

# THE ART OF CLEAN CODE

## – SOME TIPS

---

*Class 4, [venks@iiit.ac.in](mailto:venks@iiit.ac.in)*

**NOISY COMMENTS**

# Noisy Comments

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

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


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

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

# Noisy Comments

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

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



A delicate matter, requiring taste and judgement. I tend to err on the side of eliminating comments, for several reasons. First, if the code is clear, and uses good type names and variable names, it should explain itself. Second, comments aren't checked by the compiler, so there is no guarantee they're right, especially after the code is modified. A misleading comment can be very confusing. Third, the issue of typography: comments clutter code.



A common fallacy is to assume authors of incomprehensible code will somehow be able to express themselves lucidly and clearly in comments.

# VISUAL DESIGN OF CODE

“

A clean design is one that supports visual thinking so people can meet their informational needs with a minimum of conscious effort.





You convey information by the way you arrange a design's elements in relation to each other. This information is understood immediately, if not consciously, by the people viewing your designs.

# Visual Design of Code

```
public int howNotToLayoutAMethodHeader(int firstArgument,  
    String secondArgument)
```

```
public int ensureArgumentsAreAlignedLikeThis(  
    int firstArgument,  
    String secondArgument)
```

```
public int orEnsureArgumentsAreGroupedLikeThis(  
    int firstArgument, String secondArgument)
```

**Visual Property – An unifying list of arguments**

# Visual Design of Code

```
public int howNotToLayoutAMethodHeader(int firstArgument,  
String secondArgument)
```

```
public int ensureArgumentsAreAlignedLikeThis (  
int firstArgument,  
String secondArgument)
```

```
public int orEnsureArgumentsAreGroupedLikeThis (  
int firstArgument, String secondArgument)
```

**Visual Property – An unifying list of arguments**

# Visual Design of Code

```
public ResultType arbitraryMethodName (FirstArgumentType firs
                                     SecondArgumentType sec
                                     ThirdArgumentType thir
    LocalVariableType localVariable = method(firstArgument,
                                              secondArgument)
    if (localVariable.isSomething(thirdArgument,
                                  SOME_SHOUTY_CONSTANT)) {
        doSomethingWith(localVariable);
    }
    return localVariable.getSomething();
}
```

**Visual Property – Lesser lines of alignment/attention**

# Visual Design of Code

```
public ResultType arbitraryMethodName(  
    FirstArgumentType firstArgument,  
    SecondArgumentType secondArgument,  
    ThirdArgumentType thirdArgument) {  
    LocalVariableType localVariable =  
        method(firstArgument, secondArgument);  
    if (localVariable.isSomething(  
        thirdArgument, SOME_SHOUTY_CONSTANT)) {  
        doSomething(localVariable);  
    }  
    return localVariable.getSomething();  
}
```

# LEGO NAMING

## Lego Naming

```
public interface ConditionChecker  
{  
    boolean checkCondition();  
}
```

## Lego Naming

```
public interface ConditionChecker  
{  
    boolean checkCondition();  
}
```

```
public interface Condition  
{  
    boolean isTrue();  
}
```



