

Algorithms

...

YEGOR BUGAYENKO

Lecture #1 out of 10

90 minutes

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as website. Copyright belongs to their respected authors.



History

Original Intent

Abstraction

Enemies of Object Thinking

Read Watch

WARNING!

In the pursuit of academic enlightenment within this course, it is paramount to caution that the doctrines disseminated may present a potentially hazardous venture if employed in real-life software projects. This inherent risk arises from the potential incongruity with the broadly accepted canon of object-oriented programming and recognized best programming practices. If one remains resolute in their decision to adapt their coding methodologies to align with the principles propagated in this course, it would be prudent to employ a certain degree of foresight. A humorous, yet sincere suggestion, would be to secure alternate employment prior to a possible premature termination of one's current professional engagement.

Written by me, edited by ChatGPT

Chapter #1:

History

Who started it?



Ivan Sutherland's seminal **Sketchpad** application was an early inspiration for OOP, created between 1961 and 1962 and published in his Sketchpad Thesis in 1963. Any object could become a “master,” and additional instances of the objects were called “occurrences”. Sketchpad's masters share a lot in common with JavaScript's prototypal inheritance.

(c) Wikipedia

Who invented Objects, Classes, and Inheritance?



Simula was developed in the 1965 at the Norwegian Computing Center in Oslo, by Ole-Johan Dahl and Kristen Nygaard. Like Sketchpad, Simula featured objects, and eventually introduced classes, class inheritance, subclasses, and virtual methods. (c) Wikipedia

Simula-67: Sample Code

```
1 Class Figure;  
2   Virtual: Real Procedure square Is Procedure square;;  
3 Begin  
4 End;  
5 Figure Class Circle (c, r);  
6   Real c, r;  
7 Begin  
8   Real Procedure square;  
9   Begin  
10    square := 3.1415 * r * r;  
11   End;  
12 End;
```

Who coined the “Object-Oriented Programming” term?



Smalltalk was created in the 1970s at Xerox PARC by Learning Research Group (LRG) scientists, including Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Diana Merry, and Scott Wallace. (c) Wikipedia

Smalltalk: Sample Code

```
1 Object subclass: Account [  
2     | balance |  
3     Account class >> new [  
4         | r |  
5         r := super new. r init. ^r  
6     ]  
7     init [ balance := 0 ]  
8 ]  
9 Account extend [  
10     deposit: amount [ balance := balance + amount ]  
11 ]  
12 a := Account new  
13 a deposit: 42
```



“Everyone will be in a favor of OOP. Every manufacturer will promote his products as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.”

— Tim Rentsch,
Object Oriented Programming,
ACM SIGPLAN Notices 17.9, 1982

Who made it all popular?



C++ was created by Danish computer scientist Bjarne Stroustrup in 1985, by enhancing C language with Simula-like features. C was chosen because it was general-purpose, fast, portable and widely used.

You may enjoy watching this [one-hour dialog](#) of Dr. Stroustrup and me.

C++: Sample Code

```
1 class Figure {  
2     virtual float square() = 0;  
3 };  
4 class Circle : public Figure {  
5     Circle(float c, float r) : c(c), r(r) {};  
6     float square() { return 3.1415 * r * r; };  
7 private:  
8     float c, r;  
9 };
```



“There are as many definitions of OOP as there
papers and books on the topic”

— Ole Lehrmann Madsen et al.,
What Object-Oriented Programming May Be—And What
It Does Not Have to Be, ECOOP’89



“I made up the term ‘object-oriented,’ and I can tell you I didn’t have C++ in mind”

— Alan Kay, OOPSLA’97

There was an interesting debate between Alan Kay and a few readers of my blog, in the comments section under this blog post: [Alan Kay Was Wrong About Him Being Wrong](#)

What happened later?

C++ was released in 1985. And then...

Erlang 1986

Eiffel 1986

Self 1987

Perl 1988

Haskell 1990

Python 1991

Lua 1993

JavaScript 1995

Ruby 1995

Java 1995

Go 1995

PHP3 1998

C# 2000

Rust 2010

Swift 2014



“There is no uniformity or an agreement on the set of features and mechanisms that belong in an OO language as the paradigm itself is far too general”

— Oscar Nierstrasz,
A Survey of Object-Oriented Concepts, 1989

Incomplete list of OOP features, so far:

Polymorphism
Nested Objects
Traits
Templates
Generics
Invariants
Classes
NULL
Exceptions
Operators
Methods
Static Blocks
Virtual Tables
Coroutines

Monads
Algebraic Types
Annotations
Interfaces
Constructors
Destructors
Lifetimes
Volatile Variables
Synchronization
Macros
Inheritance
Overloading
Tuple Types
Closures

Access Modifiers
Pattern Matching
Enumerated Types
Namespaces
Modules
Type Aliases
Decorators
Lambda Functions
Type Inference
Properties
Value Types
Multiple Inheritance
Events
Callbacks

NULL Safety
Streams
Buffers
Iterators
Generators
Aspects
Anonymous Objects
Anonymous Functions
Reflection
Type Casting
Lazy Evaluation
Garbage Collection
Immutability



“Object oriented programs are offered as alternatives to correct ones... Object-oriented programming is an exceptionally bad idea which could only have originated in California.”

— Edsger W. Dijkstra, 1989



“C++ is a horrible language. . . C++ leads to really, really bad design choices. . . In other words, the only way to do good, efficient, and system-level and portable C++ ends up to limit yourself to all the things that are basically available in C.”

— Linus Torvalds, 2007
Creator of Linux



“OO seems to bring at least as many problems to the table as it solves”

— Jeff Atwood, 2007
Co-founder of Stack Overflow



“I think that large objected-oriented programs struggle with increasing complexity as you build this large object graph of mutable objects. You know, trying to understand and keep in your mind what will happen when you call a method and what will the side effects be.”

— Rich Hickey, 2010
Creator of Clojure

Thus, we don't know anymore what exactly is object-oriented programming, and whether it helps us write better code :(

Chapter #2:

Original Intent



“The contemporary mainstream understanding of objects (which is not behavioral) is but a pale shadow of the original idea and anti-ethical to the original intent”

— David West,
Object Thinking, 2004

You may enjoy watching our conversation with Dr. West: [part I](#) and [part II](#).

Chapter #3:

Abstraction

...

Chapter #4:

Enemies of Object Thinking

What makes us think as algorithms

- Global scope (static methods)
- Anemic objects (getters)
- Mutability (setters)
- Workers (“-er” suffix)
- NULL references
- Type casting (reflection)
- Inheritance

Chapter #5:

Read & Watch

Read and watch:

David West, Object Thinking, 2004

Yegor Bugayenko, Elegant Objects, 2016

“Object Thinking” meetup, watch on YouTube.