

# Static

## Methods, Attributes, FastJson

YEGOR BUGAYENKO

Lecture #2 out of 8

90 minutes

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as website. Copyright belongs to their respected authors.



Methods

Attributes

FastJson as Example

Read and Watch

## Chapter #1: Methods

## What static methods are for?

```
1 class Circle {  
2     public float radius;  
3 }  
4 class GeometryUtils {  
5     static float calcSquare(Circle c) {  
6         return c.radius * c.radius * 3.14;  
7     }  
8 }
```

```
1 class Circle {  
2     public float radius;  
3     float square() {  
4         return radius * radius * 3.14;  
5     }  
6 }
```

Most notable Java examples: FileUtils, IOUtils, and StringUtils from Apache Commons; Files from JDK7; Iterators from Google Guava. Read [this](#).

## What's wrong with “Utils”?

- 1) They are unbreakable dependencies
- 2) They are eager, not lazy
- 3) They are not cohesive

## Tight Coupling

```
1 void paintIt(Circle c) {  
2     float s = GeometryUtils.calcSquare(c);  
3     float p = s * 5.55;  
4     // paint it using the "p"  
5 }
```

```
1 void paintIt(Circle c) {  
2     float s = c.square();  
3     float p = s * 5.55;  
4     // paint it using the "p"  
5 }
```

Which snippet is easier to test? Try to write a test for the first one, expecting `s` to be equal to `42.0`. Read [this](#).

## Imperative, not Declarative

```
1 void paintIt(Circle c) {  
2     float s = GeometryUtils.calcSquare(c);  
3     if (t) { return; }  
4     float p = s * 5.55;  
5     // paint it using the "p"  
6 }
```

```
1 void paintIt(Circle c) {  
2     float s = new SquareOf(c);  
3     if (t) { return; }  
4     float p = s * 5.55;  
5     // paint it using the "p"  
6 }
```

Which snippet is more eager to calculate the square of the circle? Which one does it when it's really necessary? Read [this](#).

## Low Cohesion

```
1 class GeometryUtils {  
2     static float calcSquare(Circle c);  
3     static float calcPerimeter(Circle c);  
4     static float calcSinus(Angle a);  
5     static float calcCosinus(float s);  
6     // and many more...  
7 }
```

```
1 class Circle {  
2     float square();  
3     float perimeter();  
4 }  
5 class Angle {  
6     float sinus();  
7 }  
8 class Float {  
9     float cosinus();  
10 }
```

Which class looks more cohesive to you, the utility class `GeometryUtils` or the `Circle`?



## Chapter #2:

# Attributes

## Public literals

```
1 class Constants {  
2     public static float PI = 3.1415926;  
3     public static String UTF_8 = "utf-8";  
4     public static String LOCALE = "fr";  
5     // and many more  
6 }  
7 println("S'il vous plaît",  
8     Constants.LOCALE);  
9 printf("It is %see speech!",  
10     Constants.LOCALE);
```

```
1 class Print { }  
2 class TextInFrench { }  
3  
4 new Print(  
5     new TextInFrench(  
6         "S'il vous plaît"  
7     )  
8 )
```

We must solve the problem of functionality duplication, not just data duplication. Read [this](#).

## Singletons

```
1 class Canvas {  
2     public static Canvas INSTANCE =  
3         new Canvas();  
4     private Canvas() {}  
5     public void addCircle(Circle c);  
6 }  
7  
8 Canvas.INSTANCE.addCircle(c1);  
9 Canvas.INSTANCE.addCircle(c2);
```

```
1 c = new Canvas();  
2 c.addCircle(c1);  
3 c.addCircle(c2);
```

Read this.

Chapter #3:

## FastJson as Example

Chapter #4:

## Read and Watch

Utility Classes Have Nothing to Do With Functional Programming

OOP Alternative to Utility Classes

Composable Decorators

Public Static Literals ... Are Not a Solution for Data Duplication