Static

Methods, Attributes

Yegor Bugayenko

Lecture #2 out of 10
90 minutes

Theory

Chapter #1:

Theory

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.

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`.

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?

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`?