



Detecting Code Smells

4 Tiny Tasks

Simon Hodel, 03.09.2020

Recap: Code Smells Overview

→ Code Smells

Bloaters

Object-Orientation Abusers

Change Preventers

Dispensables

Couplers

Task 1: What does smell here?

```
private static Cyborgs[] = (private static) null;
private static void main(String[] args) {
    // Cyborgs[] = new Cyborgs[10];
    // ...
}
private static void main(String[] args) {
    // ...
}
private static void main(String[] args) {
    // ...
}
```

- BSOAP_RAV.java:

```
private static void main(String[] args) {
    // ...
}
private static void main(String[] args) {
    // ...
}
```

Task 1: What does smell here?

```
public class BSOAP_RAV {
    private static final String URL = "http://...";
    private static final String USERNAME = "...";
    private static final String PASSWORD = "...";

    public static void main(String[] args) {
        // ...
        BSOAP_RAV obj = new BSOAP_RAV();
        obj.doSomething();
        // ...
    }

    private void doSomething() {
        // ...
        BSOAP_RAV obj = new BSOAP_RAV();
        obj.doSomething();
        // ...
    }
}
```

- BSOAP_RAV.java:

```
public class BSOAP_RAV {
    // ...
    private void doSomething() {
        // ...
        BSOAP_RAV obj = new BSOAP_RAV();
        obj.doSomething();
        // ...
    }
}
```

Dispensables:
Dead code

Task 2: What does smell here?

```
class User { type: User, bio: string }
if (user.bio === ' ') {
  class User { type: User, bio: string }
} else {
  class User { type: User, bio: string }
}

class User { type: User, bio: string }
class User { type: User, bio: string }
class User { type: User, bio: string }
class User { type: User, bio: string }
```

Task 2: What does smell here?

```
class Foo {
    public void foo() {
        // ...
    }
}

class Bar {
    public void bar() {
        // ...
    }
}

class Baz {
    public void baz() {
        // ...
    }
}

class Qux {
    public void qux() {
        // ...
    }
}

class Goo {
    public void goo() {
        // ...
    }
}

class Hoo {
    public void hoo() {
        // ...
    }
}

class Koo {
    public void koo() {
        // ...
    }
}

class Loo {
    public void loo() {
        // ...
    }
}

class Moo {
    public void moo() {
        // ...
    }
}

class Noo {
    public void noo() {
        // ...
    }
}

class Ooo {
    public void ooo() {
        // ...
    }
}

class Poo {
    public void poo() {
        // ...
    }
}

class Qoo {
    public void qoo() {
        // ...
    }
}

class Roo {
    public void roo() {
        // ...
    }
}

class Soo {
    public void soo() {
        // ...
    }
}

class Too {
    public void too() {
        // ...
    }
}

class Uoo {
    public void uoo() {
        // ...
    }
}

class Voo {
    public void voo() {
        // ...
    }
}

class Woo {
    public void woo() {
        // ...
    }
}

class Xoo {
    public void xoo() {
        // ...
    }
}

class Yoo {
    public void yoo() {
        // ...
    }
}

class Zoo {
    public void zoo() {
        // ...
    }
}
```

Dispensables:
Comments

Task 3: What does smell here?

```
public class Main {
    public static void main(String[] args) {
        // ...
    }
}

public class SomeClass {
    // ...
}

public class AnotherClass {
    // ...
}

// ...
```

Task 3: What does smell here?

```
public class Test {  
    public void method() {  
        public void method(String s, int i, double d, boolean b, long l, float f, char c, byte b2, short s2, Object o) {  
            // ...  
        }  
    }  
}
```

Bloaters:
Long Parameter
List

Task 4: What does smell here (without reading the actual lines)?



The image shows a code editor with a large, dense block of code. The code is heavily indented and contains many lines of comments and code blocks, suggesting a complex or messy code structure. The code is written in a dark theme with various colors for syntax highlighting. The overall appearance is that of a large, unstructured code block, which is a common sign of code smell.

Task 4: What does smell here (without reading the actual lines)?

The image shows a screenshot of a code editor with a dark background and colorful syntax highlighting. A vertical red bar highlights the left side of the editor, specifically the line numbers. The code is a single, extremely long method that spans almost the entire height of the visible editor window. The code is dense and appears to be a complex implementation of a class or interface, with many lines of code, including comments and various statements. The red bar is positioned on the left side of the editor, indicating the line numbers for each line of code.

Bloaters:
Long Method



Credits

- Code Smell Catalogue:
<https://refactoring.guru/refactoring/smells>