# FPT University

# Java OOP Project

# Report

*Member: Trần Thành Nhân, Nguyễn Minh Tú*

## Problem Description

* 1. **Person**
     1. A person object has three attributes: code, name, address;

Some methods: constructors, getters, setters, and overriding method toString() returns formatted String: “*code* | *name* | *address*”.

* + 1. All person objects have a common attribute: TAX, with initial value is 10%.
  1. **Student** extends **Person** implements **Comparable**
     1. A student has two more attributes: grade (Double) and valid (Date). Field valid contains the enrolled date when that student started to study.
     2. Methods of student: constructors, getters, setters, overriding method toString() returns formatted String: “*code* | *name* | *address* | *grade*”, overriding method compareTo() compares the grade of this student to other student’s grade; method isValid() check if number of study years exceed 6 years to the current date.
  2. **Professor** extends **Person**
     1. Attributes of professor:
* Enum PositionEnum (PROFESSOR, ASSOCIATE\_PROFESSOR, HONOR\_PROFESSOR).
* Enum EducationLevel (BACHELOR, MASTER, DOCTOR).

Attributes: experience (int, > 0), basicSalary (int, default = 1000), position (PositionEnum), education (EducationLevel).

Attributes to manage a list of student: arr (List <**Person**>), count (int): holds the number of available students in arr.

* + 1. Basic methods of professor: constructors, getters, setters, overriding method toString() returns formatted String “*code* | *name* | *address* | *experience* | *realSalary*”.
    2. Common attributes of all professors:
* COF = 0.33; STEP = 3;
* SUPPOS = 0.5 or 1.25 or 0.75, depends on position is ASSOCIATE\_PROFESSOR, PROFESSOR or HONOR\_PROFESSOR correspondingly.
* SUPLEV = 0.1 or 0.25, depends on education is MASTER or DOCTOR, correspondingly.
  + 1. Common methods of all professors:
       - getRealSalary(Professor) = basicSalary \* (experience / STEP) \* (1+COF) \* (1+SUPPOS) \* (1+SUPLEV)
       - getAnnualIncome(Professor) = getRealSalary(Professor) \* 12 \* (1-TAX)
    2. Management methods of professor: to manage student list
       - boolean add(Student x): add a student x to arr and return true if added successfully, otherwise return false.
       - void addAllStudent (File f): access input file f and add all students to arr with student’s format.
       - void remove (String studentCode): remove a student with studentCode in arr.
       - void update (String studentCode): update information (name, address, grade, valid) of a student with studentCode in arr.
       - void removeInvalid(): method will find all students who have invalid date (more than 6 years from the enrolled date to the current date) and then remove them out of arr.
       - Student find(String findCode): This method will return a student who has code == findCode in the list arr.
       - void displayAllStudents(): display all students available in student list arr in ascending order of code.
       - void sortDescending(): use Bubble Sort to sort the list arr of students in descending order of code.
       - Student[] getTopGrade(): return a list of students who have the most grade in arr list.
       - Student[] getBottomGrade(): return a list of students who have the least grade in arr list.
       - void showDistribution(): display a statistic of the distribution of grade, with data retrieved from arr.
       - double averageGrade(): returns the average grade of all students in arr.
       - void showDistinctGrade(): display a statistic of all students whose grade are unique.
       - void showDuplicateName(): display all students’ names which are names of at least two students.
       - void showDistinctName(): display all students’ names which are name of exactly one student.
       - void outAllStudents(File f): store all student in arr and save in output file f.
  1. Data text file and Data format
     1. A text file, named *in\_students.txt* contains a number of lines.

The first line represents data of a professor: the owner of student list. Data of professor has format: “*code | name | address | experience | basicSalary | position | education | count*”, where count is the number of students in the list.

Each of the following lines represents a student’s data in format: “*code | name | address | grade | valid*” separated by a single vertical bar.

Valid has format: *day/month/year.*

Code has format: see the section below.

* + 1. A text file, named *out\_students.txt*, stores data from students list arr to the file when the method outAllStudents() performed. Stored data have the same format like data in file *in\_students.txt*.
    2. Student code has format: *STxxx*, where xxx is a 3-digit number in range [001, 999] inclusive.
    3. Professor code has format: *PRxxx*, where xxx is a 3-digit number in range [001, 999] inclusive.
    4. Person name must be in standard format: remove all leading and ending spaces; all spare spaces between words; only the first character of each word is uppercase.
  1. Application  
     Write a Java application with the following user interface, allows user to manage a professor’s list of students. Supported functions:
     + Add a student to professor’s list, retrieve data from keyboard.
     + Import all students from an input file to professor’s list.
     + Remove a student in the list by student’s code.
     + Remove all invalid students.
     + Update data of a student in the list by student’s code.
     + Find a specific student in the list by student’s code.
     + Display all students in ascending order of student’s code.
     + Display all students who have the most grade.
     + Display all students who have the least grade.
     + Display the average grade of all students.
     + Display distribution of grade.
     + Display the distinct grade of all students in the list.
     + Display all students who have the same name.
     + Display all students who have unique name.
     + Export the student list to an output file.
     + Quit.

