Clean code

Good practices for readability and maintainability

Rafael Sousa Herves

Contents

What is clean code?

What is clean code?

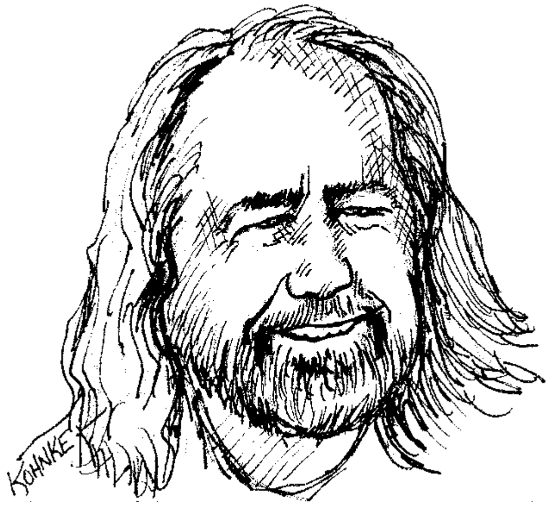
Bjarne Stroustrup, inventor of C++

“I like my code to be elegant and efficient.”

“The logic should be straightforward to make it hard for bugs to hide.”

“Clean code does one thing well.”

What is clean code?

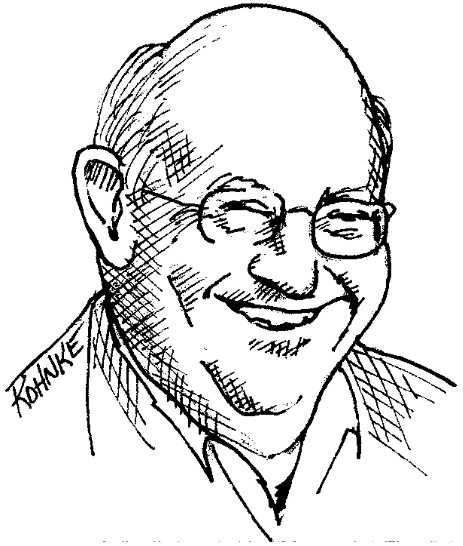


"Clean code is simple and direct."

"Reads like well-written prose."

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

What is clean code?



“Clean code can be read, and enhanced by other developers.”

“It has meaningful names.”

“It provides one way for doing one thing.”

“... minimal dependencies, … minimal API”

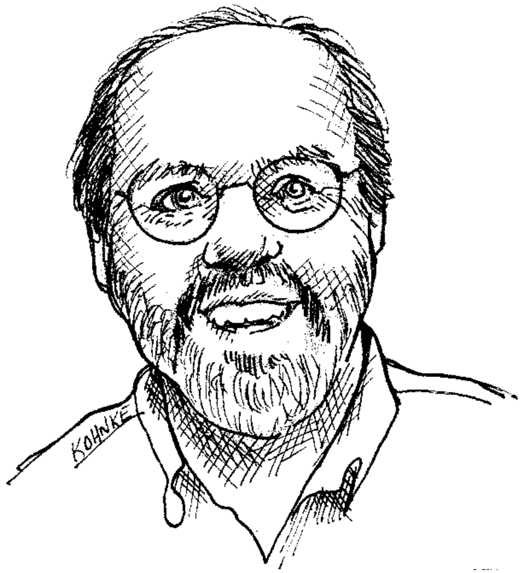
Dave Thomas, founder of OTI, godfather of the Eclipse strategy

What is clean code?

“Clean code always looks like it was written by someone who cares.”

Michael Feathers, author of *Working Effectively with Legacy Code*

What is clean code?



“On clean code, each routine you read turns out to be pretty much what you expected.”

Ward Cunningham, inventor of Wiki, inventor of Fit, coinventor of eXtreme Programming.
Motive force behind Design Patterns. Smalltalk and OO thought leader.

