CSCE 240: Advanced Programming Techniques
Lecture 7: Object Oriented Concepts, UML

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

30TH JANUARY 2024

Carolinian Creed: "I will practice personal and academic integrity."

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lecture 7

- Introduction Section
  - Announcements
  - Recap of Lecture 6
  - Programming Language Trends
- Main Section
  - Concept: UML
  - Concept: Object methods
  - Concept: Encapsulation and restriction to access
  - Background for project: Chatbots
- Concluding Section
  - About next lecture – Lecture 8
  - Ask me anything

Introduction Section

Announcements

- TA office hours:
  Wednesday: 9:30 AM to 10:30 AM
  Friday: 2:30 PM to 3:30 PM

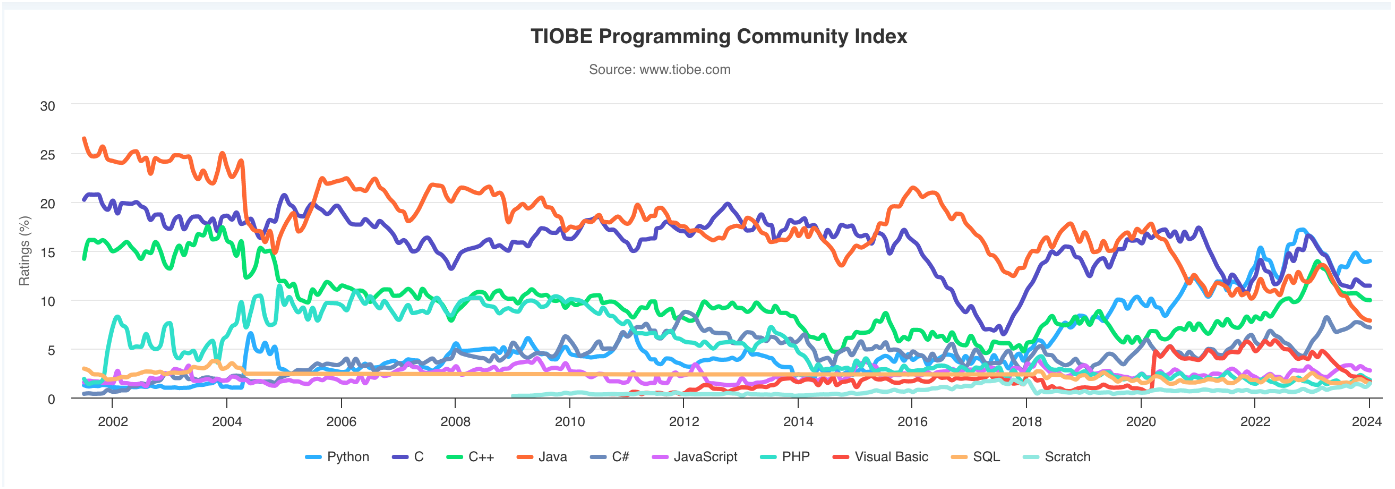
- Instructor office hours
  Monday: 1130 AM – 1230 PM
  Wednesday: 1130 AM – 1230 PM

Recap of Lecture 6

- We conducted peer review on Home Work #2
- Discussed objects v/s procedural view of problems
- Introduced Classes/ Objects

Programming Languages – Years Created

1957: Fortran
1958: LISP
1959: #COBOL
1964: BASIC
1972: C, SmallTalk
1978: SQL
1983: C++
1991: **Python**
1993: R
1995: PHP, **Java** & JS
2000: C#
2009: Go
2014: Swift



Source: https://www.tiobe.com/tiobe-index/

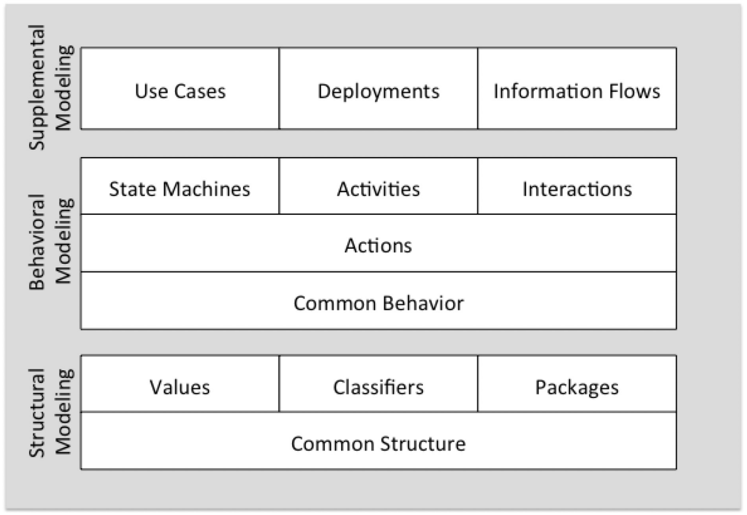
Source: https://devskiller.com/history-of-programming-languages/

