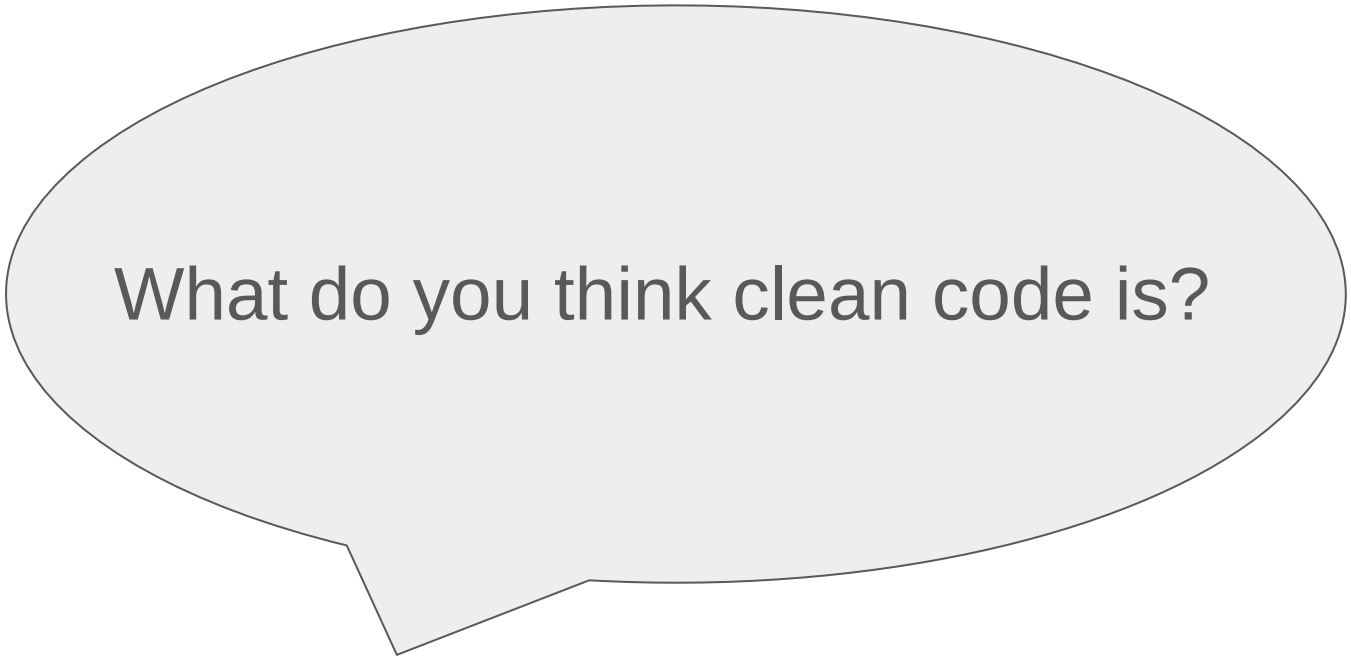
PITECH+PLUS

Clean code

Cernau Laura Diana

dev @ PITECH PLUS

Contents


What do you think clean code is?

Grady Booch, author of *Object Oriented Analysis and Design with Applications*

Clean code is simple and direct. Clean code reads like well-written prose. Clean code never obscures the designer's intent but rather is full of crisp abstractions and straightforward lines of control.

Bjarne Stroustrup, inventor of C++ and author of *The C++ Programming Language*

I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well.

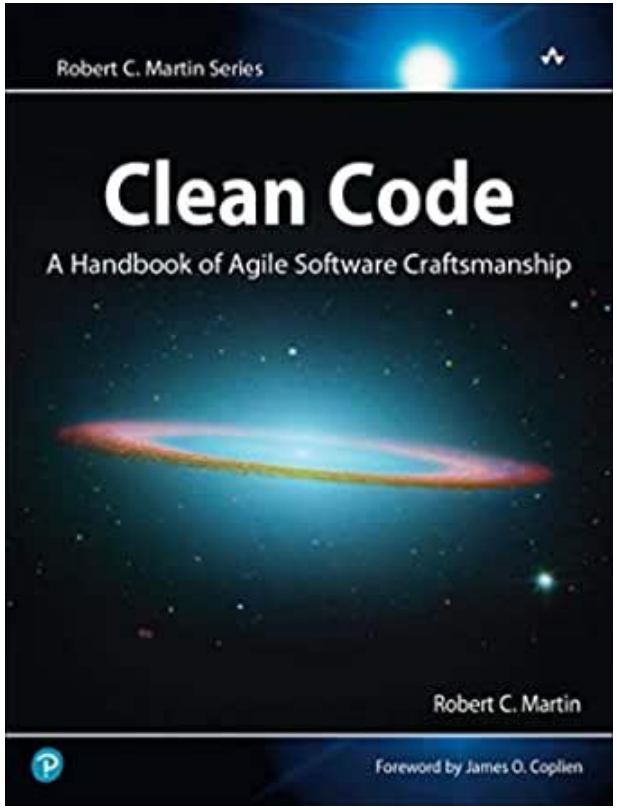
"Big" Dave Thomas, founder of OTI, godfather of the Eclipse strategy