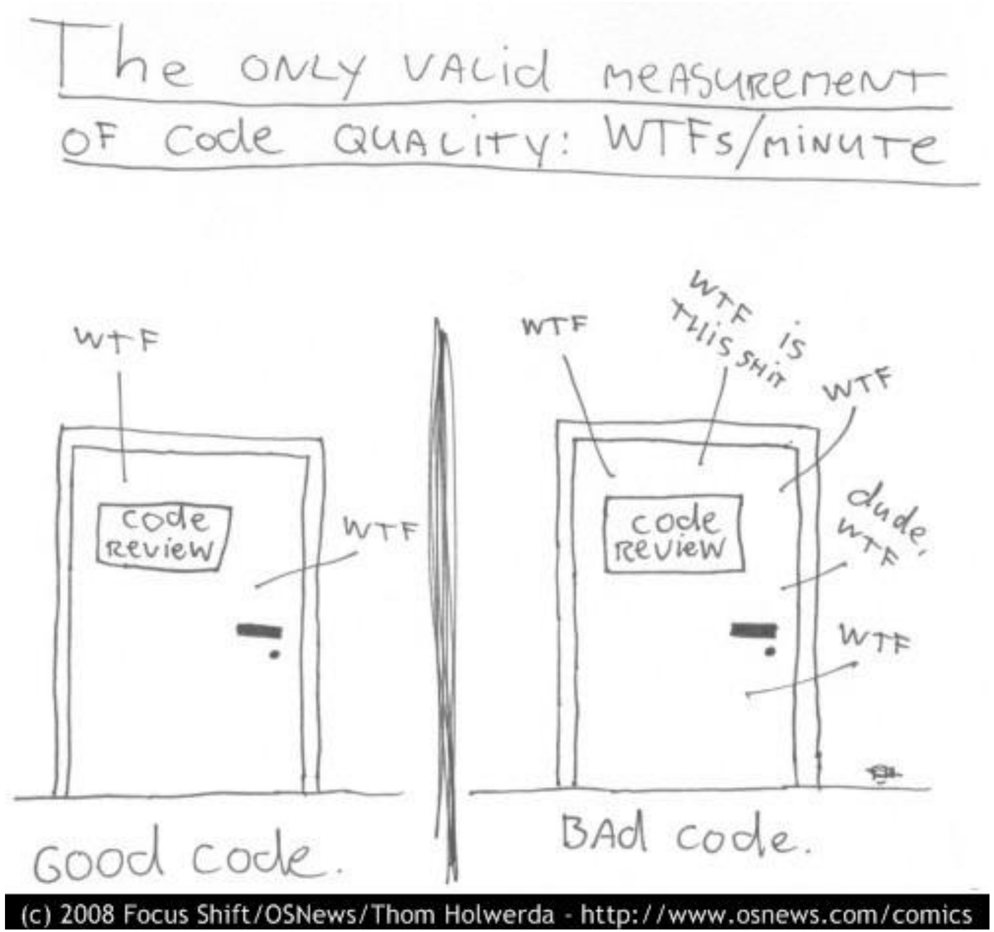
Concretely

```java
public void sendMessage(Object o) {
    //...
}
```

