Queens College of CUNY
Department of Computer Science
Programming Languages
(CSCI 316)
Winter 2026

Assignment #8
"ADTs and OOP"
Due: January 16, 2026

Introduction:

In this assignment, we explore how classes/objects are expressed in different object-oriented programming (OOP) languages. As a proof of concept, we design and implement a Fraction ADT

Submissions:

In the Google form, please submit:

- Assignment08.pdf (composite of source code and output for all four programming languages)

Tasks:

Choose three OOP languages and identify installed or online IDEs which support them. (Generally you will use C++, Java, and Python, but you may use other OOP languages as well.)

Repeat the following for each language:

- Create a class with two attributes - numerator (int), denominator (int)
- Create and implement each of the following methods. You may slightly modify the signature to conform with stylistic or syntactic requirements of your chosen languages.
  - init(n, d) - returns a new fraction with numerator = n and denominator = d
  - print() - prints in the standard format numerator / denominator, e.g. 5/7
  - isValid() - checks that the denominator is not zero
  - simplify() - reduces the fraction to lowest terms, 2/4 becomes 1 /2
  - decimal(p) - returns a decimal approximation, rounded to p decimal places
  - reciprocate() - inverts the fraction to denominator / numerator
  - add(fraction) - adds the passed fraction to the current one - you will first need a common denominator
  - subtract(fraction) - subtracts the passed fraction from the current one - you will first need a common denominator
  - multiply(fraction) - multiplies the current fraction by the passed one
  - divide(fraction) - divides the current fraction by the passed one, equivalent to multiplying by the reciprocal
  - power(exp) - raises the fraction to the exponent (n/d)^e equivalent to n^e / d^e
  - egyptianFraction(* - returns a list of Fraction objects of the form 1/d1, 1/d2, 1/d3 etc. such that their sum is the original fraction
  - OPTIONAL name() - returns the way people would say it, e.g. "one twentieth", "two thirds", "three fourths"
- Test each of those functions for a few inputs. Perhaps run in a loop over a bunch of numbers…

For Egyptian Fractions, see:

- https://en.wikipedia.org/wiki/Egyptian_fraction
- https://www.geeksforgeeks.org/dsa/greedy-algorithm-egyptian-fraction/