Static

Yegor Bugayenko

Lecture #2 out of 8
80 minutes

The slidedeck was presented by the author in this YouTube Video

Pre-Test

Methods

Attributes

FastJson as Example

Chapter #1:
Pre-Test

How many static "entities" do you see here?

```
1  class FigureUtils {
2    static final float PI = 3.1415926;
3    static final FigureUtils INSTANCE;
4    static {
5      INSTANCE = new FigureUtils();
6    }
7    private FigureUtils() { /* empty */ }
8    static float perimeter(Circle c) {
9      return 2 * c.radius * PI;
10   }
11 }
```

How many?

• three

• four

• five

• six

• maybe seven

Chapter #2:
Methods

What static methods are for?

```
1  class Circle {
2    public float radius;
3  }
4
5  class GeometryUtils {
6    static float calcArea(Circle c) {
7      return c.radius * c.radius * 3.14;
8    }
9  }
```

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

Most notable Java examples [Bugayenko, 2015a]: FileUtils, IOUtils, and StringUtils from Apache Commons; Files from JDK 7; Iterators from Google Guava.

What's wrong with "Utils"?

1) They are <u>unbreakable</u> dependencies

2) They are <u>eager</u>, not lazy

3) They are <u>not cohesive</u>

1) Tight Coupling

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

```
1 void paintIt(Circle c) {
2   float s = c.area();
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 [Bugayenko, 2014].

2) Imperative, not Declarative

```
1  void paintIt(Circle c) {
2    float s = GeometryUtils.calcArea(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 AreaOf(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 area of the circle? Which one does it when it's really necessary? [Bugayenko, 2015b]

3) Low Cohesion

```
1  class GeometryUtils {
2    static float calcArea(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 area();
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`?

Static   @yegor256

Chapter #3:
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
8  println("S'il vous plaît",
9    Constants.LOCALE);
10 printf("It is %see speech!",
11   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 [Bugayenko, 2015c].

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);
```

Forget about singletons; never use them. Turn them into dependencies and pass them from object to object through the operator `new` [Bugayenko, 2016].

Chapter #4:
FastJson as Example

[JSONPath]

JSONPath

```java
public int size(Object rootObject) {
    if (rootObject == null) {
        return -1;
    }

    init();

    Object currentObject = rootObject;
    for (int i = 0; i < segments.length; ++i) {
        currentObject = segments[i].eval(this, rootObject, currentObject);
    }

    return evalSize(currentObject);
}
```

```java
public static int size(Object rootObject, String path) {
    JSONPath jsonpath = compile(path);
    Object result = jsonpath.eval(rootObject);
    return jsonpath.evalSize(result);
}
```

Some mind-blowing statistics:

- 3 constructors

- 1 interface implemented

- 59 methods at JSONPath

- 4,363 lines of code

- 34 inner static classes

- 74 times static keyword

https://github.com/alibaba/fastjson/blob/master/src/main/java/com/alibaba/fastjson/JSONPath.java

References

Yegor Bugayenko. OOP Alternative to Utility Classes. https://www.yegor256.com/140505.html, may 2014. [Online; accessed 08-07-2024].

Yegor Bugayenko. Utility Classes Have Nothing to Do With Functional Programming. https://www.yegor256.com/150220.html, feb 2015a. [Online; accessed 08-07-2024].

Yegor Bugayenko. Composable Decorators vs. Imperative Utility Methods. https://www.yegor256.com/150226.html, feb 2015b. [Online; accessed 08-07-2024].

Yegor Bugayenko. Public Static Literals ... Are Not a Solution for Data Duplication. https://www.yegor256.com/150706.html, jul 2015c. [Online; accessed 08-07-2024].

Yegor Bugayenko. Singletons Must Die. https://www.yegor256.com/160627.html, jun 2016. [Online; accessed 08-07-2024].