# CSCI 428 - Object Oriented Programming – Assignment 1

The goal of assignment 1 is to write a C# console application that:

* Accepts 4 parameters from the command-line:
  1. Full path/name of the input text file
  2. Name of the Pre-hires output file
     + Not a full path; e.g., PreHires.txt
  3. Name of the Employees output file
  4. Name of the Retirees output file
* Verifies command-line parameters are “reasonable”
* Reads all line in the text file into a string array
* Iterates through the array, splitting Pre-hires, Employees, and Retirees into 3 strongly-typed arrays of objects
* Sorts each array by last name using a classic bubble sort - written by you
  + No use of C# Sort or LINQ
* Writes the Pre-hires, Employees, and Retirees arrays to individual text files with the specified names
  + Files go in the same folder as the input file
  + If files of the specified name(s) already exist, they must be deleted (or overwritten)

## Implementation Requirements

The following design elements must be implemented to receive full credit on the assignment:

* Implement a FileReader class with:
  + a Read() method
  + PreHires, Employees, and Retirees array properties with the appropriate access level
    - Hint: Ensure encapsulation of the arrays
* Implement a Worker base class
  + Implement properties common to all worker types with the appropriate access level
  + Implement an override of ToString() throws a custom InvalidWorkerObject exception if called
  + Implement a ToXML() function that returns the XML for a Worker object
* Implement PreHire, Employee, and Retiree child classes that inherit from Worker
  + Implement properties unique to each class of worker on the appropriate class
  + Implement a further override of ToString() that formats the class as a string of the specified format (below).
  + Implement a ToXML() override that returns the XML for each class of worker.
* Implement a Sorter class with a single Sort() method that accepts an array of workers as an argument and sorts the array according to the following roles:
  + PreHire: offer extended date, employee ID
  + Employee: last name, first name, and employee ID
  + Retiree: retirement program, retirement date, employee ID
* Implement a single FileWriter class with a Write() method that accepts an array of workers and can write a file for all worker types.

## Input File Spec

The input file will be a tab-delimited text file with line breaks between records.

Fields:

* Worker type
* Employee ID
* First name
* Last name
* Offer extended date (Pre-hires only)
* Offer acceptance date (Pre-hires only)
* Employment date
* Job title (Employees only)
* Monthly salary (Employees only)
* Retirement program (Retirees only)
* Retirement date (Retirees only)

## Output File Specs

All output file specs will be XML format and must include all properties applicable to the worker type.

### Pre-hire Output File Spec

<PreHires>

<PreHire>

<EmployeeID>123</EmployeeID>

<EmploymentDate>4/2/2022</EmploymentDate>

…

<OfferAcceptanceDate>4/2/2022</ OfferAcceptanceDate >

</PreHire>

</PreHires>

### Employee Output File Spec

<Employees>

<Employee>

<EmployeeID>123</EmployeeID>

<EmploymentDate>4/2/2022</EmploymentDate>

…

<MonthlySalary>4/2/2022</ MonthlySalary >

</Employee>

</Employees>

### Retiree Output File Spec

<Retirees>

< Retiree>

<EmployeeID>123</EmployeeID>

<EmploymentDate>4/2/2022</EmploymentDate>

…

<RetirementDate>4/2/2022</RetirementDate>

</ Retiree>

</ Retirees>

## Bonus Points

While not necessary to receive full credit, discretionary bonus points may be awarded for (correct) use of XML serialization and/or reflection.