Main Section

Concept:
Unified Modeling Language (UML)

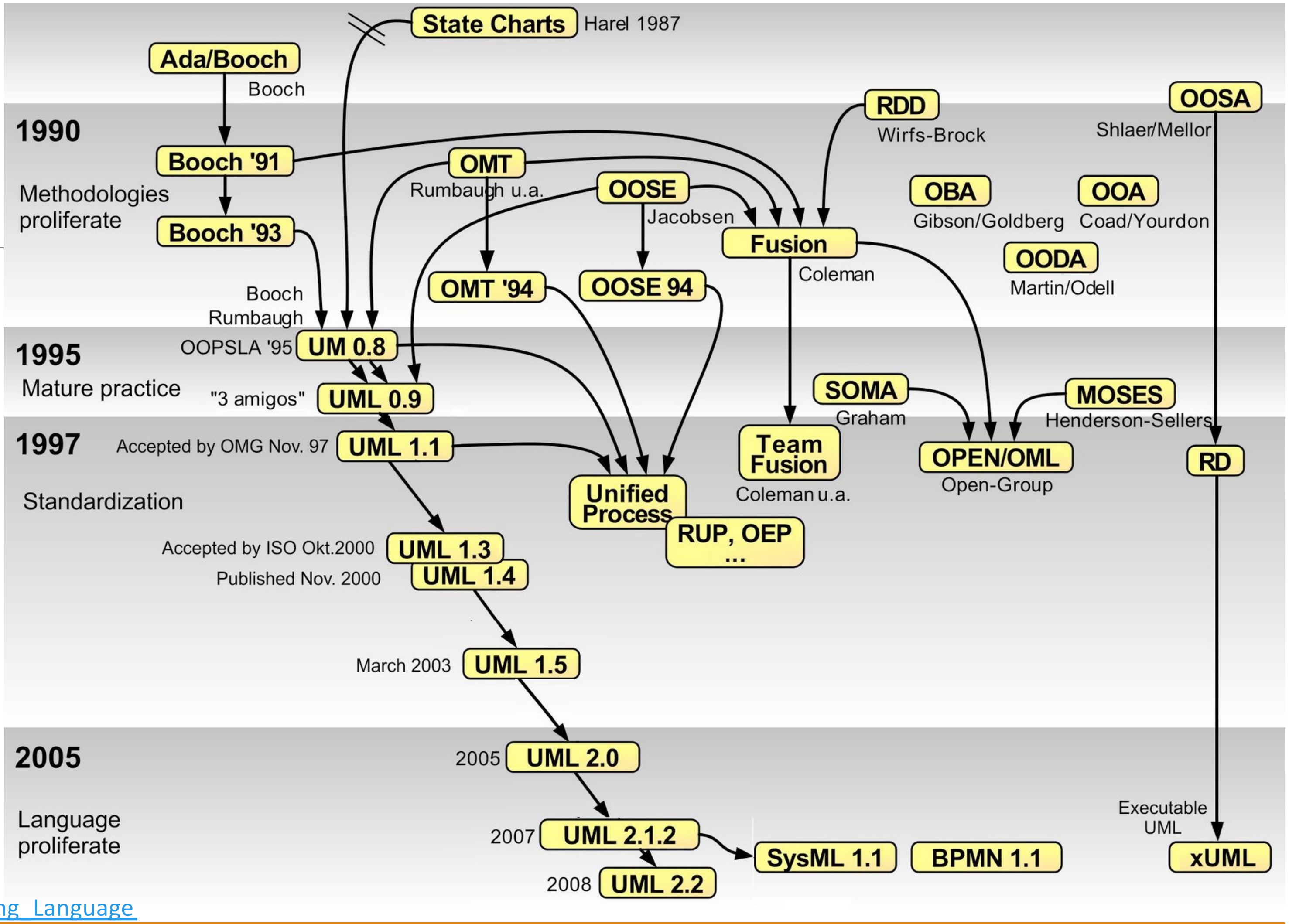
UML – What is it ?

