Static

Methods, Attributes, FastJson

Yegor Bugayenko

Lecture #2 out of 8
90 minutes

- Methods
- Attributes
- FastJson as Example
- Read and Watch

Chapter #1:

Methods

What static methods are for?

```
class Circle {
  public float radius;
}
class GeometryUtils {
  static float calcSquare(Circle c) {
    return c.radius * c.radius * 3.14;
  }
}
```

```
class Circle {
  public float radius;
  float square() {
    return radius * radius * 3.14;
  }
}
```

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

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

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

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

```
class GeometryUtils {
  static float calcSquare(Circle c);
  static float calcPerimeter(Circle c);
  static float calcSinus(Angle a);
  static float calcCosinus(float s);
  // and many more...
}
```

```
class Circle {
  float square();
  float perimeter();
}
class Angle {
  float sinus();
}
class Float {
  float cosinus();
}
```

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

Chapter #2:

Attributes

Public literals

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

```
class Print { }
class TextInFrench { }

new Print(
  new TextInFrench(
    "S'il vous plaît"
  )
)
```

We must solve the problem of functionality duplication, not just data duplication. Read this.

Singletons

```
class Canvas {
  public static Canvas INSTANCE =
    new Canvas();
  private Canvas() {}
  public void addCircle(Circle c);
}

Canvas.INSTANCE.addCircle(c1);
Canvas.INSTANCE.addCircle(c2);
```

```
c = new Canvas();
c.addCircle(c1);
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