Học viên giải nén tệp JavaCore\_Progress.zip ra để thu được project JavaCore\_Progress.

Học viên vào Eclipse tạo project có tên JavaCore\_Progress\_Exercise và phải có JRE là jdk-20 hoặc mới hơn.

Học viên click chuột phải vào project vừa tạo chọn Import… sau đó đúp chuột chọn General sau đó đúp chuột chọn File System sau đó ở trên thanh From directory chọn Browse… sau đó chọn đến đường dẫn của project JavaCore\_Progress đã giải nén ở trên sau đó tick vào JavaCore\_Progress sau đó chọn Finish sau đó chọn Yes cho tới khi nào tạo thành công.

Học viên thực hiện các tiến trình yêu cầu bên dưới vào theo hướng dẫn của giảng viên chủ nhiệm.

Tiến trình 1:

Tạo lớp Employee - Nhân viên chứa các thông tin sau:

- Object fields:

+ int ID: *mã số nhân viên*.

+ String name: *họ và tên nhân viên*.

+ int age: *tuổi nhân viên*.

+ int baseSalary: *lương cơ bản*.

- Object constructors:

+ Employee(): phương thức khởi tạo mặc định.

+ Employee(int ID, String name, int age, int baseSalary): phương thức khởi tạo đầy đủ tham số dùng để khởi tạo 4 thuộc tính tương ứng của đối tượng Employee.

- Object methods:

+ double getSalary(): trả về *lương nhân viên* theo công thức sau:

* Nếu tuổi nhân viên từ 17 trở xuống thì công thức tính lương sẽ là: lương cơ bản + 0\*lương cơ bản.
* Nếu tuổi nhân viên thuộc [18, 25] thì công thức tính lương sẽ là: lương cơ bản + 0.03\*lương cơ bản.
* Nếu tuổi nhân viên thuộc [26, 35] thì công thức tính lương sẽ là: lương cơ bản + 0.06\*lương cơ bản.
* Nếu tuổi nhân viên từ 36 trở lên thì công thức tính lương sẽ là: lương cơ bản + 0.1\*lương cơ bản.

+ Employee copy(): trả về bản sao của đối tượng.

+ void printInformation(): in ra *thông tin nhân viên* theo định dạng sau:

* Employee [id=mã số nhân viên, name=tên nhân viên, age=tuổi nhân viên, baseSalary=lương cơ bản].

Yêu cầu:

Tạo ra một đối tượng Employee (s1) sau đó in ra thông tin nhân viên, in ra lương nhân viên.

Tạo một đối tượng Employee mới (s2) được lấy từ bản sao của đối tượng s1, thay đổi lại mã số nhân viên của s2 thành -1 và in ra thông tin của nhân viên s1 và s2 kiểm tra xem mã số nhân viên của 2 đối tượng s1 và s2 này có bị ảnh hưởng bởi nhau hay không.

Tiến trình 2:

Tạo lớp Developer - Lập trình viên (kế thừa Employee) chứa các thông tin sau:

- Object fields:

+ int expYear: *năm kinh nghiệm*.

+ String teamName: *tên nhóm*.

- Object constructors:

+ Developer(): phương thức khởi tạo mặc định.

+ Developer(int ID, String name, int age, int baseSalary, int expYear, String teamName): phương thức khởi tạo đầy đủ tham số dùng để khởi tạo 6 thuộc tính tương ứng của đối tượng Developer.

- Object methods:

+ double getSalary(): (ghi đè phương thức) trả về *lương lập tình viên* theo công thức sau:

* Nếu năm kinh nghiệm nhỏ hơn 4 thì công thức tính lương sẽ là: lương nhân viên + 0\*lương nhân viên.
* Nếu năm kinh nghiệm thuộc [4, 7] thì công thức tính lương sẽ là: lương nhân viên + 0.04\*lương nhân viên.
* Nếu năm kinh nghiệm thuộc [8, 10] thì công thức tính lương sẽ là: lương nhân viên + 0.08\*lương nhân viên.
* Nếu năm kinh nghiệm từ 11 trở lên thì công thức tính lương sẽ là: lương nhân viên + 0.12\*lương nhân viên.

+ Developer copy(): (ghi đè phương thức) trả về bản sao của đối tượng.

+ void printInformation(): (ghi đè phương thức) in ra *thông tin lập trình viên* theo định dạng sau:

* Developer[id=mã số nhân viên, name=tên nhân viên, age=tuổi nhân viên, baseSalary=lương cơ bản, expYear=năm kinh nghiệm, teamName=tên nhóm].

Tạo lớp TeamLeader - Trưởng nhóm (kế thừa Developer) chứa các thông tin sau:

- Object fields:

+ double bonusRate: *tỉ lệ thưởng thêm*.