- A visual, programming language independent, notation for communicating information about an Object-Oriented software's (static – structure, and dynamic - behavior) information
- Latest standard: https://www.omg.org/spec/UML/2.5.1/About-UML/
- Standardized
  - Object Management Group (OMG) adopted in 1997
  - International Organization for Standardization (ISO) published UML as an approved standard in 2005



Semantic Areas of UML
Figure credit: UML 2.5.1 Specification

UML History



Rumbaugh, Jacobson and Booch lead the efforts

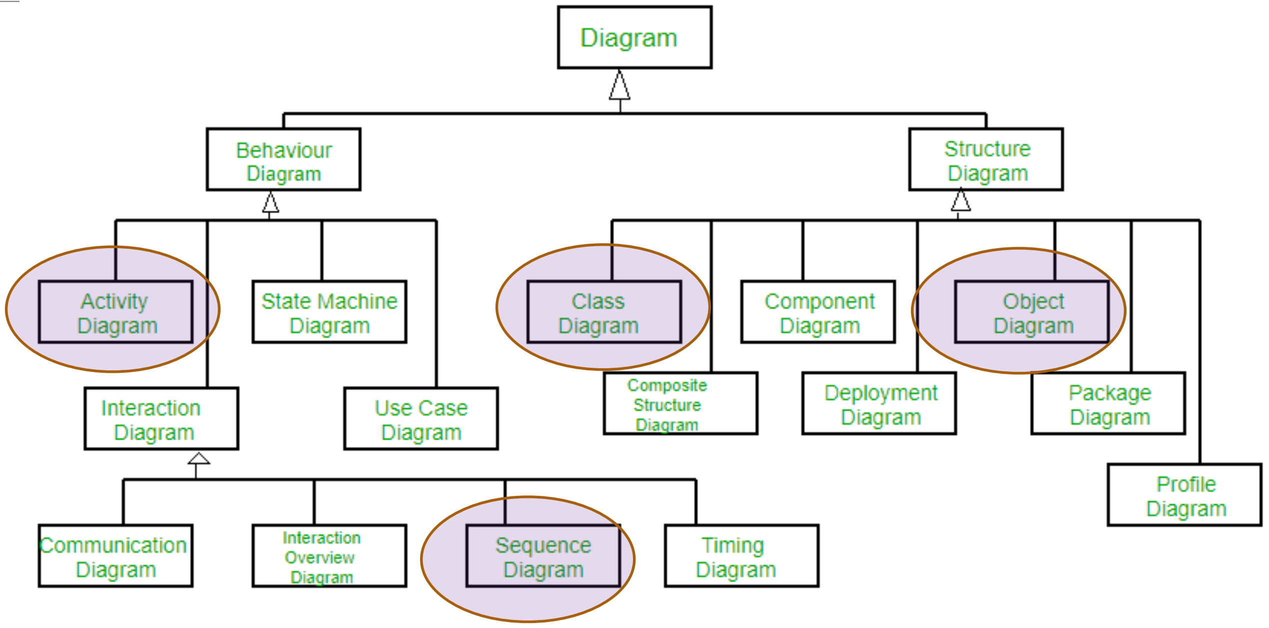
UML History

Figure credit: https://en.wikipedia.org/wiki/Unified_Modeling_Language

UML – Diagram Types

Most common types

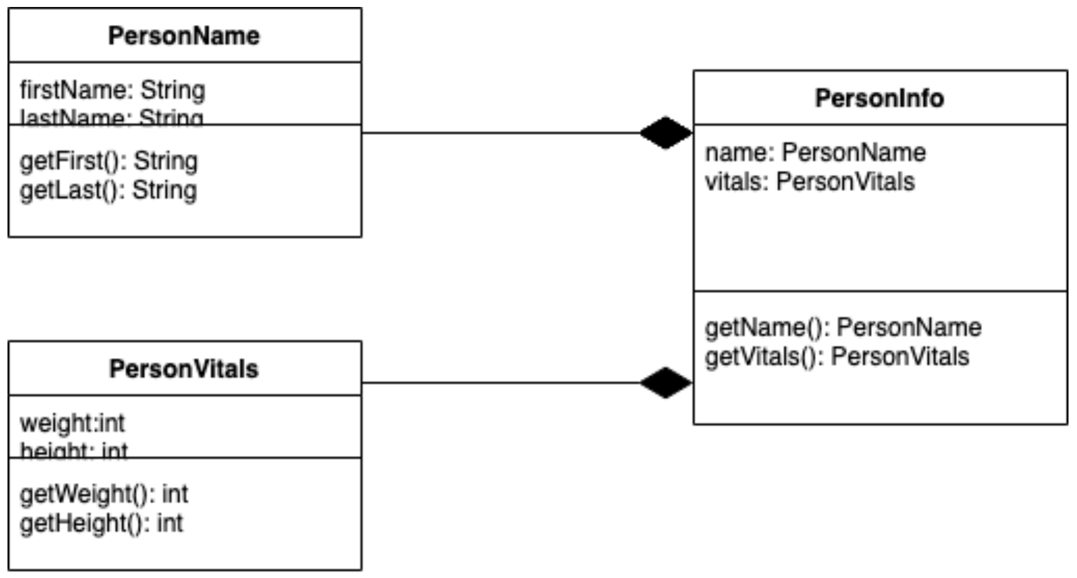
- Class and object diagrams
- Activity or sequence diagram



