

Assignment

MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

MÃ MÔN: 503005

Đề bài: Module Quản lý nhân viên trong công ty IT

(Sinh viên đọc kỹ tất cả hướng dẫn trước khi làm bài)

I. Giới thiệu bài toán

Một công ty cung cấp phần mềm cần quản lý nhân viên trong công ty. Trong bài tập này, sinh viên sẽ phải hiện thực một số module (thành phần) chức năng trong hệ thống quản lý nhân viên. Đối tượng chính cần quản lý là Lập trình viên (Developer) và nhân viên kiểm thử (Tester).

Các chức năng mà sinh viên phải thực hiện là:

- Đọc file danh sách nhân viên.
- Trả về danh sách nhân viên theo điều kiện.
- Trả về nhân viên theo điều kiện.
- Sắp xếp danh sách nhân viên.

II. Tài nguyên cung cấp

Source code được cung cấp sẵn bao gồm các file:

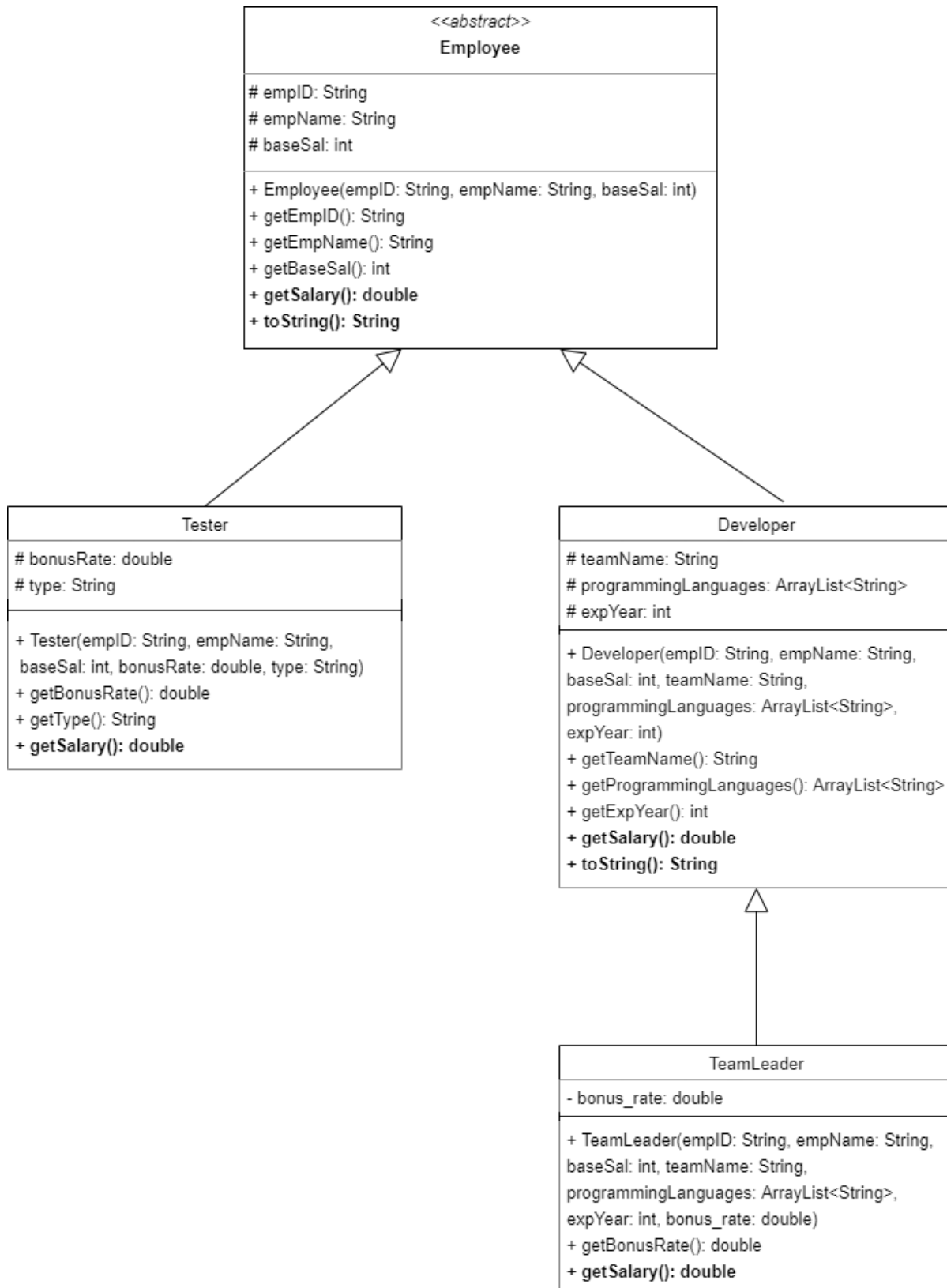
- File đầu vào và kết quả mong muốn:
 - o Folder *input* gồm *ListOfEmployees.txt*: chứa danh sách nhân viên, *PLInfo.txt* chứa ngôn ngữ lập trình mà lập trình viên thành thạo.
 - o Folder *output* gồm: 5 file *Req2.txt*, *Req3.txt*, *Req4.txt*, *Req5.txt*, *Req6.txt* chứa kết quả mẫu của các yêu cầu trong bài.
- File mã nguồn:
 - o *TestCM.java*: tạo đối tượng và gọi các phương thức sinh viên sẽ định nghĩa.
 - o *Employee.java*: chứa lớp **Employee** được định nghĩa sẵn. File này sinh viên không được chỉnh sửa.