- Object constructors:

+ TeamLeader(): phương thức khởi tạo mặc định.

+ TeamLeader(int ID, String name, int age, int baseSalary, int expYear, String teamName, double bonusRate): phương thức khởi tạo đầy đủ tham số dùng để khởi tạo 7 thuộc tính tương ứng của đối tượng TeamLeader.

- Object methods:

+ double getSalary(): (ghi đè phương thức) trả về *lương trưởng nhóm* theo công thức sau:

* Công thức tính lương sẽ là: lương lập trình viên + tỉ lệ thưởng thêm\*lương lập trình viên.

+ TeamLeader copy(): (ghi đè phương thức) trả về bản sao của đối tượng.

+ void printInformation(): (ghi đè phương thức) in ra *thông tin nhóm trưởng* theo định dạng sau:

* TeamLeader[id=mã số nhân viên, name=tên nhân viên, age=tuổi nhân viên, baseSalary=lương cơ bản, expYear=năm kinh nghiệm, teamName=tên nhóm, bonusRate=tỉ lệ thưởng thêm].

Tạo lớp Tester - Kiểm thử (kế thừa Employee) chứa các thông tin sau:

- Object fields:

+ String type: *loại kiểm thử*.

- Object constructors:

+ Tester(): phương thức khởi tạo mặc định.

+ Tester(int ID, String name, int age, int baseSalary, String type): phương thức khởi tạo đầy đủ tham số dùng để khởi tạo 5 thuộc tính tương ứng của đối tượng Tester.

- Object methods:

+ double getSalary(): (ghi đè phương thức) trả về *lương kiểm thử* theo công thức sau:

* Nếu loại kiểm thử là “administrative tester” thì công thức tính lương sẽ là: lương nhân viên + 0.25\*lương nhân viên.
* Các loại kiểm thử còn lại công thức tính lương sẽ là: lương nhân viên + 0.05\*lương nhân viên.

+ Tester copy(): (ghi đè phương thức) trả về bản sao của đối tượng.

+ void printInformation(): (ghi đè phương thức) in ra *thông tin kiểm thử* theo định dạng sau:

* Tester[id=mã số nhân viên, name=tên nhân viên, age=tuổi nhân viên, baseSalary=lương cơ bản, type=loại kiểm thử].

Yêu cầu:

Tạo ra từng đối tượng của từng lớp và test các phương thức chức năng của chúng.

Tạo ra một danh sách Employee, thêm vào ngẫu nhiên các thể hiện Developer, TeamLeader và Tester sau đó duyệt danh sách, tìm kiếm và in ra những đối tượng có lương nhân viên > 3500000.

Tiến trình 3:

Thực hiện các thay đổi sau:

\* Đối với lớp Employee:

- Thay lớp Employee thành lớp trừu tượng.

- Thêm các modifier private cho các thuộc tính và đồng thời thêm các cặp phương thức getter và setter có modifier public cho các thuộc tính tương ứng.

- Thay phương thức copy() thành phương thức trừu tượng và có modifier public, xóa đi phương thức printInformation(), thêm modifier public cho phương thức getSalary() và các phương thức khởi tạo.

\* Đối với lớp Developer:

- Thêm các modifier private cho các thuộc tính và đồng thời thêm các cặp phương thức getter và setter có modifier public cho các thuộc tính tương ứng.

- Xóa đi phương thức printInformation(), thay vào là phương thức toString() (ghi đè phương thức) trả về chuỗi thông tin lập trình viên, thêm modifier public cho phương thức getSalary() và các phương thức khởi tạo.

\* Đối với lớp TeamLeader:

- Thêm các modifier private cho các thuộc tính và đồng thời thêm các cặp phương thức getter và setter có modifier public cho các thuộc tính tương ứng.

- Xóa đi phương thức printInformation(), thay vào là phương thức toString() (ghi đè phương thức) trả về chuỗi thông tin trưởng nhóm, thêm modifier public cho phương thức getSalary() và các phương thức khởi tạo.

\* Đối với lớp Tester:

- Thêm các modifier private cho các thuộc tính và đồng thời thêm các cặp phương thức getter và setter có modifier public cho các thuộc tính tương ứng.

- Xóa đi phương thức printInformation(), thay vào là phương thức toString() (ghi đè phương thức) trả về chuỗi thông tin kiểm thử, thêm modifier public cho phương thức getSalary() và các phương thức khởi tạo.

Yêu cầu:

Tạo ra từng đối tượng của từng lớp và test các phương thức chức năng của chúng, ta thấy lớp Employee không thể dùng để tạo thể hiện của đối tượng tương ứng.