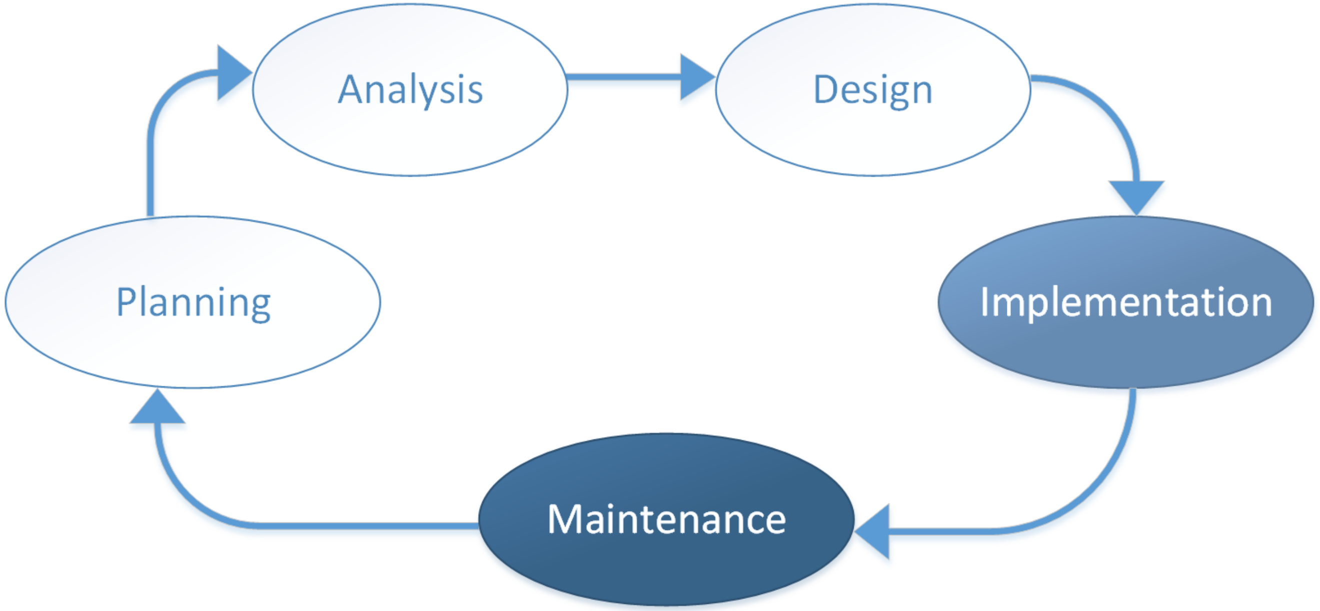
```java
public ChatMessage(String message, Object o) {
    //...
}
```

```java
public Message(Object o) {
    //...
}
```

How to measure code quality?


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

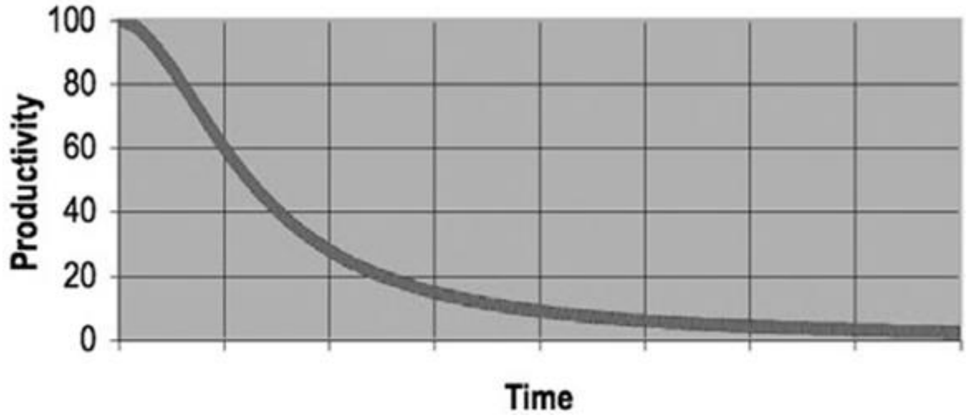
Why clean code?



Software development life cycle

Consequences of bad code

- The mess grows
- Productivity lowers
- Application cannot be maintained → game over



