



Smelly Code Refactoring

A Real Life Example Trying To Make The Code Less Smelly





What is a Code Smell?

**I'm not completely bad,
but I'm not completely
good either!**

Dance Class Booking App

localhost:3000/completed

Let's Pole Dance

username

Schedule

Upcoming **Completed** Book Classes

Monday, 16, May 17:45	Pole Beginner Yin Hui	Book Again
Monday, 30, May 17:45	Pole Beginner Yin Hui	Book Again
Tuesday, 31, May 19:00	Flow Pole Yin Hui	Book Again

localhost:3000/completed

Smelly Code

```
34 // GET: api/Schedule/1
35 [HttpGet("{memberId}")]
0 references
36 public async Task<ActionResult<IEnumerable<Schedule>>> GetMemberSchedules(int memberId, string? status)
37 {
38     var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
39     .Include(s => s.WeekPlan)
40     .ThenInclude(w => w.Class)
41     .Include(s => s.WeekPlan)
42     .ThenInclude(w => w.Instructor)
43     .OrderBy(s => s.ScheduleDate)
44     .ToListAsync();
45
46     if (schedules.Count == 0)
47     {
48         return NotFound("No member found by id");
49     }
50
51     // _logger.LogInformation("#####");
52     // _logger.LogInformation(JsonSerializer.Serialize(schedules));
53     _logger.LogInformation($"status: {status}");
54
55     var currentDate = DateTime.Now;
56
57     if (status is not null)
58     {
59         var upcomingSchedules = new List<Schedule>();
60         var completedSchedules = new List<Schedule>();
61         foreach (var s in schedules)
62         {
63             string time = s.WeekPlan.WeekTime.Split(',', ' ').Last();
64             string scheduleDateTime = s.ScheduleDate.ToShortDateString() + ' ' + time;
65             // _logger.LogInformation(scheduleDateTime);
66             if (currentDate <= DateTime.Parse(scheduleDateTime))
67             {
68                 // _logger.LogInformation(JsonSerializer.Serialize(s));
69                 upcomingSchedules.Add(s);
70             } else
71             {
72                 completedSchedules.Add(s);
73             }
74         }
75
76         if (status == "upcoming")
77         {
78             return upcomingSchedules;
79         } else if (status == "completed")
80         {
81             return completedSchedules;
82         } else {
83             return NotFound("Status must be `upcoming` or `completed`!");
84         }
85     }
86
87     return schedules;
88 }
```

```

34 // GET: api/Schedule/1
35 [HttpGet("{memberId}")]
36 public async Task<ActionResult<IEnumerable<Schedule>>> GetMemberSchedules(int memberId, string? status)
37 {
38     var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
39     .Include(s => s.WeekPlan)
40     .ThenInclude(w => w.Class)
41     .Include(s => s.WeekPlan)
42     .ThenInclude(w => w.Instructor)
43     .OrderBy(s => s.ScheduleDate)
44     .ToListAsync();
45
46     if (schedules.Count == 0)
47     {
48         return NotFound("No member found by id");
49     }
50
51     // _logger.LogInformation("#####");
52     // _logger.LogInformation(JsonSerializer.Serialize(schedules));
53     _logger.LogInformation($"status: {status}");
54
55     var currentDate = DateTime.Now;
56
57     if (status is not null)
58     {
59         var upcomingSchedules = new List<Schedule>();
60         var completedSchedules = new List<Schedule>();
61         foreach (var s in schedules)
62         {
63             string time = s.WeekPlan.WeekTime.Split(',', ' ').Last();
64             string scheduleDateTime = s.ScheduleDate.ToShortDateString() + ' ' + time;
65             // _logger.LogInformation(scheduleDateTime);
66             if (currentDate <= DateTime.Parse(scheduleDateTime))
67             {
68                 // _logger.LogInformation(JsonSerializer.Serialize(s));
69                 upcomingSchedules.Add(s);
70             } else
71             {
72                 completedSchedules.Add(s);
73             }
74         }
75
76         if (status == "upcoming")
77         {
78             return upcomingSchedules;
79         } else if (status == "completed")
80         {
81             return completedSchedules;
82         } else {
83             return NotFound("Status must be 'upcoming' or 'completed!'");
84         }
85     }
86
87     return schedules;
88 }

```



Long method



Comments

```
var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
.Include(s => s.WeekPlan)
.ThenInclude(w => w.Class)
.Include(s => s.WeekPlan)
.ThenInclude(w => w.Instructor)
.OrderBy(s => s.ScheduleDate)
.ToListAsync();
```



Message Chains



Middle Man

```
if (schedules.Count == 0)
{
    return NotFound("No member found by id");
}
```

💩 Comments / Wrong Error Message
Violates “Least Astonishment” (WTF)


```
if (status is not null)
{
    var upcomingSchedules = new List<Schedule>();
    var completedSchedules = new List<Schedule>();
    foreach (var s in schedules)
    {
        string time = s.WeekPlan.WeekTime.Split(',', ' ').Last();
        string scheduleDateTime = s.ScheduleDate.ToShortDateString() + ' ' + time;
        // _logger.LogInformation(scheduleDateTime);
        if (currentDate <= DateTime.Parse(scheduleDateTime))
        {
            // _logger.LogInformation(JsonSerializer.Serialize(s));
            upcomingSchedules.Add(s);
        } else
        {
            completedSchedules.Add(s);
        }
    }
}
```

