Ruby Monstas

Session 12: Interlude

Software Design Patterns

In software engineering, a **software design pattern** is a **general, reusable solution** to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. It is a **description or template** for how to solve a problem that can be used in many different situations. Design patterns are formalized **best practices** that the programmer can use to solve common problems when designing an application or system.

Wikipedia: Software design pattern

Anti-patterns

An **anti-pattern** is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.

Wikipedia: Anti-pattern

A particular anti-pattern

```ruby
class Student
  def initialize(id, full_name)
    @id = id
    @full_name = full_name
  end

  attr_reader :id, :full_name

  def description
    puts "Student #{id} is called #{full_name}"
  end
end


students_csv_content = File.read("students.csv")
lines = students_csv_content.split("\n")

students = []

lines[1..-1].each do |line|
  row = line.split(",")

  students << Student.new(row[0], row[1])
end

students.each do |student|
  student.description
end
```

A particular anti-pattern

```ruby
class Student
  def initialize(id, full_name)
    @id = id
    @full_name = full_name
  end

  attr_reader :id, :full_name

  def description
    puts "Student #{id} is called #{full_name}"
  end
end


students_csv_content = File.read("students.csv")
lines = students_csv_content.split("\n")

students = []

lines[1..-1].each do |line|
  row = line.split(",")

  students << Student.new(row[0], row[1])
end

students.each do |student|
  student.description
end
```

Why is this bad?

Why should the Student class know about writing to the console?
What if we wanted to save the descriptions to a file?
Or send them over the network?
How would you write a test for this method?

The solution

```ruby
class Student
  def initialize(id, full_name)
    @id = id
    @full_name = full_name
  end

  attr_reader :id, :full_name

  def description
    "Student #{id} is called #{full_name}"
  end
end


students_csv_content = File.read("students.csv")
lines = students_csv_content.split("\n")

students = []

lines[1..-1].each do |line|
  row = line.split(",")

  students << Student.new(row[0], row[1])
end

students.each do |student|
  puts student.description
end
```

The benefit

This simple change makes this possible:

```
30  File.open('student_descriptions.txt', 'w') do |file|
31    students.each do |student|
32      file.puts(student.description)
33    end
34  end
```

```
36  require 'minitest/autorun'
37
38  class StudentTest < Minitest::Test
39    def test_description
40      student = Student.new(1, 'Ferdinand Niedermann')
41      assert_equal 'Student 1 is called Ferdinand Niedermann', student.description
42    end
43  end
```

The pattern: Separation of concerns

The student class only knows about its own data, how to read and write it.

Everything else is handled outside of the class.

If the logic for returning a description gets very complex, it might warrant extracting it to another class!

When exactly logic within a class warrants extracting it to another is not a clear-cut line. Use your intuition!