Tạo ra một danh sách Employee, thêm vào ngẫu nhiên các thể hiện Developer, TeamLeader và Tester sau đó duyệt danh sách, tìm kiếm và in ra những đối tượng có lương nhân viên > 3500000.

Tiến trình 4:

Tạo giao diện (interface) IDataManagement chứa các phương thức trừu tượng sau:

+ public abstract boolean loadDataFromFile(String pathFile).

+ public abstract boolean saveDataToCurrentFile().

+ public abstract boolean saveDataAs(String pathFile).

Tạo giao diện (interface) IBasicManagement chứa các phương thức trừu tượng sau:

+ public abstract boolean create(Employee e).

+ public abstract Employee getByID(int ID).

+ public abstract boolean update(Employee e).

+ public abstract Employee deleteByID(int ID).

+ public abstract void showAll().

Tạo lớp EmployeeManagement - Quản lý nhân viên (triển khai từ 2 interface IDataManagement và IBasicManagement) chứa các thông tin sau:

- Object fields:

+ private ArrayList<Employee> employeeList: *danh sách nhân viên*.

+ private String fileOpen: *đường dẫn đến file dữ liệu đã mở*.

- Object constructors:

+ public EmployeeManagement(): phương thức khởi tạo mặc định khởi tạo danh sách nhân viên.

- Object methods:

+ public abstract boolean loadDataFromFile(String pathFile): (ghi đè phương thức) đặt cho đường dẫn đến file dữ liệu đã mở giá trị của pathFile sau đó đọc dữ liệu từ file đầu vào pathFile trích xuất và xử lý từng dòng để thêm dữ liệu vào danh sách nhân viên, đã được hướng dẫn trên lớp.

+ public abstract boolean saveDataToCurrentFile(): (ghi đè phương thức) ghi lại dữ liệu vào đường dẫn đến file dữ liệu đã mở, định dạng các dòng dữ liệu được ghi vào tương tự như file dữ liệu lúc đọc vào.

+ public abstract boolean saveDataAs(String pathFile): (ghi đè phương thức) ghi lại dữ liệu vào pathFile, định dạng các dòng dữ liệu được ghi vào tương tự như file dữ liệu lúc đọc vào.

+ public abstract boolean create(Employee e): (ghi đè phương thức) thêm nhân viên e vào danh sách nhân viên, trả về true nếu thêm thành công, ngược lại là false.

+ public abstract Employee getByID(int ID): (ghi đè phương thức) trả về đối tượng nhân viên theo ID truyền vào, trả về null nếu không tìm thấy.

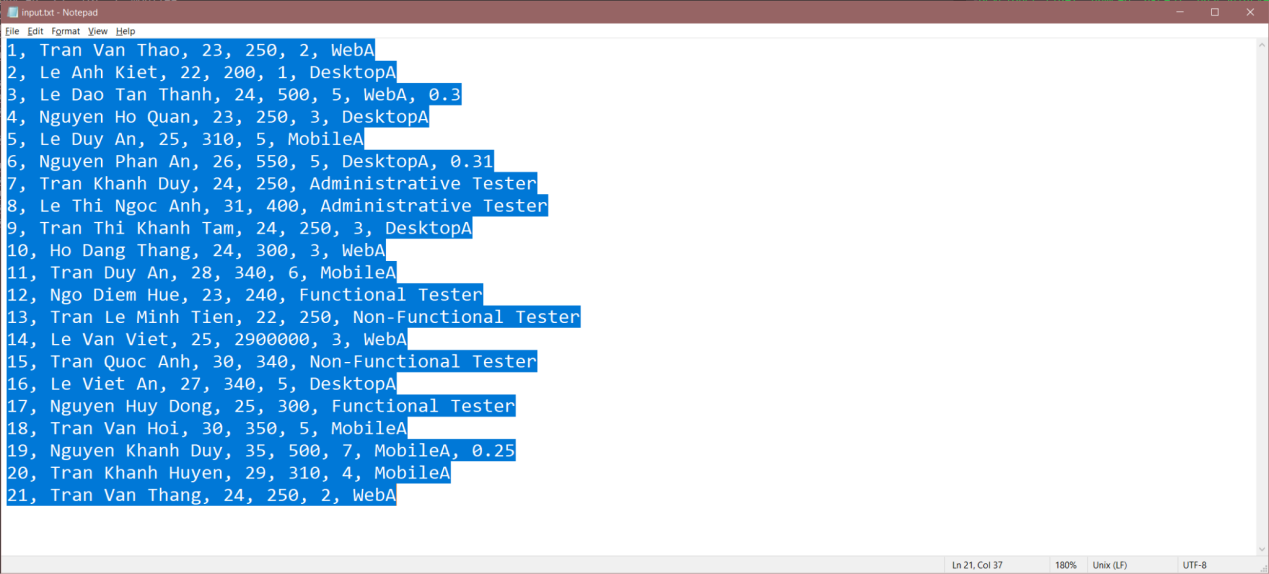
+ public abstract boolean update(Employee e): (ghi đè phương thức) cập nhật lại thông tin của nhân viên đã tìm thấy trong danh sách nhân viên theo mã số nhân viên của nhân viên e truyền vào, trả về true nếu thêm thành công, ngược lại là false.

