Object-oriented programming

History
=====

Simula 67
- class == block + name + instantiation
  - attribute == variables in block scope
  - member func == local functions
  - constructor == block runs until first detach (corutines)
  - local data/method is accessible via scope
  - garbage collection
- inheritance
- virtual function

Smalltalk (Alan Kay) from 70s.. (major version 1980)
- object
  - instance of class
  - store state (and it is not accessible even for the same class)
  - send/receive message
  - dynamic: object (structure) can be changed in runtime
- classes are first-class objects:
  they are created by sendig "subclass" message to class "Object"

Type
===
 - static type (what compiler sees)
 - dynamic/run-time type (what is the actual object)

Methodology
=========
- OO Analysis
  - abstraction
    - narrow the word to important features
  - classification
    - inner state messages to receive (and how to respond)
  - generalization/specialization

Class := data structure + methods + identity
   e.g. IndexError <: ColumnIndexError

{Package, Module} vs Object
 - both have encapsulation
 - but only the object is instantiated dynamically
Object vs Class ("static") attributes/methods

UML
- static model: structure of classes
- dynamic model: behavior of classes

Lifetime of objects
==========

Object creation
 - Constructor: set invariants, initialization
 - Factory
 - Cloning

Copy of objects
 - value semantics: C++, C# struct
 - reference semantics: Java, Scala, ADA
 (C++: copy constr., op=. ADA: limited type)

Object destruction:
 - ADA, Eiffel, Java, C#, ...: Garbage collection (+finalize; +dispose)
 - C++: destructor (also GC is possible, but not implemented)

Class attribute
 - Static data member
Static method
 - Static member function
 - Global (funcprog + current direction in C++)

Liskov substitutional principle
 - Barbara Liskov
 - 1987 conference keynote address titled Data abstraction and hierarchy.
 - Behavioral subtyping
 (other subtyping: structural subtyping)
 - Subtype Requirement: Let \phi(x) be a property provable about objects x of type T. Then \phi(y) should be true for objects y of type S where S is a subtype of T.
 - https://en.wikipedia.org/wiki/Liskov_substitution_principle

Inheritance
 - is multiple inheritance exists?
    - is exists only between interfaces
    - how to solve name collition
 - how diamond shape inheritance solved
   - Scala traits
   - C++ virtual/non-virtual base classes

Polymorphism
 - inclusion polymorphism
 - dynamic binding
 - virtual function
 - Mixin
 - CRTP

Safe redefinition in subclasses (Eiffel)