Clean code can be read, and enhanced by a developer other than its original author. It has unit and acceptance tests. It has meaningful names. It provides one way rather than many ways for doing one thing. It has minimal dependencies, which are explicitly defined, and provides a clear and minimal API. Code should be literate since depending on the language, not all necessary information can be expressed clearly in code alone.

Clean Code

“Clean code is code that is easy to understand and easy to change”

The book named “Clean code” was written by Robert C. Martin and represents a set of rules that can make your code easier to understand, have a better structure and is easy to change/refactor.



Meaningful names

Rules for creating meaningful names:

- Use intention-revealing names
- Use searchable names
- Class names should be formed by a noun or noun phrase like *Customer*, *Job*, *Account, UserProfile* etc
- Method names should contain a verb: save(), makePayment(), deleteUser()
- Pick one word per concept

PITECH+PLUS

Meaningful names - examples

```
int d;      //elapsed time in days
```

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

PITECH+PLUS

Meaningful names - examples

```
public List<int[]> getThem(List<int[]> theList) {
    List<int[]> list1 = new ArrayList<>();
    for (int[] x : theList)
        if (x[0] == 4)
            list1.add(x);
    return list1;
}
```

PITECH+PLUS

Meaningful names - examples

```
public static final int STATUS_VALUE = 0;
public static final int FLAGGED = 4;

public List<int[]> getFlaggedCells(List<int[]> gameBoard) {
    List<int[]> flaggedCells = new ArrayList<>();
    for (int[] cell : gameBoard)
        if (cell[STATUS_VALUE] == FLAGGED)
            flaggedCells.add(cell);

    return flaggedCells;
}
```

PITECH+PLUS

Functions

In a common program structure, the most important rules when writing a function are:

- Functions should be small
- Do one thing
- One level of abstraction per function
- Function arguments
- Avoid side effects
- Prefer exceptions to returning error codes
- Don’t repeat yourself (DRY)

Functions - examples

```
public static String renderPageWithSetupsAndTeardowns(PageData pageData, boolean isSuite) {
    boolean isTestPage = pageData.hasAttribute( t: "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();
}
```

Functions - examples

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

Functions - examples

```
public class UserValidator {
    private Cryptographer cryptographer;

    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;
    }
}
```

PITECH+PLUS

Comments

Good Comments

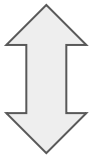
- Informative comments
- Explanation of intent
- Clarification
- TODO comments

Bad Comments

- Redundant comments
- Misleading comments
- Position markers
- Commented out code
- Too much information

Comments - examples

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



```
if (employee.isEligibleForFullBenefits())
```

PITECH+PLUS

Formatting

It is about communication, and communication is professional developer’s first order of business.

- Indentation
- The newspaper metaphor
    - Vertical openness between concepts
    - Vertical density
    - Vertical distance
    - Dependent functions
    - Conceptual affinity
    - Horizontal openness and density

Formatting - examples

```
public class ReporterConfig {
    private String m_className;
    private List<Property> m_properties = new ArrayList<Property>();

    public void addProperty(Property property) {
        m_properties.add(property);
    }
}
```

```
private void measureLine(String line) {
    lineCount++;
    int lineSize = line.length();
    totalChars += lineSize;
    lineWidthHistogram.addLine(lineSize, lineCount);
    recordWidestLine(lineSize);
}
```

PITECH+PLUS

Error handling

- Use exceptions rather than return codes
- Provide context with exceptions
- Don’t pass null

Error handling - examples

```
public class MetricsCalculator {
    public double xProjection(Point p1, Point p2) {
        return (p1.x - p2.x) * 10 ;
    }
}
```

Error handling - examples

```
public class MetricsCalculator {
    public double xProjection(Point p1, Point p2) {
        if (p1 == null || p2 == null) {
            throw new IllegalArgumentException("Invalid argument for MetricsCalculator.xProjection");
        }

        return (p1.x - p2.x) * 10;
    }
}
```

PITECH+PLUS

General tips

- use suitable naming for your variables, classes, functions
- respect the coding standards established inside the team
- be consistent with the indentation style
- remove unused or commented code
- do not overload the source code with unnecessary comments
- don’t add too many arguments for a function
- refactor the code mercilessly

PITECH+PLUS

Thank you!