+ public abstract Employee deleteByID(int ID): (ghi đè phương thức) trả về đối tượng sau khi xóa khỏi danh sách nhân viên theo ID truyền vào, trả về true nếu xóa thành công, ngược lại là false.

+ public abstract void showAll(): (ghi đè phương thức) duyệt và in ra thông tin các đối tượng nhân viên trong danh sách nhân viên từng dòng trên màn hình console.

Yêu cầu:

File dữ liệu đã cho input.txt:



Tạo ra đối tượng EmployeeManagement sau đó:

- Dùng phương thức loadDataFromFile() để đọc dữ liệu từ file input.txt đã cho.

- Thêm một vài đối tượng ngẫu nhiên bằng phương thức create()

- Thực hiện một thao tác tìm kiếm bằng phương thức getByID(), xóa bằng phương thức deleteByID(), sửa bằng phương thức update().

- Dùng phương thức saveDataAs() để lưu dữ liệu vào một file .txt khác (tên file tùy ý).

Tiến trình 5:

Tạo giao diện (interface) IAdvancedManagement chứa các phương thức trừu tượng sau:

+ public abstract List<Employee> sortByName().

+ public abstract List<Employee> sortByLastName().

+ public abstract List<Employee> sortByAge().

+ public abstract List<Employee> sortBySalary().

+ public abstract List<Employee> filterByLastName(String lastName).

+ public abstract List<Employee> filterByAge(int age).

+ public abstract List<Employee> filterByAge(int begin, int end).

+ public abstract List<Employee> filterBySalary(double salary).

+ public abstract List<Employee> filterBySalary(double begin, double end).

Cho lớp EmployeeManagement triển khai thêm từ interface IAdvancedManagement chức các thông tin sau:

- Object methods:

+ public abstract List<Employee> sortByName(): (ghi đè phương thức) tạo ra bản sao của danh sách nhân viên, sắp xếp lại bản sao theo thứ tự tăng dần của họ và tên nhân viên và trả về bản sao sau khi được sắp xếp.

+ public abstract List<Employee> sortByLastName(): (ghi đè phương thức) tạo ra bản sao của danh sách nhân viên, sắp xếp lại bản sao theo thứ tự tăng dần của tên nhân viên và trả về bản sao sau khi được sắp xếp.

+ public abstract List<Employee> sortByAge(): (ghi đè phương thức) tạo ra bản sao của danh sách nhân viên, sắp xếp lại bản sao theo thứ tự tăng dần của tuổi nhân viên và trả về bản sao sau khi được sắp xếp.

+ public abstract List<Employee> sortBySalary(): (ghi đè phương thức) tạo ra bản sao của danh sách nhân viên, sắp xếp lại bản sao theo thứ tự tăng dần của lương nhân viên và trả về bản sao sau khi được sắp xếp.

+ public abstract List<Employee> filterByLastName(String lastName): (ghi đè phương thức) lọc ra danh sách nhân viên có tên bằng với lastName truyền vào và trả về danh sách nhân viên đã lọc được.

+ public abstract List<Employee> filterByAge(int age): (ghi đè phương thức) lọc ra danh sách nhân viên có tuổi nhân viên bằng với age truyền vào và trả về danh sách nhân viên đã lọc được.

+ public abstract List<Employee> filterByAge(int begin, int end): (ghi đè phương thức) lọc ra danh sách nhân viên có tuổi nhân viên nằm trong đoạn [begin, end] truyền vào và trả về danh sách nhân viên đã lọc được.

+ public abstract List<Employee> filterBySalary(double salary): (ghi đè phương thức) lọc ra danh sách nhân viên có lương nhân viên bằng với salary truyền vào và trả về danh sách nhân viên đã lọc được.

+ public abstract List<Employee> filterBySalary(double begin, double end): (ghi đè phương thức) lọc ra danh sách nhân viên có lương nhân viên nằm trong đoạn [begin, end] truyền vào và trả về danh sách nhân viên đã lọc được.

Yêu cầu:

Thực hiện tiếp yêu cầu ở tiến trình 4, các phương thức đã được thêm vào hãy thực hiện ít nhất 1 thao tác với từng phương thức.