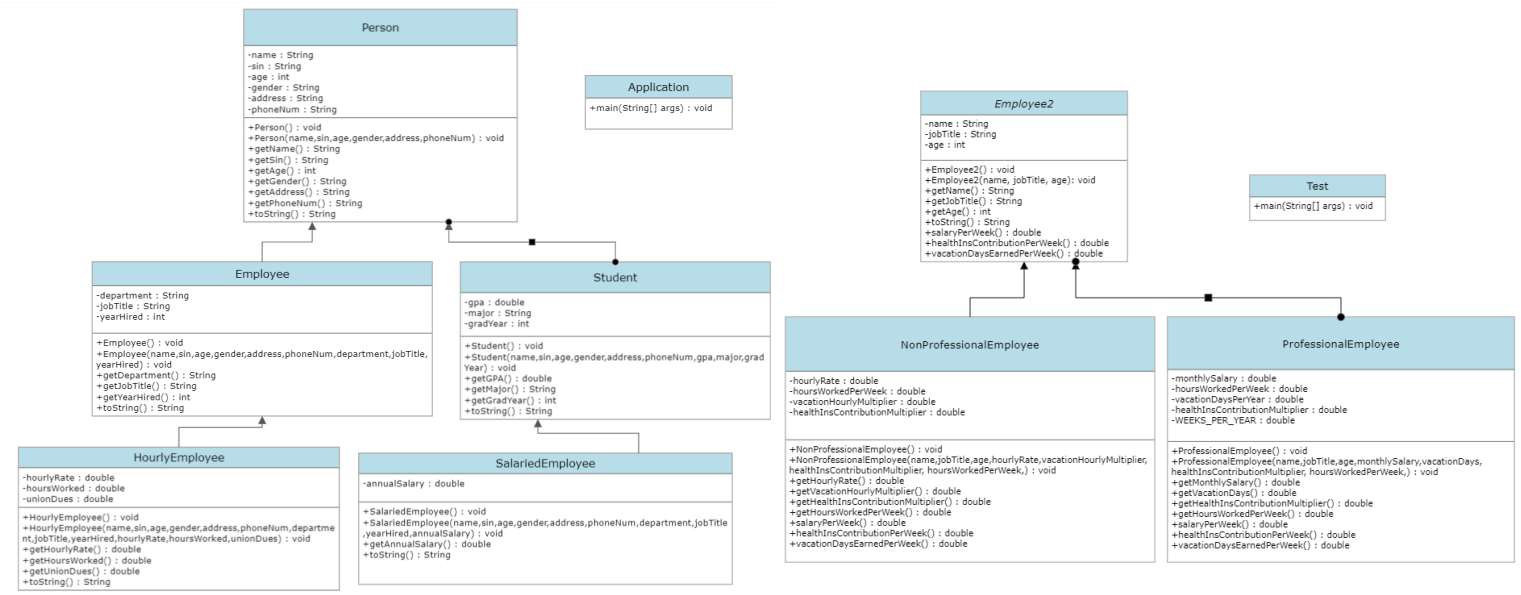
Juliane Bruck
#8297746

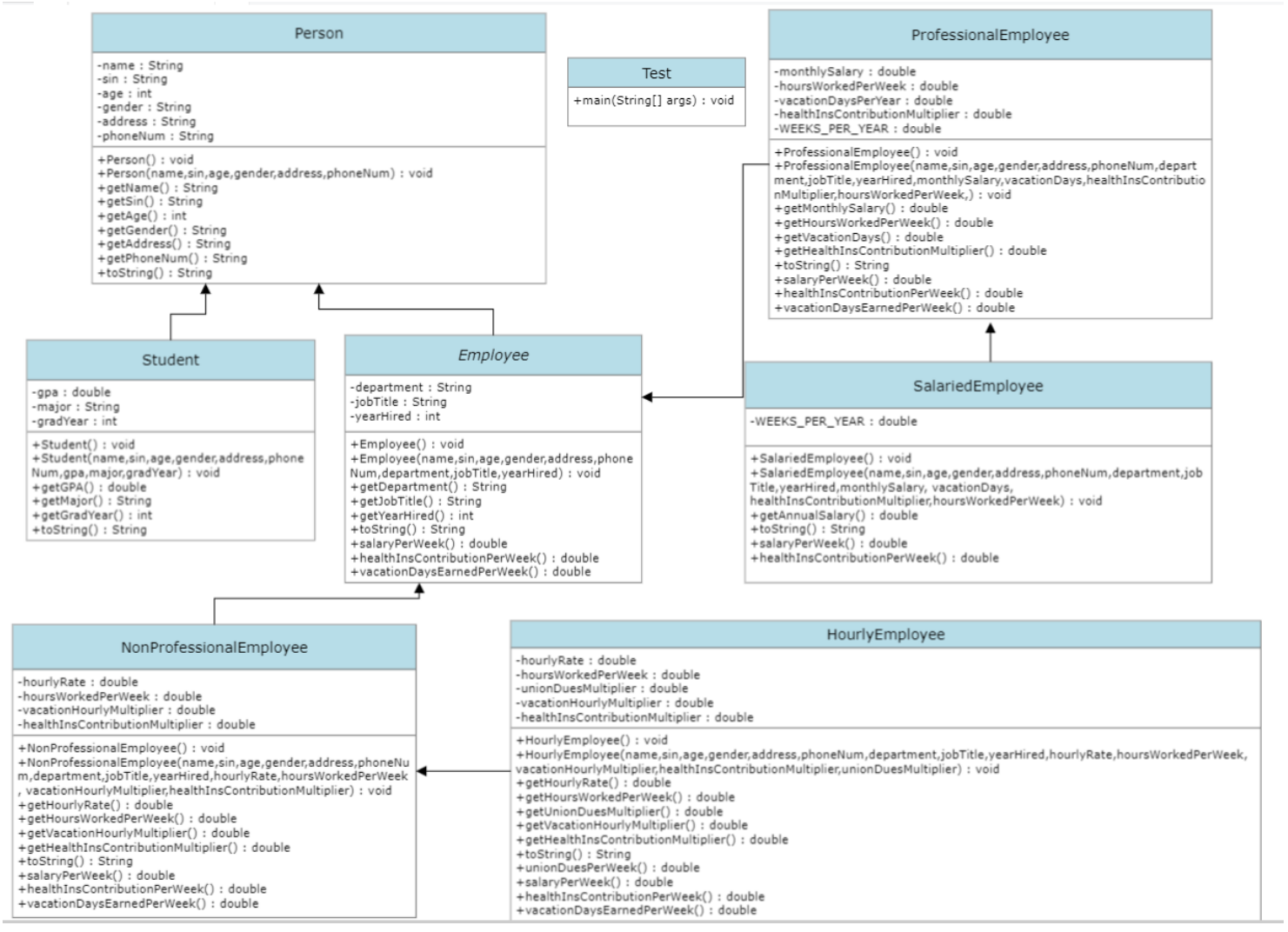
ASSIGNMENT 2

Submitted to
Professor Wassim El Ahmar
for the course
Introduction to Computing II
(ITI 1121)

University of Ottawa
June 12, 2023

a. UML diagram (from assignment 1 to assignment 2): (you may have to zoom in)





b. How the design and organization was improved from assignment 1:

In assignment 1, we had two different employee classes, Employee and Employee2, which means that Employee2 was not a subclass of class Person. Now, we have merged the two employee classes which means that the classes ProfessionalEmployee, NonProfessionalEmployee, SalariedEmployee, and HourlyEmployee are all subclasses of Employee. It was assumed that all employees had the methods salaryPerWeek(), healthCareContribution(), and vacationDaysEarnedPerWeek() which were declared as abstract methods in the abstract class Employee. This particular class hierarchy (SalariedEmployee inherits from ProfessionalEmployee and HourlyEmployee inherits from NonProfessionalEmployee) was chosen because of the variable unionDuesMultiplier in class HourlyEmployee which seemed to be a more specific variable.

This assignment could be further improved if we merged the SalariedEmployee class with the Professional Employee class as the only difference between the two are the monthlySalary variable and the annualSalary() method. We could also merge the NonProfessionalEmployee (if we supposed non professional employees had union dues) class with the HourlyEmployee class as the only difference is the unionDuesMultiplier variable and its related methods.

c. Different polymorphism examples in my implementation:

I had already used polymorphism in assignment 1, however there are more examples in my implementation for assignment 2. In any instance of any subclass of Employee, when the following methods are called: salaryPerWeek(), healthCareContribution(), vacationDaysEarnedPerWeek(). They are all overriding the abstract methods from the parent class Employee and in the case of HourlyEmployee or SalariedEmployee, they are overriding these methods in NonProfessionalEmployee and ProfessionalEmployee respectively. Furthermore, in the Test class, we use polymorphism when creating an array of type Person to iterate through many instances of the different subclasses(referencing each instance by its base type).

d. Lessons learned from assignments 1 and 2:

During both of these assignments, I have learned how to use abstract methods/ classes, how inheritance works (more precisely using super(), polymorphism/overriding, private vs. public variables), how to make UML diagrams, and object oriented guidelines (constructors, instance variables…etc.) In addition, I have learned that there are many ways to organize a program and achieve the same end result, however, by using class inheritance and polymorphism, we have a more organized and efficient(code reusability wise) structure. This can be more easy to understand and logical for new eyes looking at your program.