Types of Diagrams in UML 2.2

Figure credit: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

Example (Modified from Class 5/6)



Relationship Types

- Association: is related to
- Generalization: is a special type of
- Aggregation: is made up of, but can also exist independently
  - Example: Car and its wheels
- Composition: is made up of, but cannot exist independently
  - Example: Folder and its files

References:

1. UML 2.5.1 specs
2. Tutorial: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/

Tools for UML

- Many free and paid tools are available
  - See a recent review: https://www.gleek.io/blog/best-uml-tools.html
- We will use diagrams.net (at https://app.diagrams.net/)

Edit Example Class Diagram

- Using browser, go to: https://app.diagrams.net/
- Go to: File -> Open from -> Device -> and load file “Example.drawio”
  - File is at: https://github.com/biplav-s/course-adv-proglang/tree/main/sample-code/UML/Class7-examples
- Review and edit it
- You can save file or export the diagram in any supported format

Concept: Encapsulation

Encapsulation

- Organize
  - All the information related to a concept together
  - All the methods related to manipulation of the information related to the concept
- Illustration
  - Simple relational number
  - Functions for
    - Accessing data members
    - Utility functions

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/

Encapsulation – Access Restrictions

- Also called visibility rules, applies to both data members and functions
- Types
  - Public: any object can access
  - Protected: only object of same type or children can access
  - Private: only objects of same type can access
- **Demonstration**: SimpleRational

Code:

- https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class7and8_C%2B%2B_OO/src/implem/SimpleRelational.cpp
- https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class7and8_C%2B%2B_OO/src/Class7and8_C%2B%2B_OO.cpp

Discussion - Example

- Class Name
  - Title: String // no one can access
  - FirstName: String // no one can access
  - MiddleName: String // no one can access
  - LastName: String // no one can access
  - Suffix: String // no one can access
  - NickName: String // only descendants can access
  - PreferredName: String // anyone can access

Discussion: Course Project

Course Project – Knowing About Companies

- **Project**: Develop collaborative assistants (chatbots) that offer useful information about companies
- Specifically, use the EDGAR dataset on companies at: https://www.sec.gov/edgar/searchedgar/companysearch.
  - For Apple, it is: https://www.sec.gov/edgar/browse/?CIK=320193&owner=exclude
- **Each student will choose two companies (from thousand available).**
- Programming assignment programs will: (1) extract data about two companies from 10-k, (2) process it, (3) make content available in a command-line interface, (4) handle any user query and (5) report on interaction statistics.

Core Programs Needed for Project

- Prog 1: extract data from the 10-k report of a company filing
- Prog 2: process it based on questions
- Prog 3: make content available in a command-line interface
- Prog 4: handle any user query and
- Prog 5: report statistics on interaction of a session, across session

Programming Assignment # 1

- **Goal**: extract data from the companies of choice
  - Language of choice: Any from the three (C++, Java, Python)
- Program should do the following:
  - Take company / 10-k page (URL) as input
  - Read content about the 10-k page
    - a local text version of the report page // Store it as file with names <companyname>-<quarter-year>.txt
      // Optional: get reports for multiple quarters (say 3). Keep them as separate files with names <companyname>-<quarter-year>.txt
  - **Identify how may parts are there in the report** //Hint: You can search for a hardcoded string/ pattern
  - Report statistics of content: lines, words, chars, and parts.
  - Write content out in an output file formatted with indentation
- Code organization
  - Create a folder in your GitHub called "prog1-extractor"
  - Have sub-folders: src (or code), data, doc, test
  - Write a 1-page report in ./doc sub-folder
  - Send a confirmation that code is done to instructor, and update Google sheet

