

### **Abstract**

This document is a specification of a development assignment that students of CS F301, Sem-I 2025-26, class need to perform and submit on or before 26 November. There will be a quiz on the assignment on 26 November, 5-6 pm, that will be evaluated.

# CS F301 PoPL: Functional Object Oriented Programming Assignment

Ramprasad S Joshi

November 2025

## 1 Introduction

So far we have done programming in procedural, modular, and OO fashion. Though many submissions to the previous assignments tell me that many students still do not understand even modular programming, let alone OO, *consciously* they have *tried* to do some assignments in what they perceive (and usually practice) as the OO fashion.

### 1.1 What is OO?

Though some of you may feel belittled when I drone again on these elementary questions that you assume to have already had under your belt, let me define the following as *my* expectation of Object Orientation:

- It must begin in the earliest stages of any project, definitely long before coding starts.
- Expression of use-cases and data-flows work-flows in UML and other such standard visual modeling paradigms is the culmination of this early stage OOn.
- You may need OOA (OO *Analysis*) before this stage, if you have not had enough prior experience with the particular application domain and the class hierarchies suitable for it.
- At the start and through coding, documentation must begin. For the top level design and each class, documentation (UML, method signatures and description of their capabilities and limitations and assumptions, *contracts and API*, etc.) **must** precede coding.
- Modularity must be maintained throughout. Modules and their interfaces must be completely specified as a part of the top-level design. In coding also the static main method must be written before any other code. If teamwork requires different individuals focusing on only their share of classes, then each one must at least read the main method and keep it with them to test (with stubs and skeletons for components not in their possession or control). Ideally, github-like versioning and comprehensive build environments must be shared by the team.
- In developing classes and class hierarchies, each decision (e.g. obedient inheritance or contain-&-delegate composition?) must be made consciously and recorded in the documentation *before* detailed implementation.

In a nutshell, object-orientation is deriving the HOW *from* the understanding and the specification of the WHAT. Contrast this against process-orientation in procedural programming (including just modular *without* OO) which the WHAT (data structures, procedure signatures and interfaces, etc.) is chosen according to the understanding and detailed specification of the HOW of procedures.

## 1.2 What is Functional?

Since this is a new topic, and most of you have not learnt any of it, I am not arguing *against* any of the habits you already have imbibed. I am not asking you to *unlearn* something. Instead, I am asking you to

- Assume that the HOW part is there somewhere, need not be detailed by you even *after* the specification of the WHAT part.
- Focus on the WHAT part, in human-language terms.
- Learn particular syntax and semantics of a functional language (or the lambdas in C++/Java) to specify and test the WHAT part accurately.

## 1.3 What is Functional-OO?

When we specify the WHAT part in functional programming in terms of higher and higher composites of more elementary abstractions, we infuse functional programming with OOn. To understand this, try describing QUICKSORT to school kids by demonstrating physically on people (sorting by heights) and books (sorting by titles alphabetically).

## 1.4 What is this assignment?

This is in continuation with all the previous assignments. In fact, it is the first week assignment, but now to be done in a systematic, pre-documented manner. It cannot be done in any other paradigm but Functional-OO (or OO-functional by using lambdas in C++/Java), I believe.

# 2 The Assignment

Provide a high-level (may be menu-driven and/or text command-driven or just a library API) interface to make use of your list structures (built during the last assignment before the MIDSEM if you are satisfied with it, or afresh if not) for reading, searching and sorting files and network streams. This also must include different aggregation methods (like counting inversions, making averages, mapping transformation functions to create new lists, etc.)

I want you to **make use of LLMs and other tools**, perhaps with the code pilots with your IDE, etc.  
F O C U S   O N   T H E   W H A T .

If your specification of the WHAT part was incomplete or sparse during the last assignment, maybe you'd want to first spruce it up before doing this.

## 2.1 The Deliverables

- A chronological, timestamped sequence of your design and development actions.
- Begin with your interpretation of specifications.
- Describe the help you obtained from online and GenAI sources, in detail, with the prompts used, the obtained responses, your interpretations (including doubts and grey areas left uninterpreted), the acquired insight from the interpretation, the method of incorporating the insight into your development, its fallout, whether the incorporation was successful and sustained, and the final insight acquired at the end of it all (success or failure).
- A demonstration-ready bundle of codes, in one or several languages, with all the build scripts or installers. A single command should suffice to install and run and test the whole bundle.
- A video (size less than 50 MB) showing the demo. If you can't fit it in 50MB, use a GitHub or YouTube channel or other cloud space with easy open access to anyone in BITS.

**An Example Use Case** Given a directory (native or cloud drive) full of text files (call them “data files”) having unstructured mixed quantitative and qualitative information, scrape all of it to list a set of keywords (given in a special “keyword file” input, just separated by whitespace) in decreasing order of their occurrence frequencies in those documents (in totality). Thus, for instance, you take the keyword (from the keyword file) that is the most frequent occurrence in the data files and put it out first with its total occurrence count, then the next most occurring keyword with its count, etc.

## 2.2 Evaluation Scheme

You are **allowed, in fact encouraged**, to work in groups (teams). One deliverable per team with a clear identification of the team at the top of the report file in the bundle. It has to be a zip less than 100MB total. The easier you make it for us to evaluate, the more credits you get for the same quality codes. Credit may get divided among the teammates based on the subjective assessment of the evaluators, which we will explain in each case nevertheless. So please be candid about the team contributions. Each teammate must know all the aspects described in the report, not only the ones related to one’s own contribution.

**There will be a quiz on 26 Nov 5-6pm that will only serve as a *sanity check*, ensuring that freeboarders can be made to confess themselves.** Plus, the comprehensive exam might have questions that test your claims of design choices and insights obtained at the end of it. Thus, the whole team is expected to be on one page at the outset of development as well as at the end. In between, there may be division of labour and division of attention. It’s like: converge on the design and the plan; split up (with coordination communication and synching mechanisms) to execute it; converge again at the deliverable stage.

**Comparison between different submissions will be done on the following parameters:**

1. Ease of testing, evaluation, interpretation of the report, consistency and lucidity in the report and between the report and the codes.
2. Features and flexibilities offered. (Extending the Data Abstraction and Object Orientation last assignment.)
3. Overall modularity, OOn in the HOW part, functional design at the top level.

## 2.3 Weightage

This assignment has half the weightage of the total of the assignments and quizzes component (i.e. 15% of the total course credits). However, those who want more for this as a compensation for their losses and lapses earlier, please engage meaningfully with the IC and the TA team, we can’t otherwise help you.