Concordia University

Software Engineering (BEng) – Co-op

Development Journal of C++ Project: HR Management System

Step-by-Step Thought Process

Djazy Faradj

Email: [djazy.faradj@outlook.com](mailto:djazy.faradj@outlook.com) GitHub: <https://github.com/Djazy-Faradj>

Undergraduate Student (Year 1, Term 1)

Project Start Date: September 23rd, 2024

Target End Date: October 7th, 2024

Project End Date: …

**Abstract**

This journal will document the development process for my second individual project in the Software Engineering program, where I design and implement a console-based Human Resources (HR) management system for a small college. This application models departments, teachers, and staff and demonstrates object-oriented programming (OOP) principles, data handling, and customized error handling. The goal is to improve my OOP skills while applying them in a real-world context.

In this project, I aim to:

1. Apply OOP concepts, such as inheritance, aggregation, and polymorphism.
2. Build a clear console-based interface for easy interaction with the HR system.
3. Implement file I/O to manage employee data.
4. Ensure error handling through custom exceptions.
5. Create documentation to assist future users of the system.
6. Develop test automation which, through test cases, will report actual outcomes with expected outcomes.

**Introduction**

This project involves the development of a Human Resources (HR) management system for a small college. The system is designed to model various departments, their staff, and teaching personnel, while incorporating key programming concepts such as object-oriented design and file handling. It will include department objects containing teachers (both full-time and part-time) and staff, with each department having a dean, who must be a teacher. The system will allow users to input and manage data through a console-based interface, storing information in text files for persistence between sessions. A primary feature of the project is a payroll computation system that calculates salaries for staff and teachers based on their roles, degree qualifications, and work hours. The project also focuses on robust error handling, with custom exceptions to prevent invalid or duplicate data entries. By leveraging inheritance, aggregation, polymorphism and interface implementation, this system serves as a comprehensive application of programming techniques learned throughout my C++ course. Ultimately, this project will provide practical experience in designing and implementing a functional software model for real-world HR management tasks. The primary objective of this project is to apply object-oriented programming (OOP) concepts such as inheritance, aggregation, and polymorphism to build a functional HR management system. A clear console-based interface will be developed for easy user interaction, and file input/output operations will be implemented to manage employee data effectively. Additionally, custom exception handling will be used to ensure robust error management. I will also write the program’s pseudocode on Notepad++ and complete UML class diagrams for all classes and interfaces using Microsoft Visio. Finally, thorough documentation will be created to guide future users in navigating and using the system. This project will train all the fundamental aspects required in a software engineer's skill set.

**Initial Application Concept**

The HR Management System will:

* Model department objects containing lists of teachers (both full-time and part-time) and staff members.
* Handle payroll calculations for each type of employee (full-time, part-time, or staff)
* Implement a user interface using a console-based interaction model to allow for simple input and output of data.
* Include error handling for attempts to add employees to nonexistent departments or duplicate entries.

**Implementation Details**

1. Implement an abstract class Person that includes at least five shared fields and contains at least one abstract method of your choice (for example to define a person's category (teacher or staff)).
2. Extend class Person with concrete classes and override the toString() and equals() methods of each class.
3. The class Teacher has two instance variables: specialty and degree and some other fields for its subclasses (part-time or full-time teachers)
4. The class Staff has also two instance variables: duty and workload
5. Implement an interface PayRoll that contains ComputePayRoll() method to be implemented as follows: a. For each full-time teacher, the salary is computed as (32 \* degreeRate \* 2) \* 0.85, where degreeRate is 112, 82, 42 for PhD, Master, and Bachelor, respectively.  
   b. For each part-time teacher, the salary is computed as (hoursWorked \* degreeRate \* 2) \* 0.76, where degreeRate is 112, 82, 42 for PhD, Master, and Bachelor, respectively.  
   c. For each member of staff, the salary is computed as (workload \* 32 \* 2) \* 0.75, where the workload is the weekly working hours. The working hours cannot exceed 40.  
   d. Implement the method ComputePayRoll() inside Teacher and Staff classes.
6. Teachers and staff are added/assigned to a department, first by loading/reading from a text file during the first execution of the application. Moreover, new teachers and staff must be added/assigned through the GUI application.
7. Trying to add a teacher or a staff member to an inexistent department (based on department id) should rise/throw an exception.
8. Trying to add a teacher or a staff member that already exists/is added (based on the id) to the department should throw an exception.
9. A department class has a list of teachers and staff. A department class also has a dean, who should be a teacher of that department, otherwise, an exception must be thrown.
10. Each new teacher and staff added to a department should be stored in the text file before exiting the application, generating a new version of the text file. So, if the text file does exist (which is the case), new teachers or staff should be appended. There are files of teachers and staff.
11. Customized exception handling for the cases above must be implemented.
12. You should be able to add Teachers/Staffs/Departments through the console and the info should be saved in the corresponding text file. Next time the application is run, it should load the previously entered info.

**A Few Notes**

1. The pseudocode that you are asked to do could be in a Word document format, only for the methods that are doing some calculations. You don't need to write it for getters/setters, toString(), equals(), etc.
2. You must write all the methods inside your UML diagrams excluding getters/setters, toString(), etc.
3. A teacher's specialty is something like "Computer Science" and a teacher's degree is a "Master's".
4. Part-time or full-time is a field in the Teacher's class
5. A Dean is a special type of teacher.

**Bibliography**

**Appendix**