# 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 EducationLevelEnum (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) \* COF \* SUPPOS \* SUPLEV
       - getAnnualIncome(Professor) = getRealSalary(Professor) \* 12 \* 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 code): remove a student with studentCode in arr.
       - void update (String code): 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 fCode): This method will return a student who has code == fCode 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.
       - boolean outAllStudents(File f): store all student in arr and save in output file f, return true if successful.
  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 space.

Address has format: *number-street-district-city.*

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 identified:

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(String, String, String, int, int, PositionEnum, EducationLevel, Student[] arr, int) * Getters, Setters * String toString(): return “code-name-address-experience-realSalary” * COF = 0.33 * STEP = 3 * SUPPOS = * SUPLEV = * realSalary = basicSalary \* (experience/STEP) \*COF\*SUPPOS\*SUPLEV * annualIncome = realSalary \* 12 \* 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 your professor name: …………………..

Not found, Please enter again: ……………….

Found, all students has been loaded successfully. Students manage program:

1- Add new student

2- Remove a student by student code

3- Remove invalid date student

4- Update student data

5- Sort all students

6- Display all students who has the most and the least grade

7- Display average grade

8- Save to file

9- Quit

## Implementation

## Testing

## Task list