Object Calisthenics
by Jeff Bay

9 steps to better software design today

1. One level of indentation per method
2. Don't use the ELSE keyword
3. Wrap all primitives and Strings
4. First class collections
5. One dot per line
6. Don't abbreviate
7. Keep all entities small
8. No classes with more than two instance variables
9. No getters/setters/properties

1. One level of indentation per method

```
String board() {
    StringBuffer buf = new StringBuffer();
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++)
            buf.append(data[i][j]);
        buf.append("\n");
    }
    return buf.toString();
}
```

```
String board() {
    StringBuffer buf = new StringBuffer();
    collectRows(buf);
    return buf.toString();
}

void collectRows(StringBuffer buf) {
    for (int i = 0; i < 10; i++)
        collectRow(buf, i);
}

void collectRow(StringBuffer buf, int row) {
    for (int i = 0; i < 10; i++)
        buf.append(data[row][i]);
    buf.append("\n");
}
```

2. Don’t use the ELSE keyword

- **Early Return**: for parameter validation return immediately
- **Map**
- **Polymorphism**: is an alternative to if/else/switch statements. For instance, it is possible to use Strategy Pattern or inheritence to replace every clause in the control statement.

  There are variations to how the "right" strategy is obtained.

  - caller provides the strategy (as a method argument)
  - strategy is memorised as member variable
  - strategy is obtained from a map (or any other lookup mechanism)

- **Null Object or Optional or Empty list** : Dramatically reduces the need for null checking.
- …

3. Wrap all primitives and Strings

```
public class Discounter {

    private int discount;

    public int applyTo(int initialPrice) {
        return initialPrice - discount;
    }
```

```
public class Discounter {

    private Money discount;

    public Money applyTo(Money initialPrice) {
        return initialPrice.minus(discount);
    }

}
```

```
public class Money {

    private int discountInCents;
```

4. First class collections

- Same as 3.Wrap all primitives but for collections
- Behaviors related to the collection have a home

```
public class Accounts {

    private Map<AccountId, Account> accounts = new HashMap<AccountId, Account>();
```

5. One dot per line

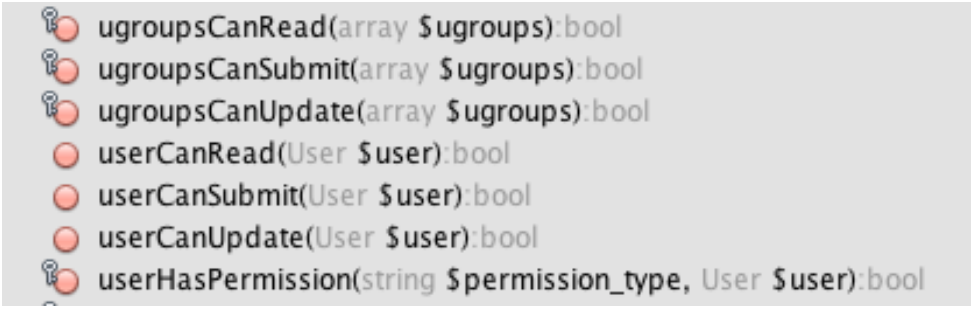
- Law of Demeter (“Only talk to your friends”)

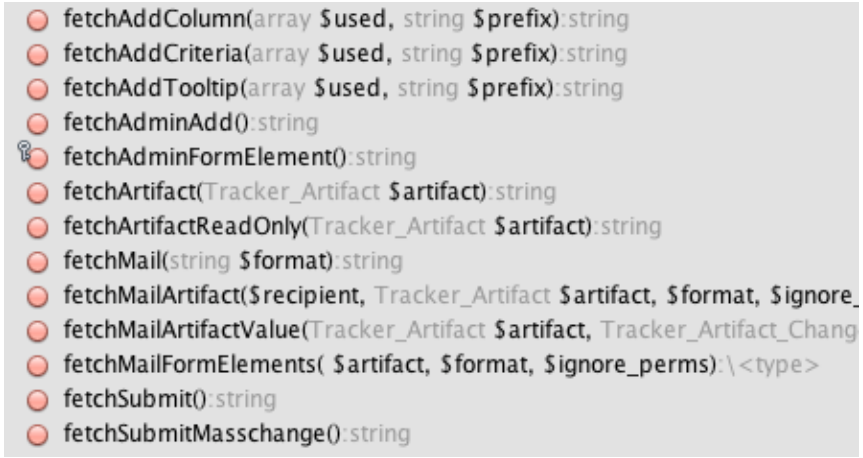
```
$this->gitBackupDir = PluginManager::instance()
                      ->getPluginByName('git')
                      ->getPluginInfo()
                      ->getPropVal('git_backup_dir');
```

6. Don’t abbreviate

- Keep class and method names to 1-2 words, and avoid names that duplicate the context

Think about why you want to abbreviate. Is it because you’re typing the same word over and over again? If that’s the case, perhaps your method is used too heavily and you are missing opportunities to remove duplication. Is it because your method names are getting long? This might be a sign of a misplaced responsibility, or a missing class.





7. Keep all entities small

- No class over 50 lines
- No package over 10 files

- SRP (Single Responsibility Principle)

8. No classes with more than two instance variables

- Class Cohesion
- Object Model decomposition

9. No getters/setters/properties

```
q.setQuality(q.getQuality() - 1);
```

```
q.decrease();
```

```
q.quality = 0;
```

```
q.dropToZero();
```

« Tell don’t ask »

Links

- Thoughtworks Anthology by PragProg
  http://pragprog.com/book/twa/thoughtworks-anthology
- Object Calisthenics
  http://www.bennadel.com/resources/uploads/2012/ObjectCalisthenics.pdf
- How object oriented are you feeling today?
  Krzysztof Jelski - Software Craftsmanship 2011 Conference
  http://fr.slideshare.net/KrzysztofJelski/how-object-oriented-are-you-feeling-today