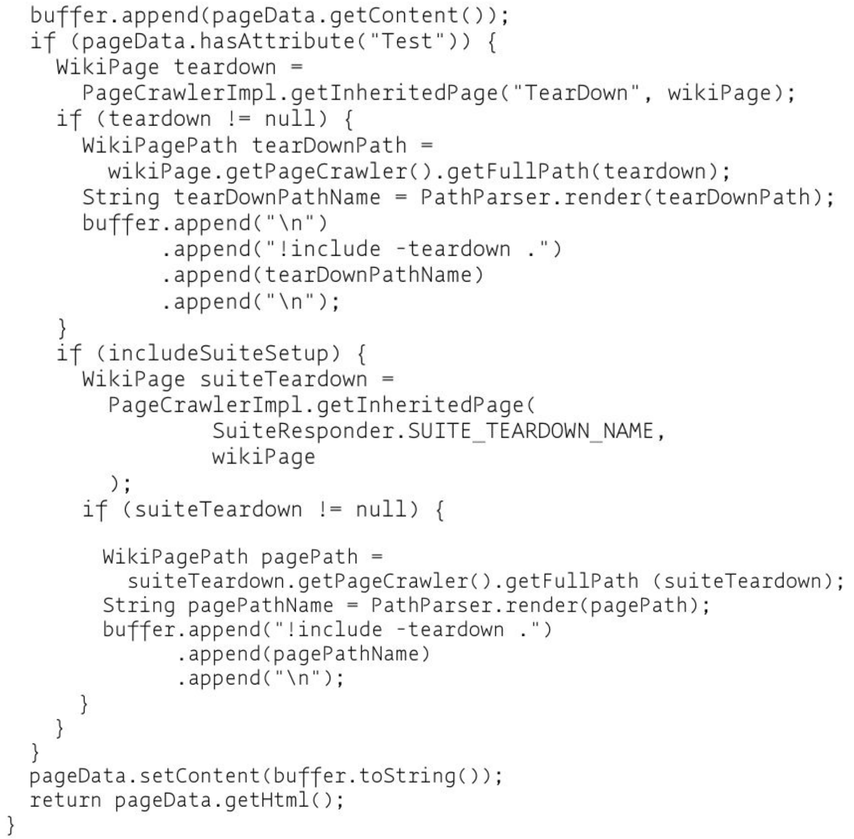
Functions

1. Small
2. Single responsibility
3. Single level of abstraction

In particular: separate computations from output

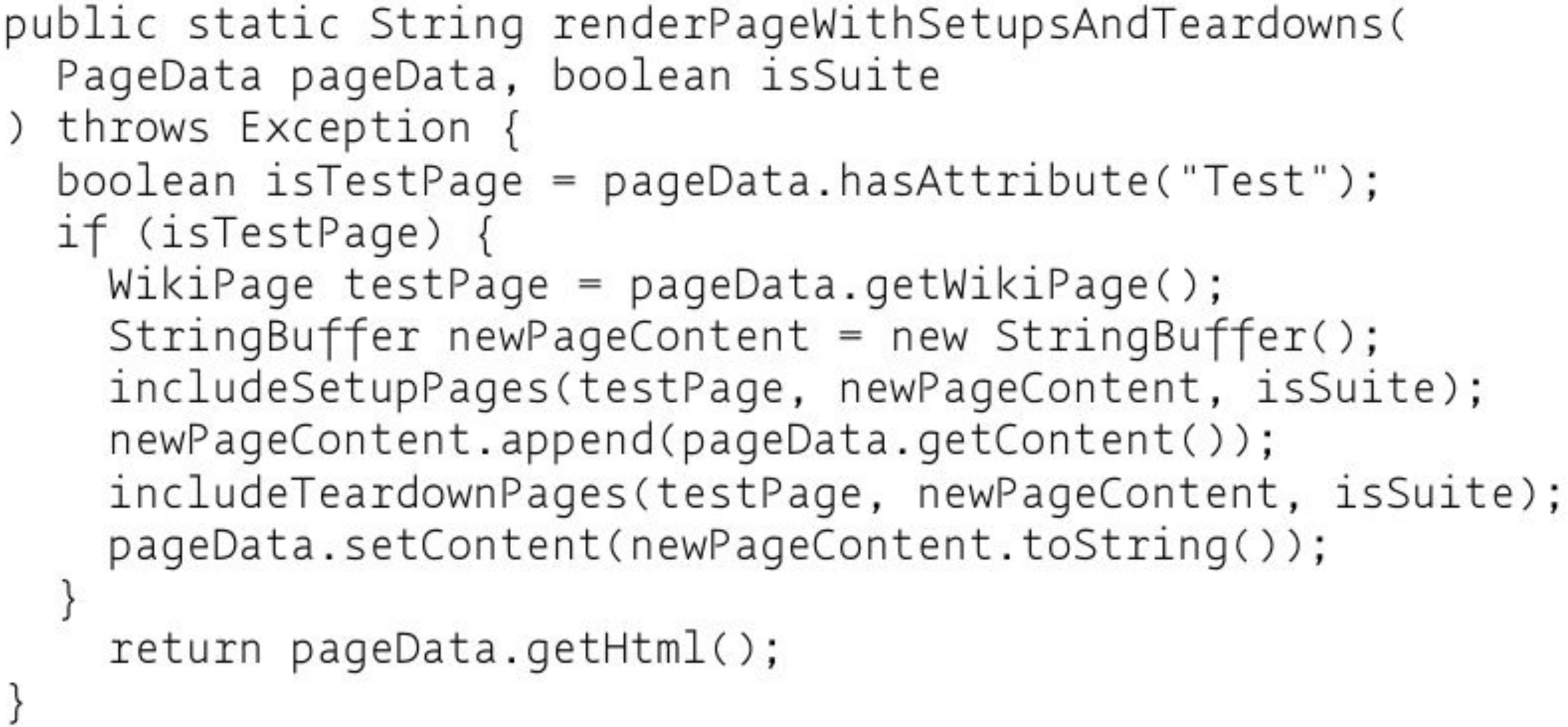


```
public static String testableHtml(
  PageData pageData,
  boolean includeSuiteSetup
) throws Exception {
  WikiPage wikiPage = pageData.getWikiPage();
  StringBuffer buffer = new StringBuffer();
  if (pageData.hasAttribute("Test")) {
    if (includeSuiteSetup) {
      WikiPage suiteSetup =
        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();
