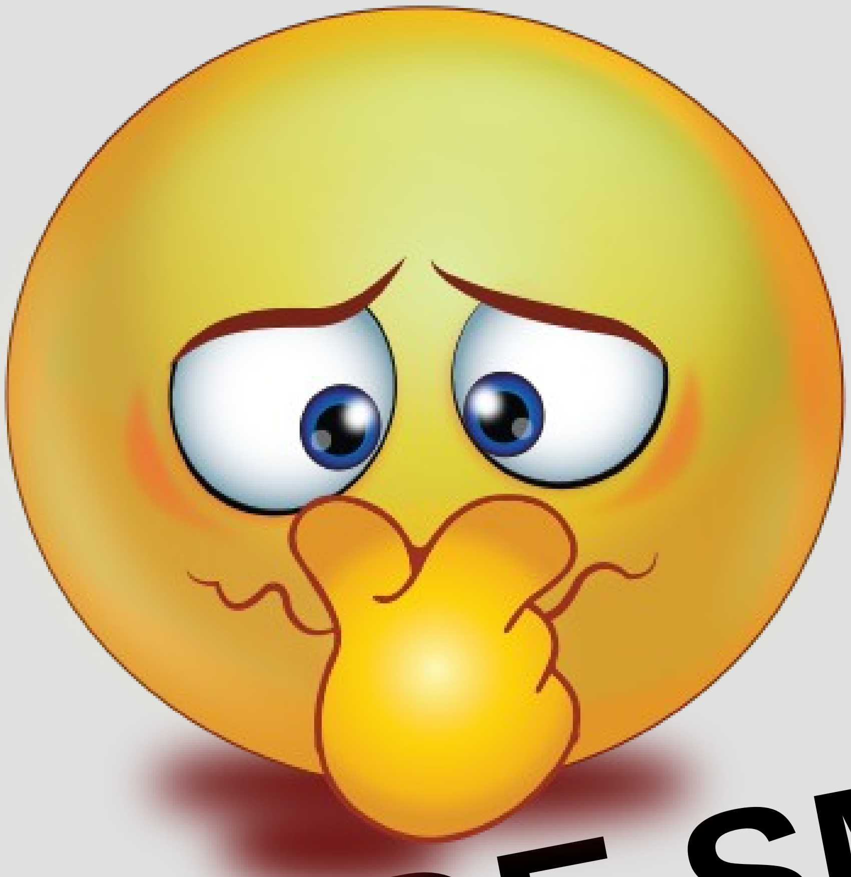


CODE SMELL

MOST IMPORTANT SMELLS TO AVOID

Marcel Mathis

What is code smell?

“Code smells are signs that your code is not as **clean** and **maintainable** as it could be. They can derive from the misuse of syntax and almost always suggest your **code needs to be refactored** or redesigned to improve the overall quality of the program(s).”

–

JB Larson

Feature envy

Coupler

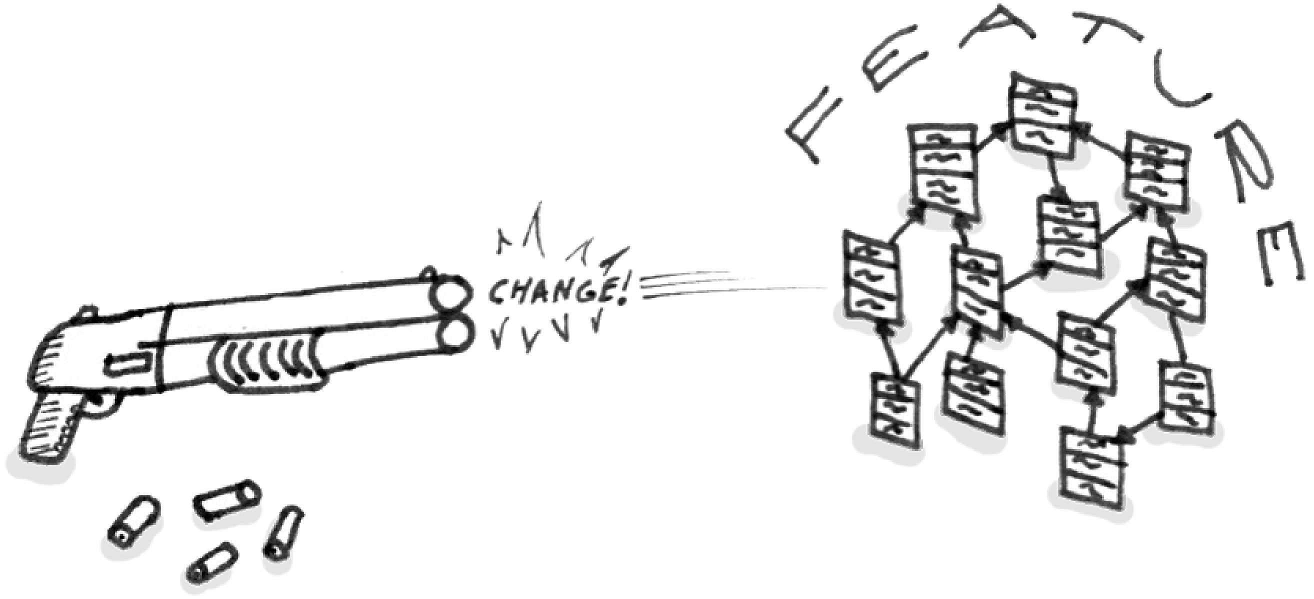
- A method accesses the data of another object more than its own data.

```
public void isFeatureEnvy() {
    featureEnvy.setNumberOfCallsFromOtherMethod();
    featureEnvy.calc();
    return featureEnvy.getValue()
}
```