- *CompanyManagement.java*: chứa lớp **CompanyManagement** được định nghĩa sẵn thuộc tính *empList* để chứa danh sách sinh viên, phương thức khởi tạo, phương thức ghi file và một số phương thức trống. Sinh viên sẽ hiện thực thêm vào các phương thức còn trống trong file này và không chỉnh sửa phương thức có sẵn.

III. Trình tự thực hiện bài

- Sinh viên tải file tài nguyên được cung cấp sẵn và giải nén.
- Trong cùng thư mục với 3 file cung cấp sẵn sinh viên tạo ra 3 file tương ứng với 3 lớp con là **Developer**, **Tester**, **TeamLeader**.
- Sinh viên tự hiện thực lớp **Developer**, **Tester**, **TeamLeader** và code thêm vào lớp **CompanyManagement** theo mô tả và yêu cầu trong các phần tiếp theo.
- Sau khi hiện thực các phương thức trong lớp **CompanyManagement**, sinh viên biên dịch và chạy với phương thức **main** trong file *TestCM.java* đã gọi sẵn các phương thức. Sinh viên thực hiện các yêu cầu ở mục dưới và so sánh kết quả output của mình với kết quả trong folder output đã được cung cấp sẵn.
- **Đối với các yêu cầu sinh viên không làm được vui lòng không xóa và phải đảm bảo chương trình chạy được với phương thức main trong TestCM.java được cung cấp sẵn.**

IV. Mô tả lớp Employee, Developer, Tester, TeamLeader



| CompanyManagement |
|--|
| - empList: ArrayList<Employee> |
| + CompanyManagement(path: String, path1: String) + getEmployeeFromFile(path: String, path1: String): ArrayList<Employee> + getDeveloperByProgrammingLanguage(pl: String): ArrayList<Developer> + getTestersHaveSalaryGreaterThan(value: double): ArrayList<Tester> + getEmployeeWithHigestSalary(): Employee + getLeaderWithMostEmployees(): TeamLeader + sorted(): ArrayList<Employee> + printEmpList(): void + writeFile(path: String, list: ArrayList<E>): boolean + writeFile(path: String, object: E): boolean |

