

## Assignment

### MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

MÃ MÔN: 504008

#### Đề bài: Quản lý hóa đơn, tìm quan hệ giữa khách hàng và sản phẩm

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

#### I. Giới thiệu bài toán

Một cửa hàng tiện lợi cần quản lý các hóa đơn của mình. Đồng thời sau mỗi hóa đơn thì khách hàng sẽ đánh giá sản phẩm bằng số sao, cửa hàng tiện lợi mong muốn tìm ra các quy luật từ mối quan hệ giữa khách hàng và sản phẩm để chuẩn bị chiến lược kinh doanh phù hợp.

Các chức năng mà sinh viên phải thực hiện là:

- Xây dựng cây nhị phân tìm kiếm cân bằng (cây AVL) để lưu trữ hóa đơn.
- Xây dựng đồ thị biểu diễn mối quan hệ của khách hàng và sản phẩm để tìm ra kết quả của các truy vấn.

#### 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 *data* gồm các file *receipt.txt* chứa danh sách các hóa đơn, *customer.txt* chứa danh sách khách hàng, *product.txt* chứa danh sách sản phẩm và *rating.txt* chứa đánh giá của sản phẩm từ khách hàng.
  - o Folder *output* gồm: 5 file *Req1.txt*, *Req2.txt*, *Req3.txt*, *Req4.txt*, *Req5.txt* chứa kết quả mẫu từ *main* cung cấp sẵn gọi đến các yêu cầu trong bài.
- File mã nguồn:
  - o *Main.java*: tạo đối tượng và gọi đến các phương thức sinh viên sẽ định nghĩa.
  - o *Receipt.java*, *Customer.java*, *Product.java*: lần lượt chứa lớp **Receipt**, **Customer**, **Product** được định nghĩa sẵn. **Sinh viên không chỉnh sửa các file này.**

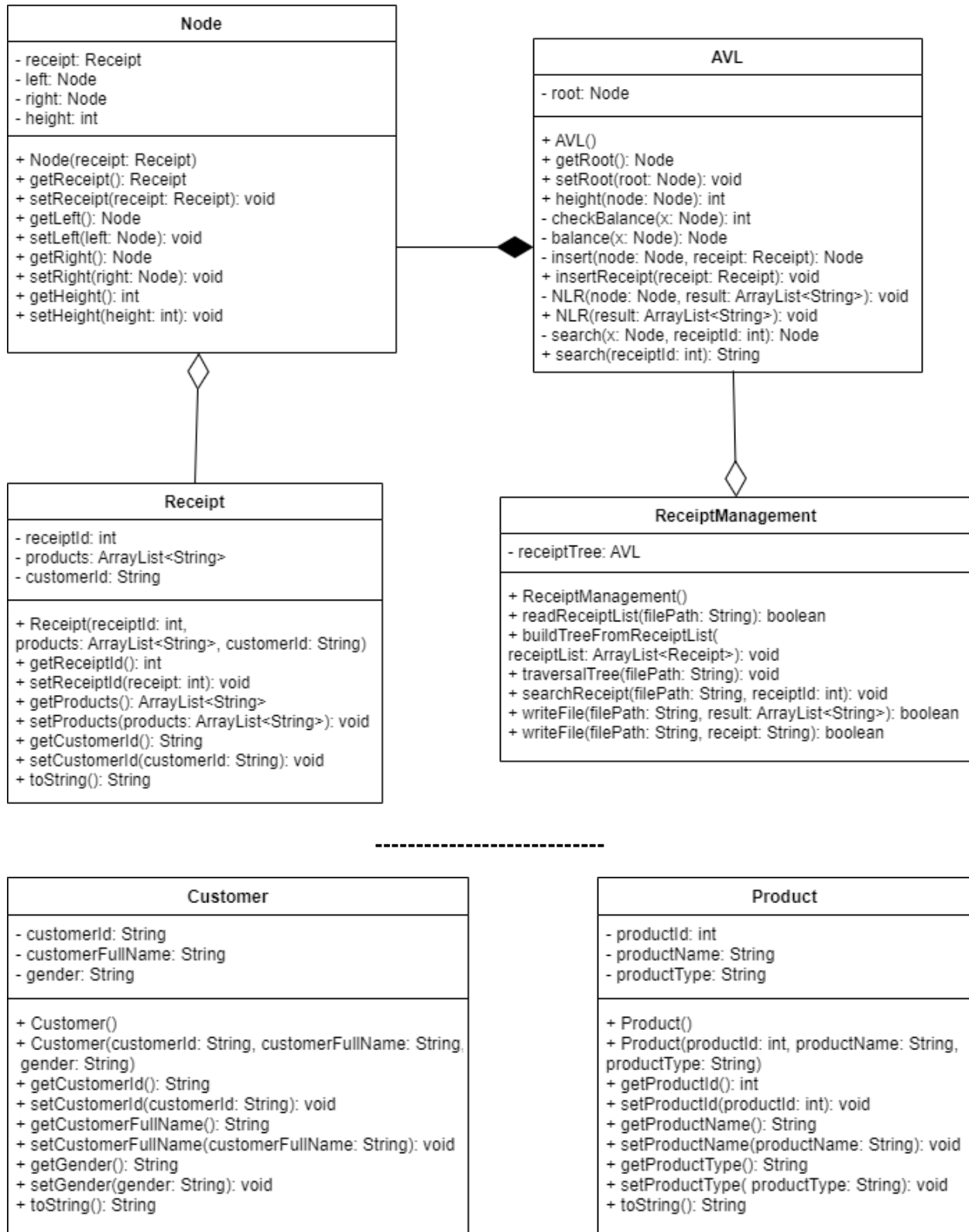
- *AVL.java*: chứa lớp **AVL** và các phương thức để xây dựng cây AVL. Sinh viên sẽ hiện thực các phương thức còn thiếu trong file này.
- *ReceiptManagement.java*: chứa lớp **ReceiptManagement** được định nghĩa sẵn thuộc tính *cây AVL* để chứa danh sách các hóa đơn (receipt). File này được định nghĩa sẵn các phương thức đọc file, ghi file và các phương thức để xây dựng cây, duyệt cây, tìm kiếm dựa trên việc gọi đến các phương thức trong lớp **AVL**. Sinh viên sẽ hiện thực các phương thức trong lớp **AVL** để các phương thức trong lớp này hoạt động được. **Sinh viên không chỉnh sửa file này.**
- *RatingQuery.java*: chứa lớp **RatingQuery** được định nghĩa sẵn các phương thức đọc file và ghi file, sinh viên sẽ hiện thực các yêu cầu liên quan vào file này.

