Pain of OOP

This series of lectures by Yegor Bugayenko was first presented to students at Innopolis University in 2023 and video recorded. The complete set of slide decks is hosted and maintained in the yegor256/painofoop GitHub repository.

Abstract:
The course is a critical review of the current situation in object-oriented programming, especially in the Java, C++, Ruby, and JavaScript worlds. In this course, certain programming idioms, which are sometimes called "best practices," are criticized for their negative impact on code quality. These include static methods, NULL references, getters and setters, ORM and DTO, annotations, traits and mixins, inheritance, and many others. Instead, much "cleaner" object-oriented programming practices will be proposed.

What is the goal?
The primary objective of the course is to help students understand the difference between "object thinking," originally motivated the appearance of OOP, and the modern practices that often severly impact the quality of code in a negative way.

Who is the teacher?
Yegor is developing software for more than 30 years, being a hands-on programmer (see his GitHub account: @yegor256) and a manager of other programmers. At the moment Yegor is a director of an R&D laboratory in Huawei (Moscow, Russia). His primary research focus is software quality problems. Some of the lectures he has recently presented at some software conferences could be found at his YouTube channel. Yegor also published a few books and wrote a blog about software engineering and object-oriented programming. Yegor previously tought a few courses in Innopolis University (Kazan, Russia) and HSE University (Moscow, Russia), for example, Practical Program Analysis (2023) and Software Quality Metrics (2024) (all videos are available).

Why this course?
Maintainability of object-oriented software that most of us programmers write these days is way below our expectations. One of the main reasons for that is our misunderstanding of what *objects* are. This course may help clear things up.

What's the methodology?
Each lecture is a critical review of one of the existing "best practices," such as static methods or getters, with an intent to highlight its negative impact on the quality of software.

Course Structure

Prerequisites to the course (it is expected that a student knows this):

- How to write code
- How to design software
- How to use Git and GitHub

After the course a student hopefully will understand:

- What is the difference between objects and data?
- Why are static methods bad?
- What is immutability and why is it good?
- How should one design a constructor?
- How can exceptions be handled correctly?
- What is data hiding for?
- What's wrong with Printers, Writers, Scanners, and Readers?
- Why are NULL references considered a billion-dollar mistake?
- What's wrong with mixins and traits?
- How can getters and setters be avoided?
- How is declarative programming better than imperative programming?
- Why is composition better than inheritance?
- Where should data be stored if DTOs are considered bad practice?
- How can the ORM design pattern be avoided?
- Why are long variable names considered bad design?
- Why are type casting and type checking against OOP principles?
- What is cohesion and why does it matter?
- How can SOLID and SRP principles be applied?
- What is Inversion of Control for, and why are DI Containers considered harmful?
- Why is MVC considered a bad design idea?

Lectures

The following topics are discussed:

1. Algorithms
2. Static Methods
3. Getters
4. Setters and Mutability
5. “-ER” suffix
6. NULL references
7. Type casting and reflection
8. Inheritance

Grading

In order to pass the course, students must attend lectures, labs, and contribute to one of the following GitHub repositories, which are written in more or less "pure" object-oriented style:

- `yegor256/cactoos` (Java + Maven)
- `yegor256/takes` (Java + XSLT)
- `objectionary/eo2js` (JavaScript)
- `zerocracy/baza` (Ruby + Sinatra + PostgreSQL)
- `yegor256/cam` (Bash + Python + Make)

There is no exam at the end of the course. Instead, each student earns points for the following results:

Result	Points	Limit
Attended a lecture	+1	6
Attended a lab	+1	8
Submitted a ticket (that was accepted)	+2	8
Submitted a pull request (that was merged)	+4	32

Then, 25+ points mean "A," 17+ mean "B," and 9+ mean "C."

An online lecture is counted as "attended" only if a student was personally presented in Zoom for more than 75% of the lecture's time. Watching the lecture from the computer of a friend doesn't count.

Learning Materials

[1] Yegor Bugayenko. *Elegant Objects*. Amazon, 2016.

[2] Yegor Bugayenko. *Elegant Objects*. Amazon, 2017.

[3] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, 2008. DOI: 10.5555/1388398.

[4] Steve McConnell. *Code Complete, Second Edition*. Microsoft Press, 2004. DOI: 10.5555/1096143.

[5] David West. *Object Thinking*. Pearson Education, 2004. DOI: 10.5555/984130.

You may also want to read the blog posts of Yegor Bugayenko, on his blog, watch his video lectures on YouTube, and watch Object Thinking meetup presentations, on YouTube.