- Lớp **Developer** và lớp **Tester** kế thừa từ lớp **Employee**, lớp **TeamLeader** kế thừa từ lớp **Developer**.
- Giải thích một số thuộc tính và phương thức như sau:
 - Lớp **Employee**
 - *empID*: mã nhân viên.
 - *empName*: tên nhân viên.
 - *baseSal*: lương cơ bản của nhân viên.
 - Phương thức abstract **getSalary()**.
 - **toString()**: trả về chuỗi theo định dạng: *empID_empName_baseSal*
 - Lớp **CompanyManagement**
 - *empList*: danh sách nhân viên
 - **CompanyManagement(String path, String path1)**: phương thức khởi tạo, khởi tạo *empList* bằng cách gọi phương thức đọc file.
 - **getEmployeeFromFile(String path, String path1)**: đọc từ file vào danh sách *empList* với *path* là đường dẫn đến file *ListOfEmployee.txt* chứa danh sách nhân viên và *path1* là đường dẫn đến file *PLInfo.txt* chứa danh sách ngôn ngữ lập trình mà mỗi nhân viên thành thạo.
 - **getDeveloperByProgrammingLanguage(String pl)**: trả về danh sách lập trình viên thỏa điều kiện thành thạo ngôn ngữ **pl** truyền vào.
 - **getTestersHaveSalaryGreaterThan(double value)**: trả về danh sách nhân viên kiểm thử có tổng lương lớn hơn giá trị tham số truyền vào.
 - **getEmployeeWithHigestSalary()**: trả về nhân viên có lương cao nhất trong danh sách nhân viên.
 - **getLeaderWithMostEmployees()**: trả về trưởng nhóm của nhóm có nhiều lập trình viên nhất.
 - **sorted()**: trả về danh sách nhân viên *empList* đã được sắp xếp.
 - **printEmpList()**: in danh sách *empList* ra màn hình command prompt.

- **writeFile(String path, ArrayList<E> list):** ghi file danh sách nhân viên (phương thức này sinh viên không được sửa).
- **writeFile(String path, E object):** ghi file nhân viên theo yêu cầu. (phương thức này sinh viên không được sửa).

○ **Lớp Developer**

- *teamName*: tên nhóm của lập trình viên. Mỗi lập trình viên chỉ thuộc 01 nhóm.
- *programmingLanguages*: danh sách các ngôn ngữ lập trình mà lập trình viên thành thạo.
- *expYear*: số năm kinh nghiệm của lập trình viên.
- **getSalary()**: trả về lương, sẽ được mô tả ở bên dưới.
- **toString()**: trả về chuỗi theo định dạng:

empID_empName_baseSal_teamName_programmingLanguages_expYear

programmingLanguages theo định dạng:

[programmingLanguages1,programmingLanguages2,...]

(Sinh viên xem thêm trong file output kết quả để xác định đúng định dạng)

○ **Lớp Tester**

- *bonusRate*: tỉ lệ thu nhập tăng thêm.
- *type*: loại nhân viên kiểm thử (Automation Test - AM/Manual Test - MT).
- **getSalary()**: trả về lương, sẽ được mô tả ở bên dưới.

○ **Lớp TeamLeader**: TeamLeader cũng là một lập trình viên, mỗi nhóm chỉ có duy nhất 01 TeamLeader.

- *bonus_rate*: tỉ lệ thu nhập tăng thêm.
- **getSalary()**: trả về lương, sẽ được mô tả ở bên dưới.

- Mô tả phương thức tính lương:

○ **Developer**

- Số năm kinh nghiệm ≥ 5 :
 - Lương = lương cơ bản + số năm kinh nghiệm * 2.000.000 (2 triệu).
- $5 > \text{Số năm kinh nghiệm} \geq 3$
 - Lương = lương cơ bản + số năm kinh nghiệm * 1.000.000 (1 triệu).
- Trường hợp còn lại:
 - Lương = lương cơ bản.