Chatbots - Background

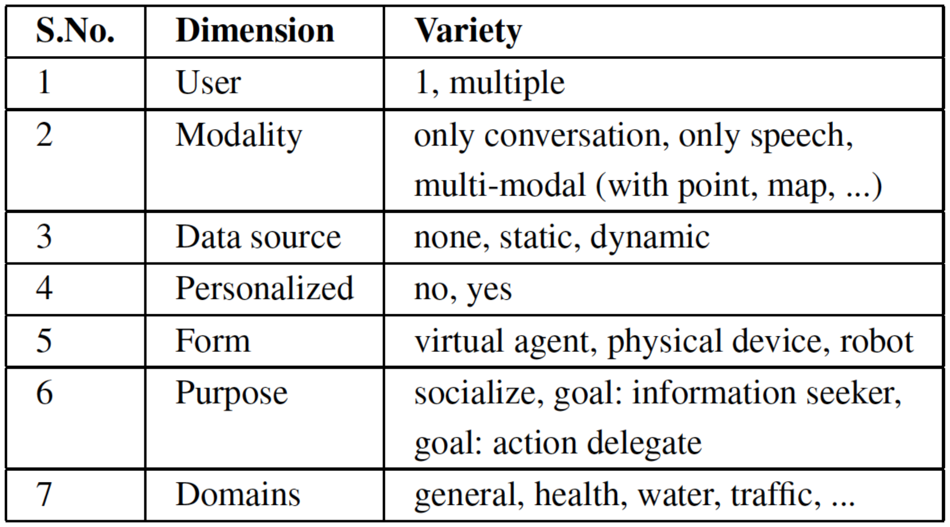
- Conversation agents and interfaces (chatbots) are getting easy to build and deploy
  - Can be text-based or speech-based
  - Usually multi-modal (i.e, involving text, speech, vision, document, maps)
- Current chatbots typically interact with a single user at a time and conduct
  - Informal conversation, or
  - Task-oriented activities like answer a user's questions or provide recommendations

Demonstrations

- *Eliza*, http://www.manifestation.com/neurotoys/eliza.php3
- *Mitsuku*, https://www.pandorabots.com/mitsuku/

Chatbots and Their Complexity

- **Users**: 1 or more; stay same or change over time
- **Modality**: variety of input data
- **Data:** static, changes, e.g. sensor data
- **Personalization**: Language, communication style, ..
- **Form**: interface variety
- **Purpose**: what does it help with?
- **Domains**: scope



S.No.	**Dimension**	**Variety**
1	User	1, multiple
2	Modality	only conversation, only speech, multi-modal (with point, map, ...)
3	Data source	none, static, dynamic
4	Personalized	no, yes
5	Form	virtual agent, physical device, robot
6	Purpose	socialize, goal: information seeker, goal: action delegate
7	Domains	general, health, water, traffic, ...

Unique Characteristics of a Chatbot
(Decision Support Collaborative Assistant)

- Should be useful for a purpose
- Should be able to work with any user utterance (example: text for a text-based chatbot)
- Optional, but desirable
  - User should be able to refer to the history of conversation in utterance
  - Improves performance over time
  - Model the user (understand impact of system on human) and not harm her

Discussion: Nature and Simplifications

- Once you select a company, the scope of answers is fixed.
- Some simplifications
  - **Download local copy** v/s web query
  - **Read static content first**
  - **Handle a subset of content**
  - **Have default handling for questions** the chatbot does not understand
- Do project in a language you are most comfortable with
- Use all advanced programming concepts to simplify coding

Suggested Scope is a Drastic Simplification

- **Users**: 1
- **Modality**: text
- **Data:** static (optionally: dynamic – voting history)
- **Personalization**: none
- **Form**: command line
- **Purpose**: information provider
- **Domain**: specific to companies and their 10-K report

Discussion: Chatbot Loop

- Input: from user (called utterance)
  - Problem specific query (i.e., about company risk factors)
  - Chitchat
  - Unrelated
- Output: from system (response)
  - Handle unrelated
  - Handle chitchat
  - Answer to query

- **Do it until user say over!**

Concluding Section

Lecture 7: Concluding Comments

- We introduced UML – a language independent notation for communicating about OO software
- Looked at concept of encapsulation
- Discussed background of chatbot

About Next Lecture – Lecture 8

Lecture 8: Object Oriented Continued, UML Notations

- Code organization for OO project
- Larger OO examples
- Project discussion – Programming Assignment #1 (due Thursday, Feb 1)