}
```

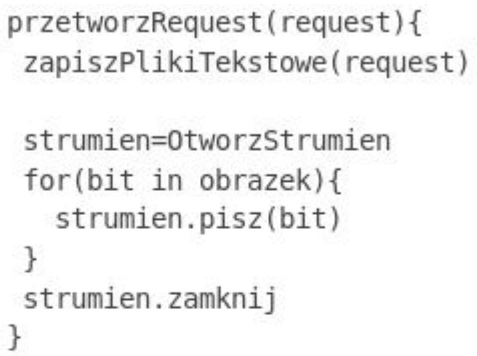
Code excerpt from Clean Code



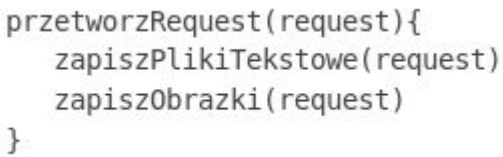
```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite
) throws Exception {
  boolean isTestPage = pageData.hasAttribute("Test");
  if (isTestPage) {
    WikiPage testPage = pageData.getWikiPage();
    StringBuffer newPageContent = new StringBuffer();
    includeSetupPages(testPage,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

Code excerpt from Clean Code

Single level of abstraction

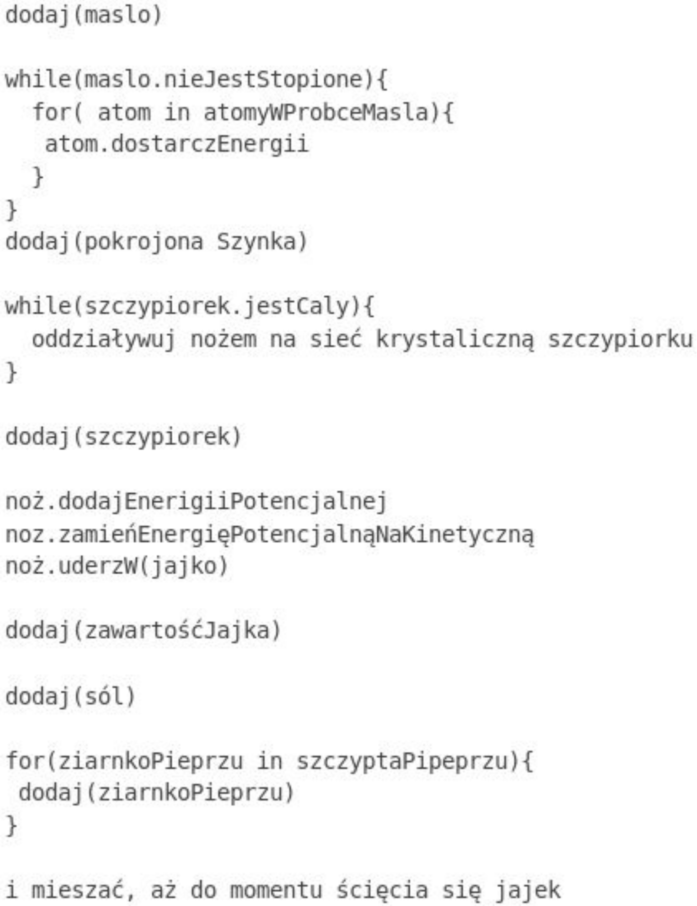