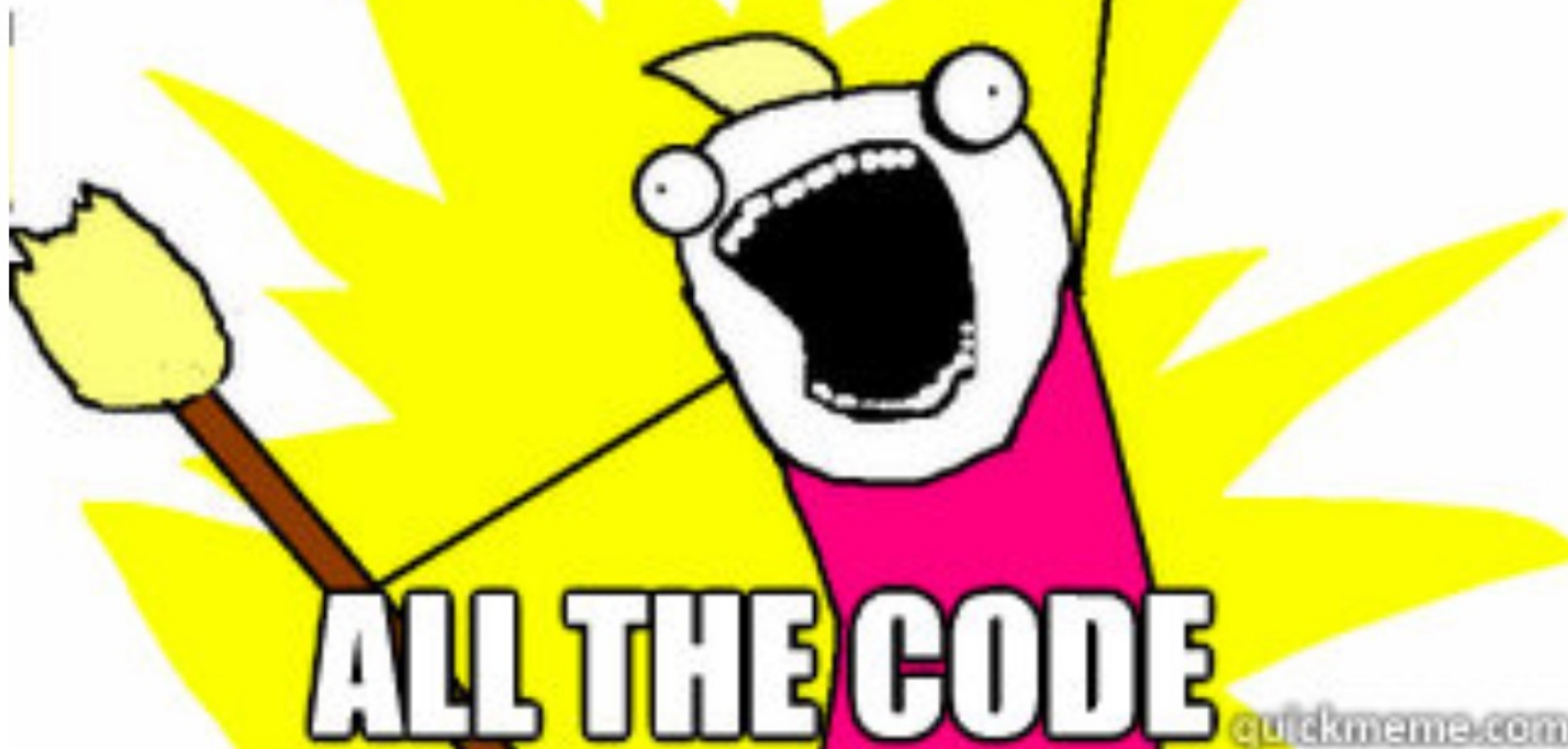
- 💩 Message Chains
- 💩 Data Class
- 💩 Nested Conditional

```
if (status == "upcoming")
{
    return upcomingSchedules;
} else if (status == "completed")
{
    return completedSchedules;    Yuqing, 9 months ago • Added up
} else {
    return NotFound("Status must be `upcoming` or `completed`!");
}
```

💩 Primitive Obsession

💩 Else block (object calisthenics)

REFACTOR



ALL THE CODE

quickmeme.com

```

1 [HttpGet("{memberId}")]
2 public async Task<ActionResult<IEnumerable<Schedule>>> GetMemberSchedules(int memberId, string status)
3 {
4     var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
5     .OrderBy(s => s.ScheduleDate)
6     .ToListAsync();
7
8     if (schedules.Count == 0)
9     {
10         return NotFound("No schedule found by memberId");
11     }
12
13     var currentDate = DateTime.Now;
14
15     List<Schedule> upcomingSchedules, completedSchedules;
16     SeparateSchedules(schedules, currentDate, out upcomingSchedules, out completedSchedules);
17
18     if (status == STATUS.UPCOMING)
19     {
20         return upcomingSchedules;
21     }
22
23     if (status == STATUS.COMPLETED)
24     {
25         return completedSchedules;
26     }
27
28     return NotFound("Invalid status");
29 }
30
31 private static void SeparateSchedules(List<Schedule> schedules, DateTime currentDate, out List<Schedule> upcomingSchedules, out List<Schedule> completedSchedules)
32 {
33     upcomingSchedules = new List<Schedule>();
34     completedSchedules = new List<Schedule>();
35     foreach (var s in schedules)
36     {
37         string scheduleDateTime = s.DateTime();
38         if (currentDate <= DateTime.Parse(scheduleDateTime))
39         {
40             upcomingSchedules.Add(s);
41             continue;
42         }
43         completedSchedules.Add(s);
44     }
45 }
46

```

 Make status non-nullable

 Rewrite Schedule class, remove middle man Weekplan

 Correct the error message

 Delete comments

 Extract method

 Wrap primitives

 Get rid of nested conditionals

 Create method to “ask, don’t tell”

 Remove Else block

Thank you very much!

Grazie Mille!

Author: Yuqing Lu

Contact: luyuqing0708@gmail.com