○ **TeamLeader**

- Nếu **Developer** là **TeamLeader** sẽ có thu nhập tăng thêm.
 - Lương = lương developer + tỉ lệ thu nhập tăng thêm * lương developer

- **Tester**

- $Lương = lương\ cơ\ bản + tỉ\ lệ\ thu\ nhập\ tăng\ thêm * lương\ cơ\ bản$

V. Mô tả file input và file output

- File input có tên *ListOfEmployees.txt* chứa danh sách nhân viên với mỗi dòng ứng với thuộc tính của 01 nhân viên được cách nhau bằng dấu “,” theo định dạng:

- **Developer**

Số thứ tự, Mã nhân viên, Tên nhân viên, Tên nhóm, Số năm kinh nghiệm, Lương cơ bản

1,D01,Nguyen Dinh Minh Khoi,Run,1,5000000
2,D02,Pham Le Anh Khoa,Fly,3,6000000

- **TeamLeader**

Số thứ tự, Mã nhân viên, Tên nhân viên, Tên nhóm, Số năm kinh nghiệm, Ký tự L cho biết đây là TeamLeader, Tỉ lệ thu nhập tăng thêm, Lương cơ bản

6,D04,To Quoc Bao,Walk,3,L,0.3,10000000

- **Tester**

Số thứ tự, Mã nhân viên, Tên nhân viên, Tỉ lệ thu nhập tăng thêm, Loại nhân viên kiểm thử, Lương cơ bản

3,T01,Vu Bao Dang Khoa,0.2,AT,3000000
4,T02,Truong Pham Thao Mi,0,MT,2000000

- File output gồm có 5 file ứng với kết quả mẫu cho các yêu cầu:
 - *Req2.txt*: kết quả danh sách nhân viên thành thạo ngôn ngữ C++.
 - *Req3.txt*: kết quả danh sách nhân viên có lương > 4,700,000.
 - *Req4.txt*: kết quả nhân viên có lương cao nhất.
 - *Req5.txt*: kết quả TeamLeader có nhiều thành viên nhất.
 - *Req6.txt*: kết quả danh sách nhân viên sau khi được sắp xếp.
- Mỗi dòng trong từng file output ứng với 01 nhân viên với các thông tin được ghi ra file theo định dạng trong phương thức **toString()** của lớp **tương ứng**:
 - **Developer, TeamLeader**
empID_empName_baseSal_teamName_programmingLanguages_expYear
 - **Tester**
empID_empName_baseSal

D04_To Quoc Bao_10000000_Walk_[C,C++]_3
D05_Tran Bao Tin_10000000_Jump_[Java,C#]_3
D12_Tran Duc Minh_9000000_Walk_[C,Java,PHP]_3
T07_Do Thi Nhan_8000000
D02_Pham Le Anh Khoa_6000000_Fly_[C,Java]_3
D10_Dang Thanh Tu_9000000_Run_[Java]_2
D06_Le Van Nam_8000000_Jump_[PHP,Ruby]_1
T03_Pham Thi Nhu_6000000

Lưu ý:

- Sinh viên có thể tự thêm dữ liệu vào file input để thử nhiều trường hợp khác nhau nhưng lưu ý thêm dữ liệu phải đúng định dạng đã được nêu bên trên.
- Sinh viên nên đọc kỹ phương thức **main** để xác định hướng hiện thực các class và các phương thức theo thứ tự.
- Sinh viên có thể thêm một số thao tác vào phương thức **main** để kiểm các phương thức đã định nghĩa tuy nhiên đảm bảo bài làm của sinh viên **phải chạy được trên phương thức main ban đầu đã cung cấp sẵn**.
- Sinh viên có thể thêm phương thức mới để phục vụ bài làm tuy nhiên bài làm của sinh viên phải chạy được với file TestCM.java đã được cung cấp sẵn.
- **Tuyệt đối không chỉnh sửa tên của các phương thức đã có sẵn (tuân theo đúng sơ đồ lớp phía trên).**

VI. Yêu cầu

1. YÊU CẦU 1 (3 điểm)

Sinh viên hiện thực phương thức **public ArrayList<Employee> getEmployeeFromFile(String path, String path1)** để lấy danh sách nhân viên từ file *ListOfEmployees.txt* (*path*) và kết hợp đọc file *PLInfo.txt* (*path1*) để lấy thông tin ngôn ngữ lập trình của từng lập trình viên để khởi tạo các đối tượng **Employee** đưa vào danh sách *empList*.