```
przetworzRequest(request){
 zapiszPlikiTekstowe(request)

 strumien=OtworzStrumien
 for(bit in obrazek){
   strumien.pisz(bit)
 }
 strumien.zamknij
}
```



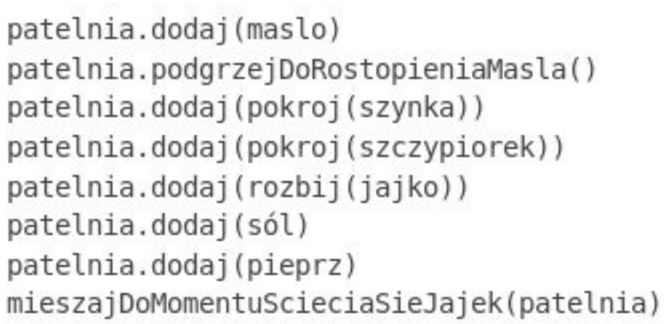
```
przetworzRequest(request){
   zapiszPlikiTekstowe(request)
   zapiszObrazki(request)
}
```

http://pawelwlodarski.blogspot.com/2011/02/poziomy-abstrakcji.html



```
dodaj(maslo)

while(maslo.nieJestStopione){
  for( atom in atomyWProbceMasla){
   atom.dostarczEnergii
  }
}
dodaj(pokrojona Szynka)

while(szczypiorek.jestCaly){
  oddziaływuj nożem na sieć krystaliczną szczypiorku
}

dodaj(szczypiorek)

noż.dodajEnerigiiPotencjalnej
noz.zamieńEnergięPotencjalnąNaKinetyczną
noż.uderzW(jajko)

dodaj(zawartośćJajka)

dodaj(sól)

for(ziarnkoPieprzu in szczyptaPipeprzu){
 dodaj(ziarnkoPieprzu)
}

i mieszać, aż do momentu ścięcia się jajek
```

http://pawelwlodarski.blogspot.com/2011/02/poziomy-abstrakcji.html



