

A grayscale photograph of a person with dark hair wearing large headphones, looking intently at a laptop screen. Their hands are clasped together near their chin. The background is blurred, showing what appears to be a modern office or studio environment. A white rectangular border is superimposed over the image, framing the text.

# INTRODUCTION

Java clean code

# Outline

## **Lesson 1.**

Clean code

## **Lesson 2.**

Names

## **Lesson 3.**

Methods

## **Lesson 4.**

Classes

## **Lesson 5.**

Comments

*Any fool can write code that  
a computer can understand.  
Good programmers write code that  
humans can understand.*

**Martin Fowler**



# Grady Booch

*"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."*



# Rod Johnson

*"If you know what something does, you got a pretty good chance guessing the name of the Spring class or interface for it."*





# Ron Jeffries

*"In recent years I begin, and nearly end, with Beck's rules of simple code. In priority order, simple code:*

- Runs all the tests;*
- Contains no duplication;*
- Expresses all the design ideas that are in the system;*
- Minimizes the number of entities such as classes, methods, functions, and the like."*



# Bad code

- Tight coupling (Spagetti)
- Duplication of code
- Huge Classes and methods
- Not maintainable





[illegible]

- [illegible]



# Bad code

```
37
38 public static boolean isNull(int value) {
39     Integer integer = new Integer(value);
40     Dead code if (integer == null) {
41         return true;
42     } else {
43         return false;
44     }
45 }
```



# Bad code

Maybe one if statement could do the job.

```
boolean isUserAuthorized = user.isSuperAdmin();  
if(!isUserAuthorized){  
    isUserAuthorized = isAdminOfEntity1();  
}if(!isUserAuthorized){  
    isUserAuthorized = isAdminOfEntity2();  
}if(!isUserAuthorized){  
    throw new AccessDeniedException(  
        "Authenticated user is not admin ");  
}
```



A grayscale photograph of three people in a room with a perforated wall. Two men and one woman are gathered around a table. One man is sitting on a stool, leaning over the table. The other man is standing behind him, also looking at the table. The woman is standing to the right, looking at the table. On the table is a laptop and some papers. The scene appears to be a collaborative work or study environment.

# Bad code

Are you really running? No, are you really running?

```
switch(kafkaStreams.state()){  
    case RUNNING: {  
        if (kafkaStreams.state().isRunning()){  
            countDownLatch.countDown();  
        }  
    }  
    break;  
};
```



# Bad code

Difference matters

```
if(CATEGORY_NORMAL.equalsIgnoreCase(categorie)){  
    return assignGroupStartWithPrefix(assignmentGroups);  
}else if(CATEGORY_EXTERNAL.equalsIgnoreCase(categorie)){  
    return assignGroupStartWithPrefix(assignmentGroups);  
}
```





# Bad code

There is a slight chance this `googleApiClient` object would change by a mysterious power !!

```
private void EnableGPSAutomatically() {  
    GoogleApiClient googleApiClient = null;  
    if (googleApiClient == null) {  
        googleApiClient = new GoogleApiClient.Builder(this).  
            addApi(LocationServices.API).addConnectionCallbacks(this).  
            addOnConnectionFailedListener(this).build();  
        /* about 50 lines of code were here ... * all depends on this if  
        * statement ... ! Actually the whole method  
        * * depends on this if statement ! */  
    }  
}
```





# Bad code

Who needs names?

```
private HashMap<String, Tuple6<float[],  
String[], String[], String[], String>>  
memberNameChangedToProtectTheInnocent = null;
```





# Bad code

Throw an exception - the easy way

```
private <T> Supplier<T> abort(Class<T> exception){  
    return () -> {  
        try {  
            return exception.newInstance();  
        } catch (InstantiationException | IllegalAccessException e) {  
            throw new RuntimeException(e); } };  
    throw abort(MyException.class).get();  
}
```





# Good code Summary

- Easy to read and understand
- Easy to maintain
- Easy to add new functionality
- Loose coupling



# Course Progress

Lesson 1

Clean code

Lesson 2

Names

Lesson 3

Methods

Lesson 4

Classes

Lesson 5

Comments

A sepia-toned photograph of a person clapping their hands. In the foreground, a wooden desk holds an open notebook with a smartphone resting on it. A laptop is partially visible in the background. A dark grey rectangular box with a thin white border is overlaid on the left side of the image, containing the text 'THANK YOU!'.

THANK YOU!