Phương thức **main** gọi sẵn phương thức **printEmpList()**, nếu sinh viên hiện thực đúng phương thức **getEmployeeFromFile** thì khi chạy chương trình sẽ in ra màn hình danh sách nhân viên ứng với danh sách trong file đầu vào.

Lưu ý: Đây là phương thức sinh viên phải định nghĩa được mới bắt đầu tính điểm cho bài, nếu không đọc được file thành danh sách các đối tượng **Employee** thì các yêu cầu bên dưới không được tính điểm.

Tất cả các yêu cầu bên dưới đều thao tác trên danh sách *empList*.

2. YÊU CẦU 2 (2 điểm)

Hiện thực phương thức:

public ArrayList<Developer> getDeveloperByProgrammingLanguage(String pl)

trả về danh sách các Developer thành thạo ngôn ngữ lập trình truyền vào. Biết chuỗi truyền vào luôn chỉ chứa duy nhất 01 ngôn ngữ lập trình.

Sinh viên hiện thực phương thức trên sao cho khi thực hiện các lệnh gọi phương thức trong phương thức **main** sẽ cho kết quả ghi ra file “Req2.txt” như file output mẫu.

3. YÊU CẦU 3 (2 điểm)

Hiện thực phương thức

public ArrayList<Tester> getTestersHaveSalaryGreaterThan(double value)

trả về danh sách các Tester có lương lớn hơn giá trị **value** truyền vào. Giả sử **value** truyền vào là 5.000.000 thì danh sách trả về các Tester có lương > 5,000,000.

Sinh viên hiện thực phương thức trên sao cho khi thực hiện các lệnh gọi phương thức trong phương thức **main** sẽ cho kết quả ghi ra file “Req3.txt” như file output mẫu.

4. YÊU CẦU 4 (1 điểm)

Hiện thực phương thức

public Employee getEmployeeWithHigestSalary()

trả về Employee có lương cao nhất trong danh sách. Nếu lương bằng nhau thì trả về người cuối cùng có lương cao nhất trong danh sách.

Sinh viên hiện thực phương thức trên sao cho khi thực hiện các lệnh gọi phương thức trong phương thức **main** sẽ cho kết quả ghi ra file “Req4.txt” như file output mẫu.

5. YÊU CẦU 5 (1 điểm)

Hiện thực phương thức

public TeamLeader getLeaderWithMostEmployees()

trả về TeamLeader có số lượng thành viên trong nhóm đông nhất. Nếu số lập trình viên của nhóm bằng nhau thì trả về TeamLeader có nhiều năm kinh nghiệm hơn. (Testcase sẽ không có trường hợp nào ngoài điều kiện trên). Biết rằng mỗi nhóm chỉ có duy nhất 01 TeamLeader và mỗi lập trình viên chỉ thuộc duy nhất 01 nhóm.

Sinh viên hiện thực phương thức trên sao cho khi thực hiện các lệnh gọi phương thức trong phương thức **main** sẽ cho kết quả ghi ra file “Req5.txt” như file output mẫu.