```
patelnia.dodaj(maslo)
patelnia.podgrzejDoRostopieniaMasla()
patelnia.dodaj(pokroj(szynka))
patelnia.dodaj(pokroj(szczypiorek))
patelnia.dodaj(rozbij(jajko))
patelnia.dodaj(sól)
patelnia.dodaj(pieprz)
mieszajDoMomentuScieciaSieJajek(patelnia)
```

http://pawelwlodarski.blogspot.com/2011/02/poziomy-abstrakcji.html

Aby dołączyć konfiguracje i rozbiory, dołączamy konfigurację, następnie zawartość strony testowej, po czym rozbiory.

Aby dołączyć konfigurację, dołączamy konfigurację zestawu, jeżeli jest to zestaw, a następnie zwykłą konfigurację.

Aby dołączyć konfigurację zestawu, wyszukujemy w hierarchii nadrzędnej stronę „SuiteSetUp” i dodajemy instrukcję `include` *ze ścieżką do tej strony.*

Aby przeszukać hierarchię nadrzędną…

Functions

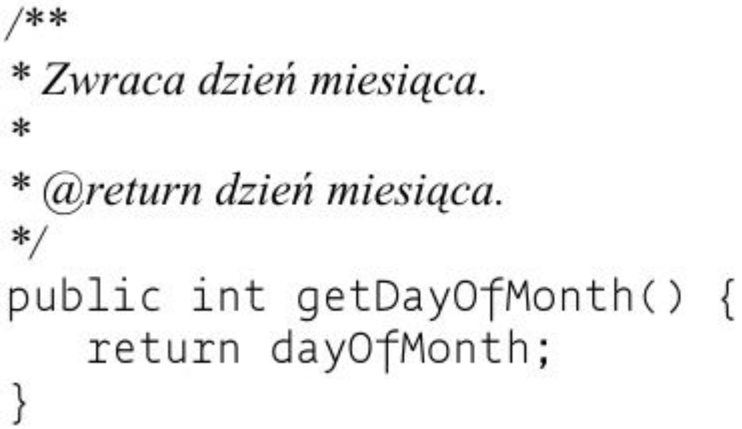
- Code should be read top to bottom, each function on a lower level of abstraction
- Functions should be small
- Functions should do just one thing (and do it well)
- Single level of abstraction
- Functions should have 1-3 parameters
- **Similar rules apply to classes**

Comments

- Clean code should explain itself
- Comments won’t mask a messy code

Exceptions:

- TODO/FIXME comments
- Comments explaining reasons for specific implementation details

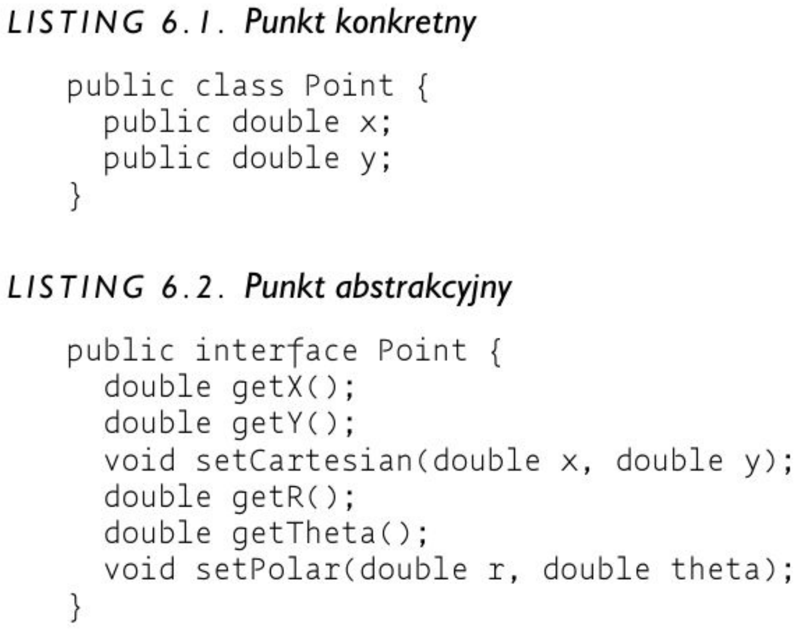


```
/**
* Zwraca dzień miesiąca.
*
* @return dzień miesiąca.
*/
public int getDayOfMonth() {
    return dayOfMonth;
}
```

More

Objects vs data structures

- Objects hide data, but have an interface
- Data structures show data, but lack an interface



LISTING 6.1. Punkt konkretny

```
public class Point {
  public double x;
  public double y;
}
```

LISTING 6.2. Punkt abstrakcyjny