ClassNotFoundException

EnumConstantNotPresentException

IllegalArgumentException

IllegalAccessException

IndexOutOfBoundsException

NegativeArraySizeException

NoSuchMethodException

TypeNotPresentException

UnsupportedOperationException

ClassNotFoundException

EnumConstantNotPresent

IllegalArgumentException

IllegalAccess

IndexOutOfBoundsException

NegativeArraySize

NoSuchMethod

TypeNotPresent

UnsupportedOperation

“

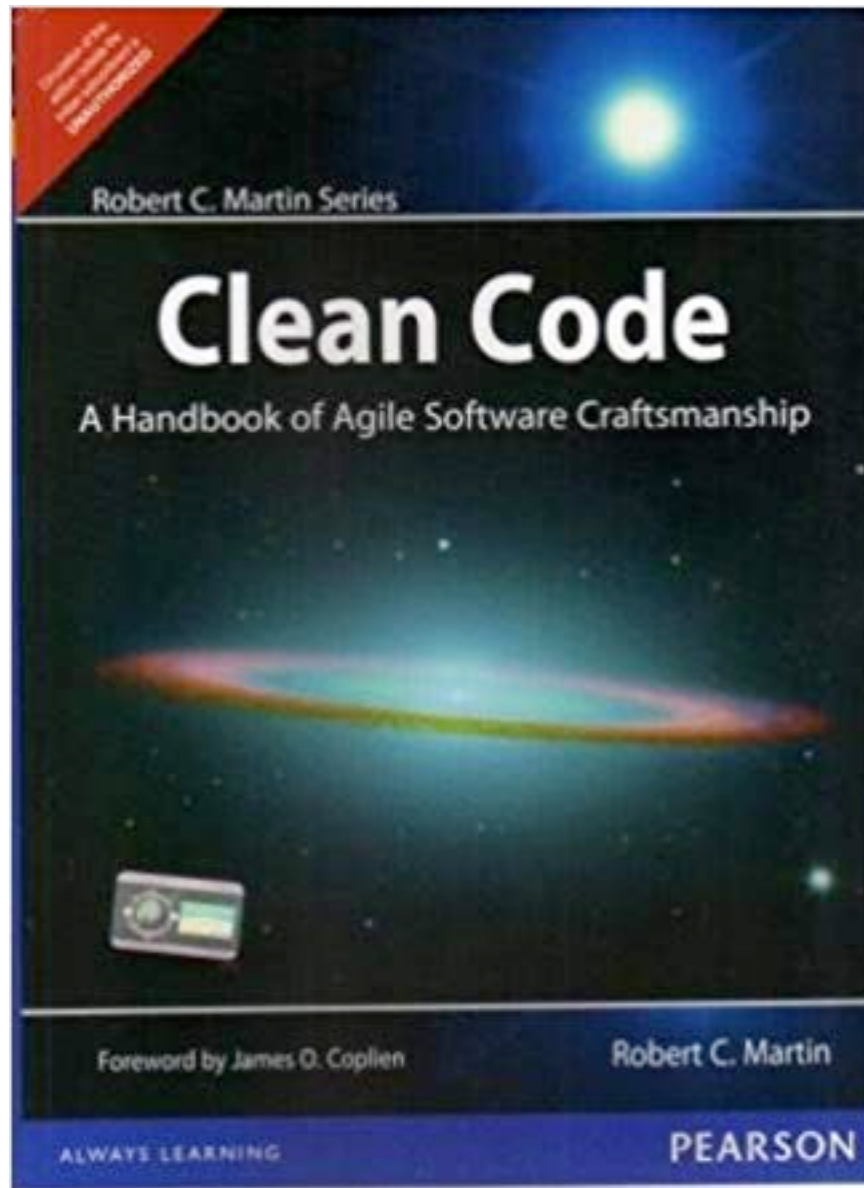
People will be using the words you choose in their conversation for the next 20 years. You want to be sure you do it right.

## Naming - Tips

- Use **intention-revealing** names.
- Name should speak for itself.
- A big descriptive name is better than short enigmatic name. A big descriptive name is better than descriptive comment.
- Name should be combination of noun/verb. Classes are noun. Functions are verb/noun pair. Boolean name should answer yes/no.
- Pick **one** word per concept.
- Add meaningful **context**.
- Use solution **domain** names.

# CLEAN CODE BOOK

.....



<https://amzn.to/3a2XRNd>

# CLEAN CODE

---

- It's a good book to start. However...
- Some of the code samples in the book are not nearly as clean as one would expect them to be.
- <https://github.com/unclebob/javaargs>

## Javaargs

```
public class ArgsMain {  
    public static void main(String[] commandLineArgs) {  
        try {  
            Args val = new Args("l,p#,d*", commandLineArgs);  
  
            boolean logging = val.getBoolean('l');  
            int port = val.getInt('p');  
            String directory = val.getString('d');  
  
        } catch (ArgsException e) {  
            System.out.println(e.errorMessage());  
        }  
    }  
}
```

## Schema:

- char - Boolean arg.
- char\* - String arg.
- char# - Integer arg.
- char## - double arg.
- char[\*] - one element of a string array.

Example schema: (f,s\*,n#,a##,p[\*])

Corresponding command line:

```
"-f -s Name -n 1 -a 3.2 -p e1 -p e2 -p e3
```



# ASSIGNMENT

---

- Reference: <https://github.com/unclebob/javaargs>
- Clean it up.
- Due 20-01-20.
- If you are copying/forking code, adhere to the license of the original code.
- Create your Github account and package name. Share it with the TAs.
- TAs to schedule a tutorial on Java fundamentals for those who need it. The tutorial is optional.

# REFERENCES

---

- Talks by Kevlin Henney on Youtube
- <https://medium.com/@mosquitowz/clean-code-quotes-5-formatting-b7bdd4a828d6>
- Clean Code by Robert C. Martin - <https://amzn.to/3a2XRNd>
- <https://junit.org/junit5/docs/current/user-guide/#writing-tests>