Pain of OOP

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

What is the course about?
This course offers a critical review of the current state of object-oriented programming, with a particular focus on the Java, C++, Ruby, and JavaScript ecosystems. It examines various programming idioms—often referred to as "best practices"—and critiques their negative impact on code quality. These include static methods, null references, getters and setters, ORM and DTO patterns, annotations, traits and mixins, inheritance, and others. In contrast, the course proposes alternative approaches that promote cleaner object-oriented programming practices.

What is the goal?
The primary objective of the course is to help students understand the difference between "object thinking"—which originally motivated the emergence of object-oriented programming—and modern practices that often severely degrade code quality.

Who is the teacher?

Yegor has been developing software for over 30 years, working both as a hands-on programmer (see his GitHub account: @yegor256) and as a manager of other programmers. He is currently a software developer at Huawei (Moscow, Russia). His primary research focus is on software quality issues. Some of the lectures he has recently presented at software conferences can be found on his YouTube channel. Yegor has also published several books and maintains a blog about software engineering and object-oriented programming. He has previously taught courses at Innopolis University (Kazan, Russia) and HSE University (Moscow, Russia), including Practical Program Analysis (2023) and Software Quality Metrics (2024) (all videos are available).

Why this course?
The maintainability of object-oriented software that most of us programmers write today falls far short of our expectations. One of the main reasons for this is a fundamental misunderstanding of what objects truly are. This course may help clear things up.

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 completing the course, a student will hopefully 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 and labs. They must also contribute to one of the following GitHub repositories, which are written in more or less "pure" object-oriented style:

- `objectionary/eo` (Java + XSLT)
- `yegor256/takes` (Java)
- `cqfn/aibolit` (Python)
- `objectionary/eoc` (JavaScript)

There is no oral or written 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	3
Submitted a ticket (that was accepted)	+2	8
Submitted a pull request (that was merged)	+4	32

Final grades are based on accumulated points: 25+ earns an "A," 17+ a "B," and 9+ a "C."

If eight or fewer points are accumulated during the course, the student may still earn them before the retake.

An online lecture is counted as "attended" only if a student was personally present 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] Kent Beck. Smalltalk Best Practice Patterns. Prentice Hall, 1997.

[2] Grady Booch et al. Object-Oriented Analysis and Design With Applications. Addison-Wesley, 1994. DOI: 10.5555/1407387.

[3] Yegor Bugayenko. Elegant Objects. Amazon, 2016.

[4] Yegor Bugayenko. Elegant Objects. Amazon, 2017.

[5] Robert C. Martin. Clean Architecture. Prentice Hall, 2017. DOI: 10.5555/3175742.

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

[7] Steve McConnell. Code Complete. 2nd ed. Microsoft Press, 2004. DOI: 10.5555/1096143.

[8] Richard Pawson et al. Naked Objects. Wiley, 2002. DOI: 10.5555/599809.

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

You may also want to explore Yegor's blog posts, on his blog, his video lectures on YouTube, and Object Thinking meetup presentations on YouTube.