```
public interface Point {
  double getX();
  double getY();
  void setCartesian(double x, double y);
  double getR();
  double getTheta();
  void setPolar(double r, double theta);
}
```

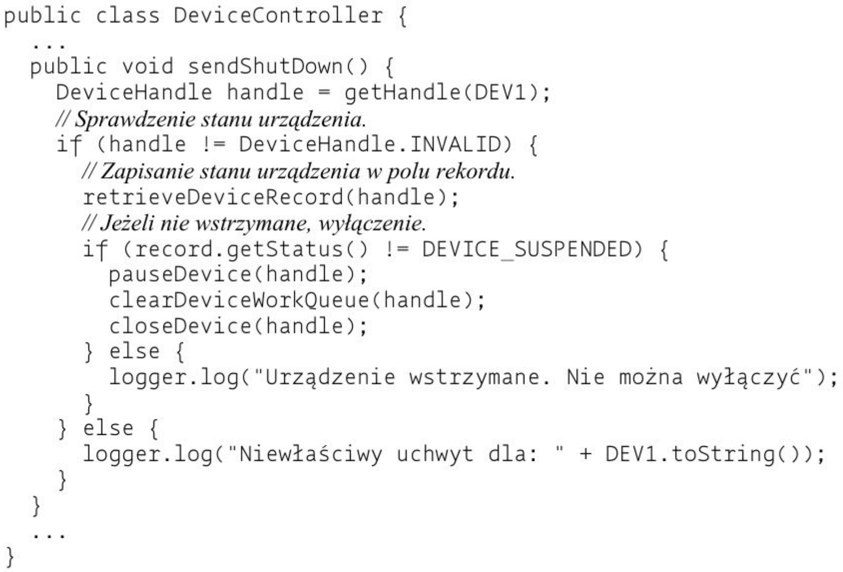
Test-Driven Development

Write tests before (production) code

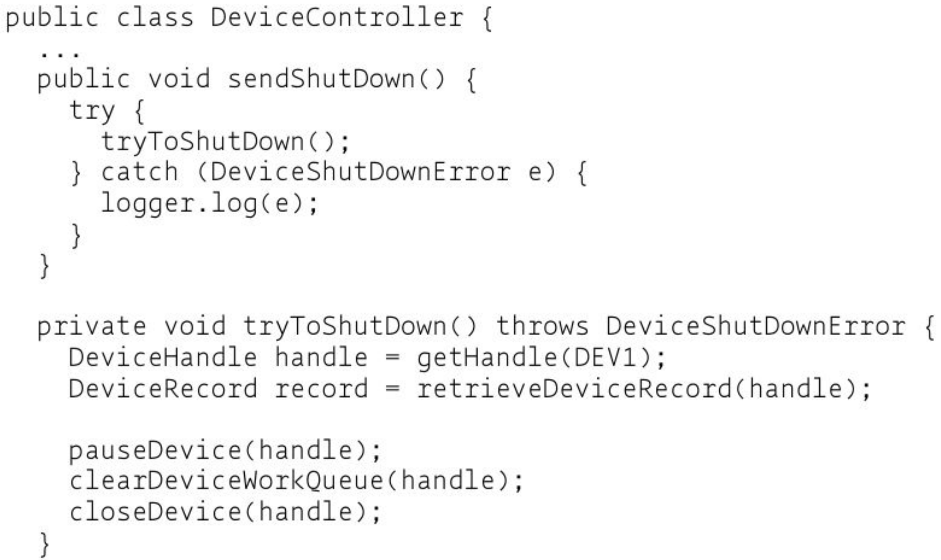
Three rules of TDD:

1. You are not allowed to write any production code unless it is to make a failing unit test pass.
2. You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.
3. You are not allowed to write any more production code than is sufficient to pass the one failing unit test.

Handling exceptions