6. YÊU CẦU 6 (1 điểm)

Hiện thực phương thức












public ArrayList<Employee> sorted()

Phương thức trên trả về danh sách nhân viên đã được sắp xếp theo tiền lương giảm dần. Nếu tiền lương bằng nhau thì sắp xếp theo chữ cái đầu tiên (theo thứ tự alphabet) của tên nhân viên (tên là từ cuối cùng trong họ tên). Việc sắp xếp thực hiện trên một danh sách Employee mới sao chép từ danh sách *empList* (Testcase sẽ không có trường hợp bằng tiền lương và trùng chữ cái đầu của tên)

Sinh viên hiện thực phương thức trên sao cho khi thực hiện các lệnh gọi phương thức trong phương thức **main** sẽ cho kết quả ghi ra file “Req6.txt” như file output mẫu.

VII. Lưu ý kiểm tra trước khi nộp bài

- Nếu sinh viên không thực hiện được yêu cầu nào thì để nguyên phương thức của yêu cầu đó, TUYỆT ĐỐI KHÔNG XÓA PHƯƠNG THỨC CỦA YÊU CẦU sẽ dẫn đến lỗi khi chạy phương thức **main**. Trước khi nộp phải kiểm tra chạy được với phương thức **main** được cho sẵn.
- Tất cả các file ReqX.txt ($X = \{2,3,4,5,6\}$) được ghi ra cùng cấp thư mục với thư mục chứa các file source code. Đối với sinh viên sử dụng IDE (Eclipse, Netbean, ...) phải đảm bảo file chạy được bằng command prompt, đảm bảo bài làm không nằm trong package, vị trí ghi file kết quả ReqX.txt phải nằm cùng thư mục với file code.
- File kết quả ghi đúng thư mục khi chạy chương trình sẽ có dạng như sau:

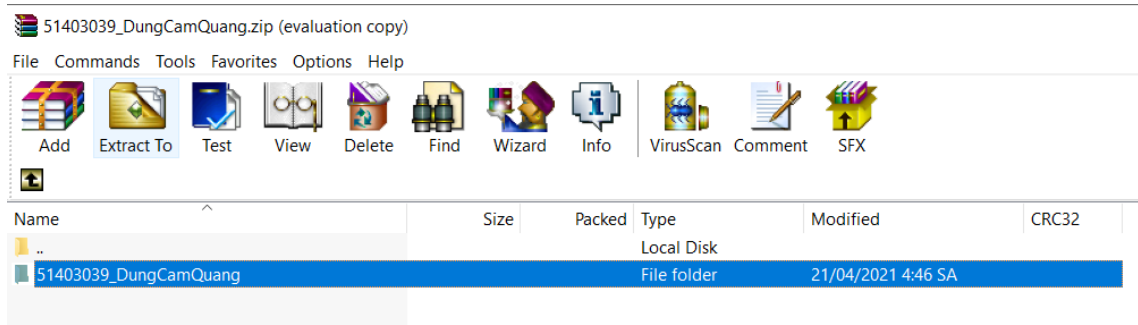
| | | | |
|---|---------------------|---------------|------|
|  input | 15/04/2021 11:00 SA | File folder | |
|  CompanyManagement.java | 04/04/2021 7:01 CH | JAVA File | 7 KB |
|  Developer.java | 30/03/2021 2:07 CH | JAVA File | 2 KB |
|  Employee.java | 30/03/2021 2:16 CH | JAVA File | 1 KB |
|  ICompanyManagement.java | 04/04/2021 7:01 CH | JAVA File | 1 KB |
|  Req2.txt | 21/04/2021 4:44 SA | Text Document | 1 KB |
|  Req3.txt | 21/04/2021 4:44 SA | Text Document | 1 KB |
|  Req4.txt | 21/04/2021 4:44 SA | Text Document | 1 KB |
|  Req5.txt | 21/04/2021 4:44 SA | Text Document | 1 KB |
|  Req6.txt | 21/04/2021 4:44 SA | Text Document | 1 KB |
|  TeamLeader.java | 20/04/2021 8:00 CH | JAVA File | 1 KB |
|  TestCM.java | 21/04/2021 4:44 SA | JAVA File | 2 KB |
|  Tester.java | 30/03/2021 10:45 SA | JAVA File | 1 KB |