Naming

Naming - Meaningful names

Bad

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

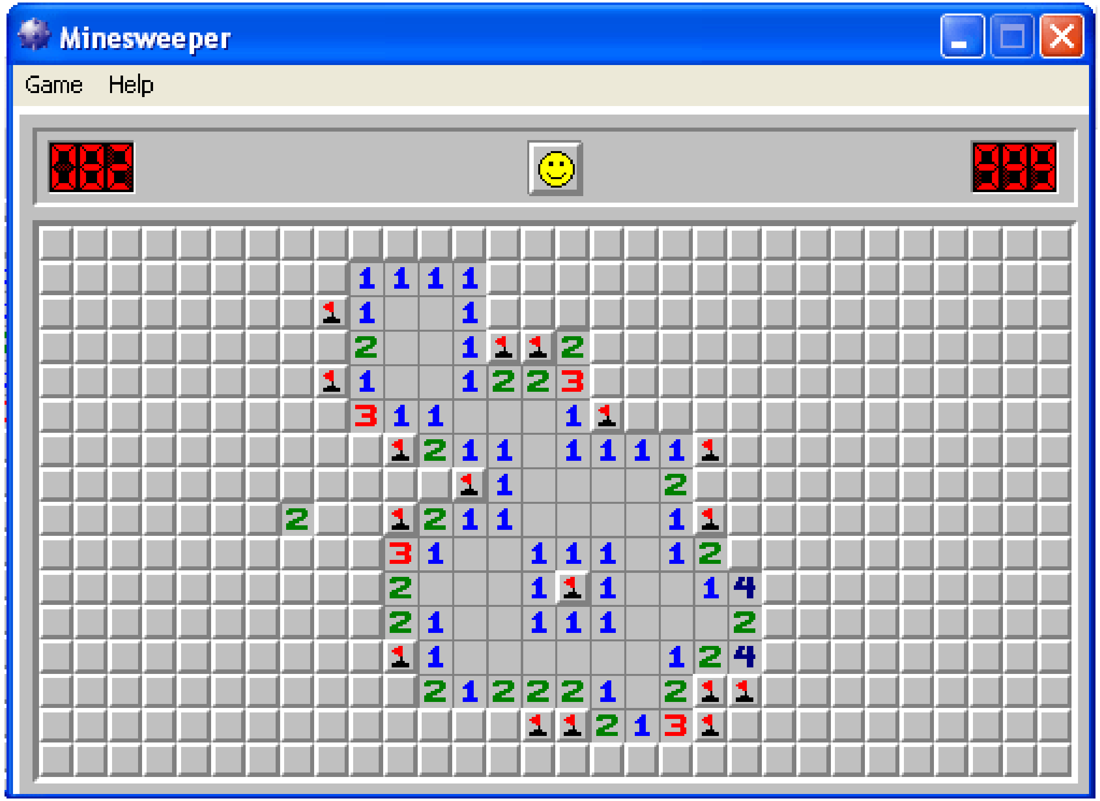
Good

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

Naming - Intention revealing names

Bad

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


Minesweeper
Game
Help

Naming - Intention revealing names

Bad

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

Good

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


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

Naming - Pronounceable names

Bad

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

Good

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

Naming - Avoid encodings

Bad

```
private String m_dsc;

PhoneNumber phoneString;
// name not changed when type changed!
```

Good

```
private String description;

PhoneNumber phone;
```

Naming - Add meaningful context

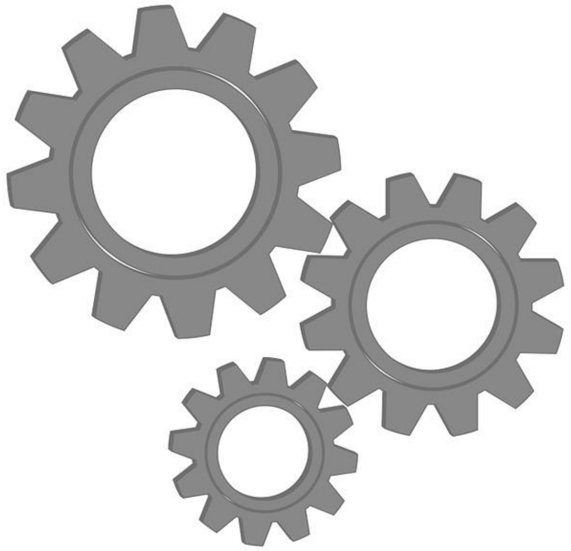
```
firstName, lastName, street, city, state, zipcode

// better
addrStreet, addrCity, addrState

// best
Address customerAddress = new Address();
customerAddress.getState();
```

Functions

$f(x) =$



Functions

1. Small
2. Smaller than that

    No bigger than the screen

        80’s: 24 lines, 80 columns

        Today: 100 lines, 150 columns

    Ideally: 2, 3, 4 lines (divide and rule!)

Functions

3. Do one thing

Functions should do one thing.
They should do it well.
They should do it only.