```
public class DeviceController {
  ...
  public void sendShutDown() {
    DeviceHandle handle = getHandle(DEV1);
    // Sprawdzenie stanu urządzenia.
    if (handle != DeviceHandle.INVALID) {
      // Zapisanie stanu urządzenia w polu rekordu.
      retrieveDeviceRecord(handle);
      // Jeżeli nie wstrzymane, wyłączenie.
      if (record.getStatus() != DEVICE_SUSPENDED) {
        pauseDevice(handle);
        clearDeviceWorkQueue(handle);
        closeDevice(handle);
      } else {
        logger.log("Urządzenie wstrzymane. Nie można wyłączyć");
      }
    } else {
      logger.log("Niewłaściwy uchwyt dla: " + DEV1.toString());
    }
  }
  ...
}
```

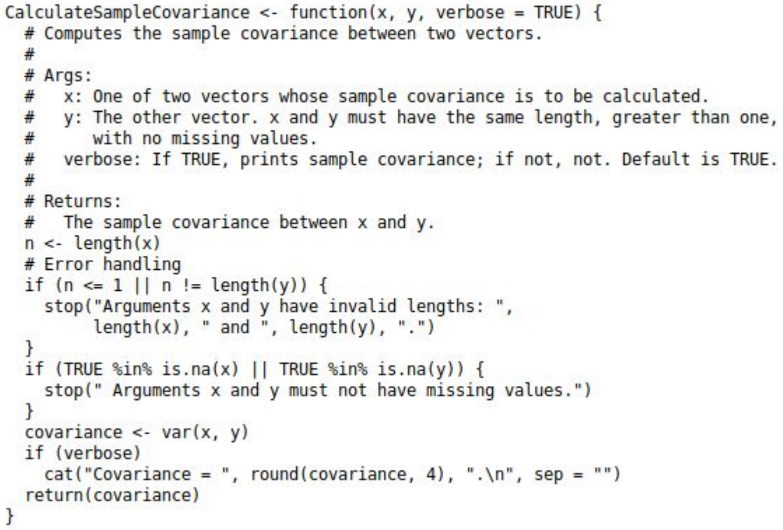


```
public class DeviceController {
  ...
  public void sendShutDown() {
    try {
      tryToShutDown();
    } catch (DeviceShutDownError e) {
      logger.log(e);
    }
  }

  private void tryToShutDown() throws DeviceShutDownError {
    DeviceHandle handle = getHandle(DEV1);
    DeviceRecord record = retrieveDeviceRecord(handle);

    pauseDevice(handle);
    clearDeviceWorkQueue(handle);
    closeDevice(handle);
  }
```

Clean code in R

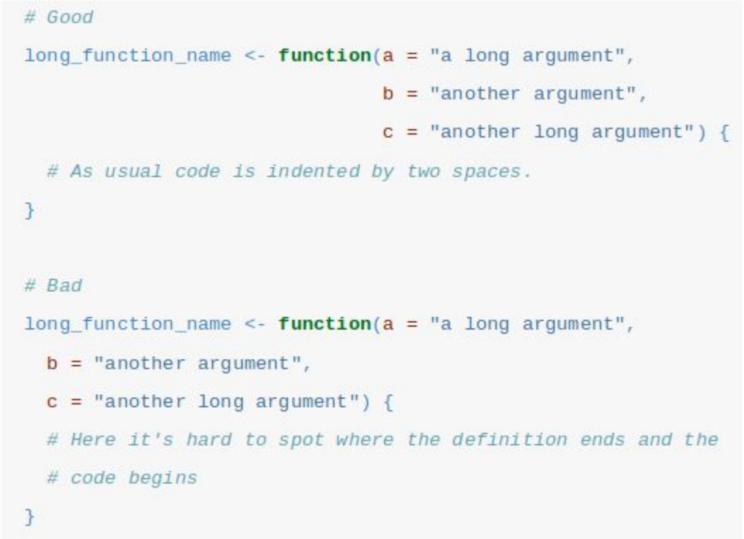
Code style

Google style: https://web.stanford.edu/class/cs109l/unrestricted/resources/google-style.html