VIII. Hướng dẫn nộp bài

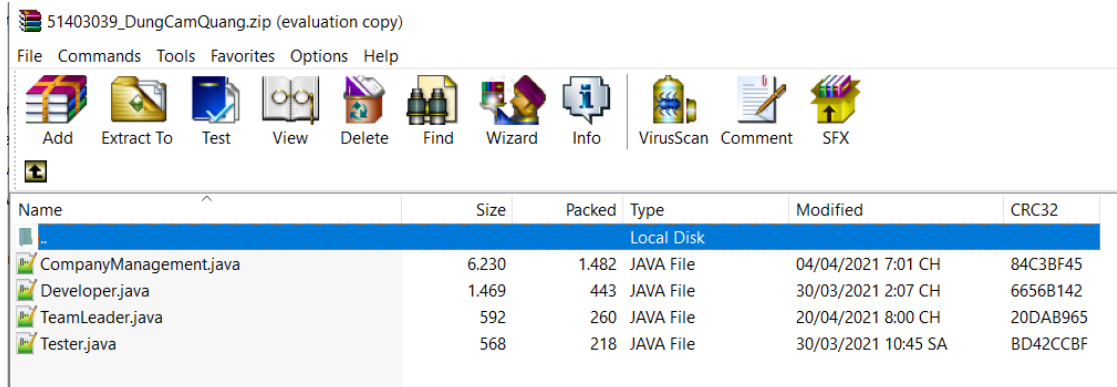
- Khi nộp bài sinh viên nộp lại file *Developer.java*, *TeamLeader.java*, *Tester.java* và file *CompanyManagement.java*, **không nộp kèm bất cứ file nào khác và tuyệt đối không được sửa tên 4 file này.**
- **Sinh viên đặt 4 file bài làm vào thư mục *MSSV_HoTen*** (HoTen viết liền, không dấu) và nén lại với định dạng **.zip** nộp theo sự hướng dẫn của giảng viên thực hành.
- Trường hợp làm sai yêu cầu nộp bài (đặt tên thư mục sai, không để bài làm vào thư mục khi nộp, nộp dư file, ...) thì bài làm của sinh viên sẽ bị **0 điểm**.
- File nộp đúng sẽ như sau:
 - o File nén nộp bài:

 51403039_DungCamQuang.zip 21/04/2021 4:47 SA WinRAR ZIP archive 4 KB

- o Bên trong file nén:



- o Bên trong thư mục:



IX. Đánh giá và quy định

- Bài làm sẽ được chấm tự động thông qua testcase (file input và output có định dạng như mẫu đã gửi kèm) do đó sinh viên tự chịu trách nhiệm nếu không thực hiện đúng theo Hướng dẫn nộp bài hoặc tự ý sửa tên các phương thức đã có sẵn dẫn đến bài làm không biên dịch được khi chấm.
- Testcase sử dụng để chấm bài là file *ListOfEmployees.txt* và *PLInfo.txt* khác với file sinh viên đã nhận, sinh viên chỉ được điểm mỗi YÊU CẦU khi chạy ra đúng hoàn toàn kết quả của yêu cầu đó.
- Nếu bài làm của sinh viên biên dịch bị lỗi thì **0 điểm cả bài**.
- **Tất cả code sẽ được kiểm tra đạo văn. Mọi hành vi sao chép code trên mạng, chép bài bạn hoặc cho bạn chép bài nếu bị phát hiện đều sẽ bị điểm 0 vào điểm Quá trình 2 hoặc cấm thi cuối kì.**
- Nếu bài làm của sinh viên có dấu hiệu sao chép trên mạng hoặc sao chép nhau, sinh viên sẽ được gọi lên phòng vấn code để chứng minh bài làm là của mình.
- **Hạn chót nộp bài: 23h00 ngày 20/05/2021.**

-- HẾT --