- Move it to the featureEnvy class!

Shotgun Surgery

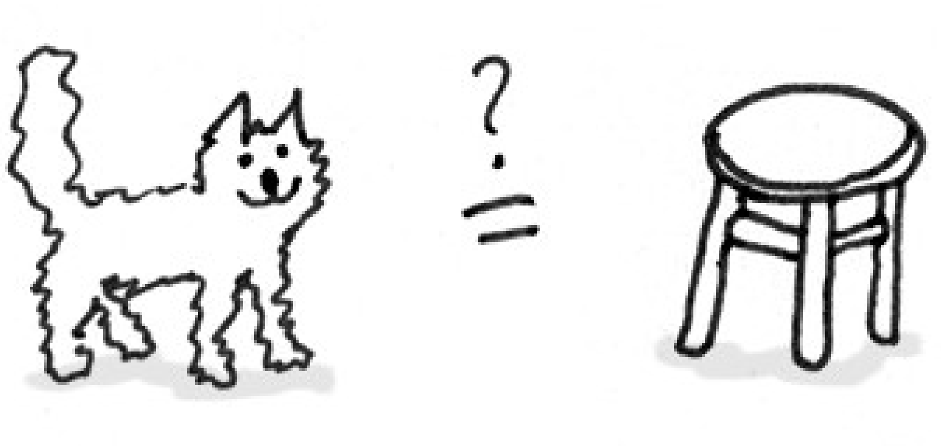
Change preventer



- Move methods and fields to existing or new classes.

Refused Bequest

Object orientation abuser



- If inheritance makes no sense, eliminate inheritance.
- If inheritance is appropriate, get rid of unneeded fields and methods in the subclass.

Switch Statements

Object orientation abuser

- Can lead to same switch statement scattered in different places.

```
void doOption(Options options) {
    switch (options) {
        case OPTION_1:
            doOption1();
            break;
        case OPTION_2:
            doOption2();
            break;
    }
}
```

```
void doOption(IOption option) {
    option.doOption();
}
```

Primitive obsession

Bloater

- Use small object instead of primitives

```
class Person {
    // Give the first name after the name in the name parameter
    // Mind the \n to separate street and town in address
    // Birthday is the number in seconds after 1970
    Person(String name, String address, int birthday) {
    }
}
```

```
class Person {
    Person(Name name, Address address, Date birthday)
{
    }
}

class Name {
    Name(String lastName, String firstName) {
    }
}

class Address {
    Name(String street, int postalCode, String town) {
    }
}
```

ℹ Comments are also code smells :-)

Personal conclusion

Avoiding code smell needs some effort and it must be regularly trained. But then ...

- … code can get so much easier to read
- … code maintenance gets much simpler
- … complex problems may disappear
- … code is less error prone

Personal conclusion

… but we do not want perfect code.

- Refactor when you spot obvious smells, but don’t search them explicitly
- There is still some room for personal preferences.
  - e.g. Lazy class vs large class
- Use the help of code analysis tools
  - It’s an indication but also has false positives

Any questions

Thank you

Marcel Mathis

marcel.mathis@css.ch

linkedin.com/in/marcel-mathis-83ab64252

References

- https://sourcemaking.com/refactoring/smells/
- Alcor Academy
- https://www.i2symbol.com/extension/stickers/smileys/bad-smell-face-1bc851fb837a9caa815462a80c714a8e
- https://clipartmag.com/any-queries-images-for-presentation