Functions

4. One level of abstraction

High	`getHtml();`
Medium	`String pagePathName = PathParser.render(pagePath);`
Low	`.append("\n");`

Functions

5. Descriptive names

$$\left.\begin{array}{l}\text{Spend time}\\ \text{Easy with IDE's}\\ \text{No fear of long names}\end{array}\right\}\quad\text{favors refactor}$$

Functions

7. Reduce number of arguments

    0: ideal

    1: ok

    2: ok

    3: justify

    4 or more: avoid

Functions

7. Reduce number of arguments

- wrapping objects

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

- instance variables

```
void appendText(StringBuilder builder, String toAppend)

private StringBuilder builder;
void appendText(String toAppend)
```

Comments

“Don’t comment bad code—rewrite it.”
—Brian W. Kernighan and P. J. Plaugher

Comments

- Can be helpful or damaging
- Inaccurate comments are worse than no comments at all
- Used to compensate failure expressing with code

  → refactor instead
- They lie
- Must have them, but minimize them

Comments

Express yourself in code

Bad

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

Good

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

Comments

Noise

```
/**
* Add a CD to the collection
*
* @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);
}
```

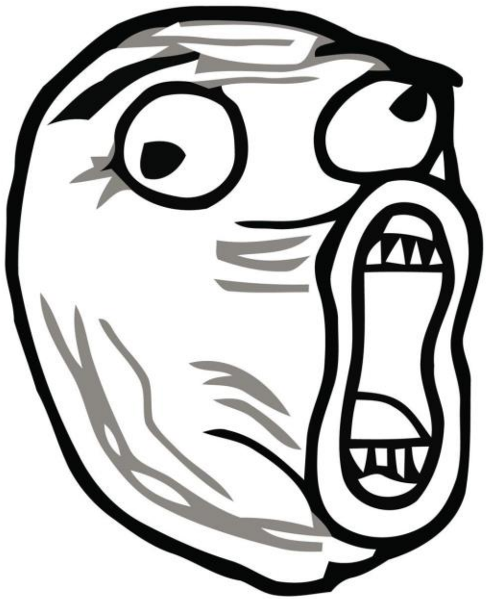
Comments

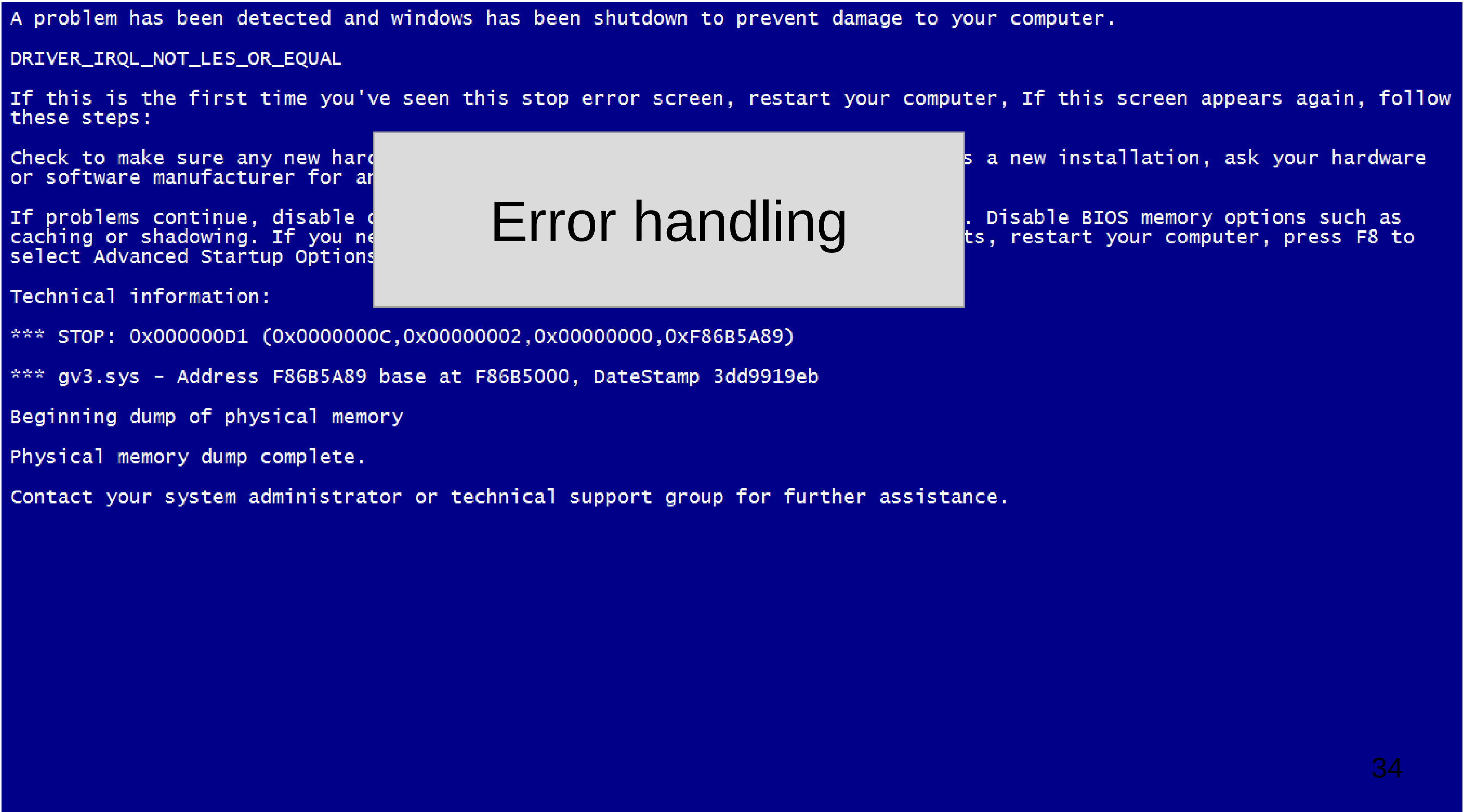
Scary noise

```
/** The name. */
private String name;

/** The version. */
private String version;

/** The day of the month. */
private int dayOfMonth;
```





```
A problem has been detected and windows has been shutdown to prevent damage to your computer.

DRIVER_IRQL_NOT_LES_OR_EQUAL

If this is the first time you've seen this stop error screen, restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new harc                                s a new installation, ask your hardware
or software manufacturer for ar

If problems continue, disable c                                . Disable BIOS memory options such as
caching or shadowing. If you ne                                ts, restart your computer, press F8 to
select Advanced Startup Options

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

*** gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd9919eb

Beginning dump of physical memory

Physical memory dump complete.

Contact your system administrator or technical support group for further assistance.
```