```
CalculateSampleCovariance <- function(x, y, verbose = TRUE) {
  # Computes the sample covariance between two vectors.
  #
  # Args:
  #   x: One of two vectors whose sample covariance is to be calculated.
  #   y: The other vector. x and y must have the same length, greater than one,
  #      with no missing values.
  #   verbose: If TRUE, prints sample covariance; if not, not. Default is TRUE.
  #
  # Returns:
  #   The sample covariance between x and y.
  n <- length(x)
  # Error handling
  if (n <= 1 || n != length(y)) {
    stop("Arguments x and y have invalid lengths: ",
         length(x), " and ", length(y), ".")
  }
  if (TRUE %in% is.na(x) || TRUE %in% is.na(y)) {
    stop(" Arguments x and y must not have missing values.")
  }
  covariance <- var(x, y)
  if (verbose)
    cat("Covariance = ", round(covariance, 4), ".\n", sep = "")
  return(covariance)
}
```

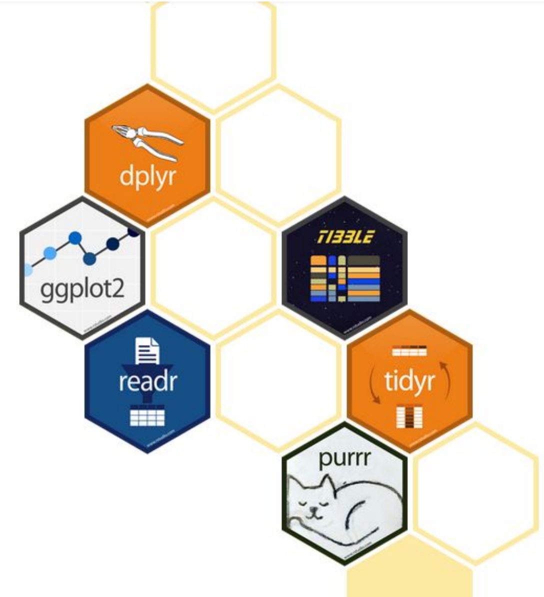
Tidyverse style: https://style.tidyverse.org/



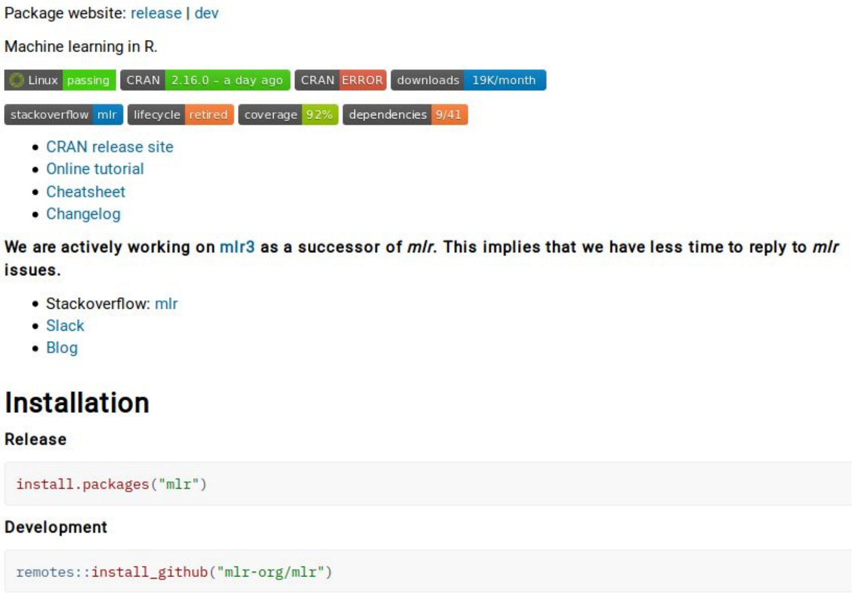
```
# Good
long_function_name <- function(a = "a long argument",
                               b = "another argument",
                               c = "another long argument") {
  # As usual code is indented by two spaces.
}

# Bad
long_function_name <- function(a = "a long argument",
  b = "another argument",
  c = "another long argument") {
  # Here it's hard to spot where the definition ends and the
  # code begins
}
```

Examples of good projects

Tidyverse (https://tidyverse.org)



mlr: Machine Learning in R (https://mlr.mlr-org.com)



Package website: release | dev

Machine learning in R.

Linux passing | CRAN 2.16.0 - a day ago | CRAN ERROR | downloads 19K/month

stackoverflow mlr | lifecycle retired | coverage 92% | dependencies 9/41

- CRAN release site
- Online tutorial
- Cheatsheet
- Changelog

We are actively working on mlr3 as a successor of *mlr*. This implies that we have less time to reply to *mlr* issues.

- Stackoverflow: mlr
- Slack
- Blog

Installation

Release

```
install.packages("mlr")
```

Development

```
remotes::install_github("mlr-org/mlr")
```

Tools for keeping the code clean in R

- Unit tests: testthat package (https://cran.r-project.org/package=testthat)
- Code style:
  - styler (http://styler.r-lib.org/)
  - lintr (https://github.com/jimhester/lintr)
- Clean code checks: cleanr package (https://cran.r-project.org/package=cleanr)

Hands-on

Let’s talk code:

https://github.com/StatsIMUWr/R_Workshops/blob/master/Meetings/11_28_CleanCode/hands_on.R