## Analysis. From the problem description, the following user-cases are idendified:

User

Add a student

System

Export students to file

Display unique names

Display duplicate names

Display grade distribution

Display distinct grade

Display bottom students

Display top students

Display all student

Find a student

Import students from file

Remove a student

Remove invalid students

Update a student

Display average grade

## Design

|  |  |
| --- | --- |
| **Concept** | **Class** |
| Person | Person (code, name, address) |
| Student | Student (grade, valid) extends Person implements Comparable |
| Professor | Professor (experience, basicSalary, position, education, Student[]) extends Person |
| Menu | A manage student menu for each professor |
| Program | Manage program |

**Class detail**

|  |
| --- |
| **Person** |
| * static double TAX = 10% * code: String * name: String * address: String * Person() * Person (String, String, String) * Getters, Setters * String toString(): “code | name | address” |

|  |
| --- |
| **Student extends Person** |
| * grade: double * valid: date * Student() * Student(String, String, String, double, Date) * Getters, Setters * String toString(): “code | name | address | grade” * int compareTo(Student s): compare grade between 2 students * boolean isValid(): valid date is more than 6 years than current date |

|  |
| --- |
| **Menu** |
| * void displayMenu() * int getChoice() |

|  |
| --- |
| **Manage Program** |
| * void main(String[] args) |

|  |
| --- |
| **Professor extends Person** |
| * PositionEnum: PROFESSOR, ASSOCIATE\_PROFESSOR, HONOR\_PROFESSOR * EducationLevel: BACHELOR, MASTER, DOCTOR * experience: int * basicSalary: int * position: PositionEnum * education: EducationLevel * String [] student * count: int * Professor() * Professor(int, int, PositionEnum, EducationLevel, Student[], int, String, String, String) * Getters, Setters * String toString(): return “code | name | address | experience | realSalary” * COF = 0.33 * STEP = 3 * SUPPOS = * SUPLEV = * realSalary = basicSalary \* (experience/STEP) \* (1+COF) \* (1+SUPPOS) \* (1+SUPLEV) * annualIncome = realSalary \* 12 \* (1-TAX) * boolean addStudent(Student s): return true if add successful * boolean addAllStudent(File f): read all Student from file, return true if successful * void removeStudent(String code): remove student by student code * void update(String code): update information of a student by student code. * void removeInvalid(): remove Students has valid date is more than 6 to current * Student find(String fcode): return student match the fcode * void displayAllStudents(): display all student in ascending order of student code * void sortStudent(): sort list of students in descending * Student[] getTopGrade(): return Students have the most grade * Student[] getBottomGrade(): return Students have the least grade * void showDistribution(): display all distribution of grade * double averageGrade(): return average grade * void showDistinctGrade(): display all distinct grade * void showDuplicateName(): display all student have the same name * void showDistinctName(): display all distinct student name * boolean outAllStudents(File f): write all student into file, return true if successful |

(+: public members, -: private members)

**User Interface:**

*Menu of the program will be seen as:*

Enter input data file name: …………………..

File ……… not found!

Enter input data file name: …………………..

Data from file has been loaded successul!

========== Manage Student Program ===========

1. Add student
2. Remove student
3. Update student
4. Find student
5. Display students
6. Display staticstic
7. Export student list to file
8. About Professor
9. More display options
10. Quit

*Sub-menu of each option:*

* Option 1 (Add):
  1. Add a new student
  2. Import student list from file
* Option 2 (Remove):
  1. Remove a student
  2. Remove invalid students
* Option 5 (Display):
  1. Display students sorted by student code (ascending)
  2. Display students sorted by student name (ascending)
  3. Display students sorted by student grade (descending)
  4. Display highest grade students
  5. Display lowest grade students
* Option 6 (Statistic):
  1. Display average grade
  2. Display distribution of grade
  3. Display distinct grade
  4. Display duplicate student names
  5. Display unique student names
* Option 8 (About professor):
  1. Update information of professor
  2. Display information of professor
* Option 9 (More display options):
  1. Display students have grades higher than average
  2. Display students have grades lower than average
  3. Display n-th year students
  4. Display number of students enrolled in a specific year

## Implementation

1. **Initial data of the program (file *initial data.txt*)**

Please explore the software structure

1. **Software**

Please run the program

## Testing

## Task list

1. **Member’s Tasks**
   1. Trần Thành Nhân
      1. Problem Description
      2. Analysis
      3. Implementation
         1. Enum EducationLevel (20%)
         2. Enum PositionEnum (20%)
         3. Class Student (30% methods)
         4. Class Person (20% methods)
         5. Class Project (include main class) (50% methods)
         6. Class Professor (50% methods)
         7. Additional support class (80% methods)
         8. Class Menu (80%)
   2. Nguyễn Minh Tú
      1. Design
      2. Testing
      3. Implementation
         1. Enum EducationLevel (80%)
         2. Enum PositionEnum (80%)
         3. Class Person (80%)
         4. Class Student (70%)
         5. Class Project (50%)
         6. Class Professor (50%)
         7. Additional support class (20%)
         8. Class Menu (20%)
2. **Contributing rate**
3. Trần Thành Nhân: 50%
4. Nguyễn Minh Tú: 50%