Error handling

Error handling - Use Exceptions, not returning 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;
}
```

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

Error handling - Extract Try/Catch blocks

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

```java
public void delete(Page page) {
   try {
       deletePageAndAllReferences(page);
   }
   catch (Exception e) {
       logger.log(e.getMessage());
   }
}


private void deletePageAndAllReferences(Page page) throws
Exception {
   deletePage(page);
   registry.deleteReference(page.name);
   configKeys.deleteKey(page.name.makeKey());
}
```

Error handling - Don’t return Null

- Error-prone
- Forces to have null-checks everywhere

```java
public void registerItem(Item item) {
    if (item != null) {
        ItemRegistry registry = peristentStore.getItemRegistry();
        if (registry != null) {
            Item existing = registry.getItem(item.getID());
            if (existing != null && existing.getBillingPeriod() != null) {
                if (existing.getBillingPeriod().hasRetailOwner()) {
                    existing.register(item);
                }
            }
        }
    }
}
```

Error handling - Don’t return Null

Alternatives:

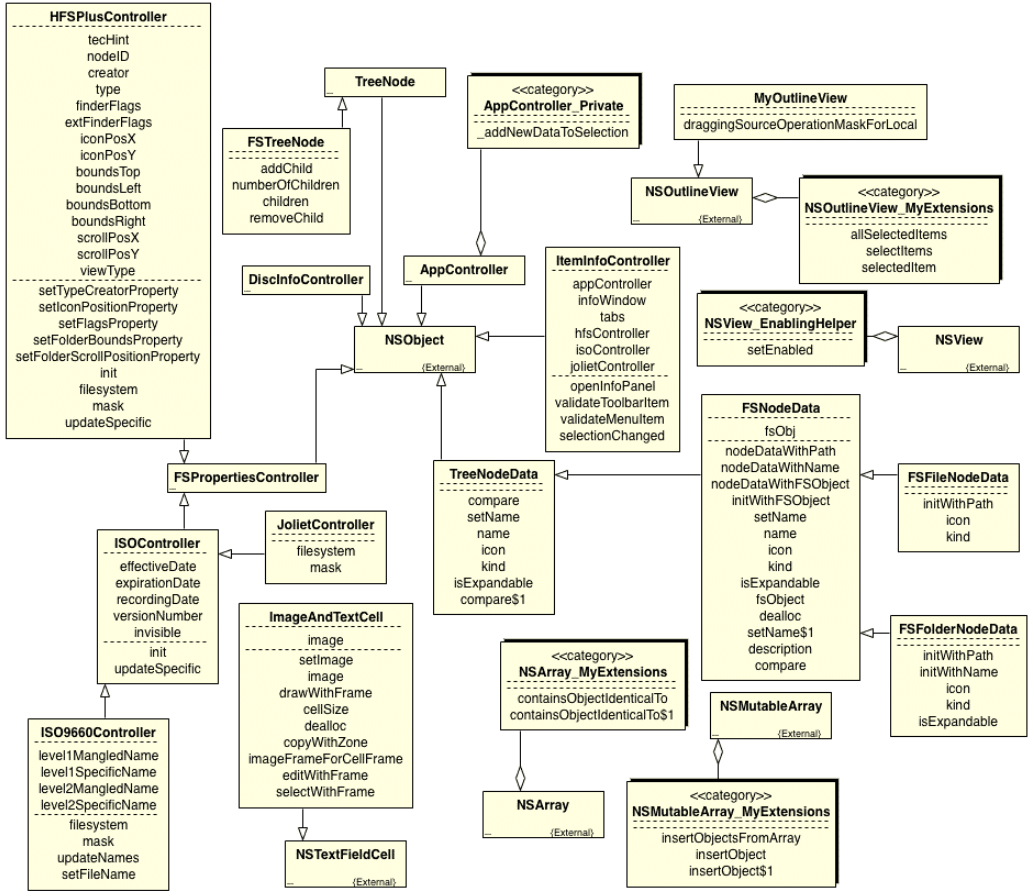
- Return empty collections

```
public List<Employee> getEmployees() {
   // if( .. there are no employees .. )
   return Collections.emptyList();
}
```

- Return Optional

```
public Optional<Player> getWinner() {
   // if( .. there is no winner .. )
   return Optional.empty();
}
```

Classes


HFSPlusController
tecHint
nodeID
creator
type
finderFlags
extFinderFlags
iconPosX
iconPosY
boundsTop
boundsLeft
boundsBottom
boundsRight
scrollPosX
scrollPosY
viewType
setTypeCreatorProperty
setIconPositionProperty
setFlagsProperty
setFolderBoundsProperty
setFolderScrollPositionProperty
init
filesystem
mask
updateSpecific
TreeNode
FSTreeNode
addChild
numberOfChildren
children
removeChild
<<category>>
AppController_Private
_addNewDataToSelection
MyOutlineView
draggingSourceOperationMaskForLocal
NSOutlineView
{External}
<<category>>
NSOutlineView_MyExtensions
allSelectedItems
selectItems
selectedItem
DiscInfoController
AppController
ItemInfoController
appController
infoWindow
tabs
hfsController
isoController
jolietController
openInfoPanel
validateToolbarItem
validateMenuItem
selectionChanged
<<category>>
NSView_EnablingHelper
setEnabled
NSView
{External}
NSObject
{External}
FSNodeData
fsObj
nodeDataWithPath
nodeDataWithName
nodeDataWithFSObject
initWithFSObject
setName
name
icon
kind
isExpandable
fsObject
dealloc
setName$1
description
compare
FSPropertiesController
TreeNodeData
compare
setName
name
icon
kind
isExpandable
compare$1
FSFileNodeData
initWithPath
icon
kind
ISOController
effectiveDate
expirationDate
recordingDate
versionNumber
invisible
init
updateSpecific
JolietController
filesystem
mask
ImageAndTextCell
image
setImage
image
drawWithFrame
cellSize
dealloc
copyWithZone
imageFrameForCellFrame
editWithFrame
selectWithFrame
FSFolderNodeData
initWithPath
initWithName
icon
kind
isExpandable
<<category>>
NSArray_MyExtensions
containsObjectIdenticalTo
containsObjectIdenticalTo$1
NSMutableArray
{External}
ISO9660Controller
level1MangledName
level1SpecificName
level2MangledName
level2SpecificName
filesystem
mask
updateNames
setFileName
NSArray
{External}
<<category>>
NSMutableArray_MyExtensions
insertObjectsFromArray
insertObject
insertObject$1
NSTextFieldCell
{External}

Classes should be small

- One responsibility
- Denoted by the name

  Don’t use words such “Super”, “Manager”, “Processor”, etc.

- Brief description of the class without using words “and”, “or”, “but”, …

```java
public class SuperDashboard extends JFrame{
   public Component getLastFocusedComponent()
   public void setLastFocused(Component lastFocused)
   public int getMajorVersionNumber()
   public int getMinorVersionNumber()
   public int getBuildNumber()
}
```

```java
public class SuperDashboard extends JFrame {
   public String getCustomizerLanguagePath()
   public void setSystemConfigPath(String systemConfigPath)
   public String getSystemConfigDocument()
   public void resetDashboard()
   public boolean getGuruState()
   public boolean getNoviceState()
   public boolean getOpenSourceState()
   public void showObject(MetaObject object)
   public void showProgress(String s)
   public boolean isMetadataDirty()
   public void setIsMetadataDirty(boolean isMetadataDirty)
   public Component getLastFocusedComponent()
   public void setLastFocused(Component lastFocused)
   public void setMouseSelectState(boolean isMouseSelected)
   public boolean isMouseSelected()
   public LanguageManager getLanguageManager()
   public Project getProject()
   public Project getFirstProject()
   public Project getLastProject()
   public String getNewProjectName()
   public void setComponentSizes(Dimension dim)
   public String getCurrentDir()
   public void setCurrentDir(String newDir)
   public void updateStatus(int dotPos, int markPos)
   public Class[] getDataBaseClasses()
   public MetadataFeeder getMetadataFeeder()
   public void addProject(Project project)
   public boolean setCurrentProject(Project project)
   public boolean removeProject(Project project)
   // more and more ...
}
```

Classes - Single Responsibility Principle

Responsibility = Reason to change

- Classes should have only one reason to change

```
public class SuperDashboard extends JFrame{
   public Component getLastFocusedComponent()
   public void setLastFocused(Component lastFocused)
   public int getMajorVersionNumber()
   public int getMinorVersionNumber()
   public int getBuildNumber()
}
```

```
public class Version {
   public int getMajorVersionNumber()
   public int getMinorVersionNumber()
   public int getBuildNumber()
}
```

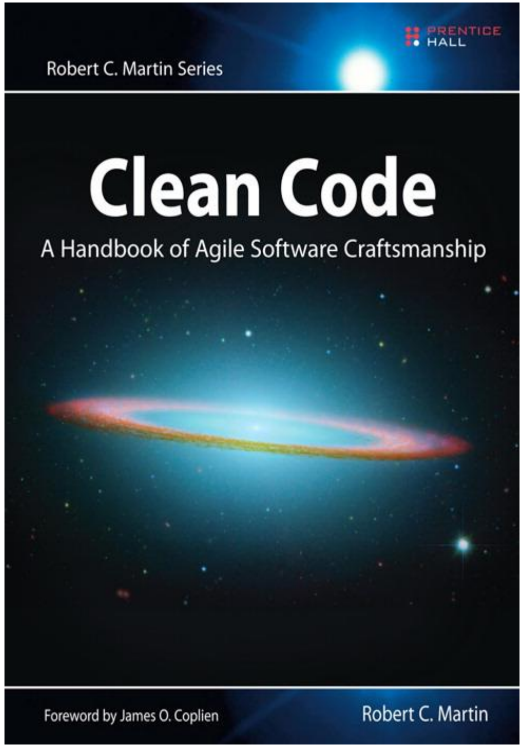
Conclusion

- Make code work and make code clean are two different activities.
- The more a system grows, the more difficult to maintain.
- Benefits on development, benefits in the future, including you.
- We are professionals, we should do it well, we should do it clean.

Reference book

Robert C. Martin,

"Clean code: A Handbook of Agile Software Craftsmanship"


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