### III. Trình tự thực hiện bài

- Sinh viên tải file tài nguyên được cung cấp sẵn và giải nén.
- Bài gồm có 05 yêu cầu. Các yêu cầu được ký hiệu:
  - Yêu cầu 1: YC1
  - Yêu cầu 2: YC2
  - Yêu cầu 3: YC3
  - Yêu cầu 4: YC4
  - Yêu cầu 5: YC5
- Ở YC1 và YC2, sinh viên hiện thực cây **AVL** bằng các tạo thêm file *Node.java* chứa lớp **Node** và hiện thực các phương thức còn thiếu trong file *AVL.java*.
- Nếu sinh viên hiện thực cây **AVL** đúng thì các phương thức trong lớp **ReceiptManagement** sẽ hoạt động.
- Ở YC3, YC4 và YC5, sinh viên hiện thực lớp **Rating** để biểu diễn đồ thị ở dạng danh sách cạnh, hiện thực các phương thức trong lớp **RatingQuery** để tìm ra các thông tin trong dữ liệu theo yêu cầu.
- Sau khi hiện thực các phương thức trong lớp **RatingQuery**, sinh viên biên dịch và chạy với phương thức **main** trong file *Main.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 Main.java được cung cấp sẵn.**
- Đường link Google Drive gửi đề này cho sinh viên sẽ chứa kèm một file *version.txt*, sinh viên nên thường xuyên vào xem file này, nếu thấy có nội dung mới thì nên đọc kỹ mô tả và tải lại đề mới nhất. File này được dùng để thông báo nội dung chỉnh sửa và ngày chỉnh sửa trong trường hợp đề bị sai hoặc lỗi.

#### IV. Mô tả lớp **Product**, **Customer**, **Receipt**, **ReceiptManagement**, **Node**, **AVL**, **Rating**, **RatingQuery**

- Giải thích một số thuộc tính và phương thức như sau:
  - Lớp **Receipt** (sinh viên chỉ đọc hiểu và không chỉnh sửa lớp này):
    - *receiptId*: mã hóa đơn
    - *products*: danh sách các mã sản phẩm
    - *customerId*: mã khách hàng
  - Lớp **Node**:
    - *receipt*: hóa đơn
    - *left*, *right*: liên kết trái và liên kết phải
    - *height*: độ cao của nút
  - Lớp **AVL**:
    - *root*: nút gốc
    - **checkBalance(x: Node)**: kiểm tra balance factor của một nút.
    - **balance(x: Node)**: cân bằng cây tại một nút.
    - **insert(node: Node, receipt: Receipt)**: đệ quy thêm một nút vào cây.
    - **NLR(node: Node, result: ArrayList<String>)**: duyệt NLR và lưu kết quả vào danh sách **result** truyền vào.
    - **Search(x: Node, receiptId: int)**: đệ quy tìm kiếm nút có đang lưu hóa đơn có mã là **receiptId** truyền vào.
  - Lớp **ReceiptManagement** (sinh viên chỉ đọc hiểu và không chỉnh sửa lớp này):
    - *receiptTree*: cây AVL chứa hóa đơn
    - **readReceiptList(filePath: String)**: đọc danh sách hóa đơn từ file *receipt.txt* và xây dựng cây từ danh sách này.
    - **traversalTree(filePath: String)**: duyệt cây để ghi file.
    - **searchReceipt(filePath: String, receiptId: int)**: tìm kiếm hóa đơn theo mã hóa đơn và ghi file.



- Giải thích một số thuộc tính và phương thức như sau:
  - Lớp **Customer** (sinh viên chỉ đọc hiểu và không chỉnh sửa lớp này):
    - *customerId*: mã khách hàng
    - *customerFullName*: tên khách hàng
    - *gender*: giới tính (“M” là nam và “F” là nữ)
  - Lớp **Product** (sinh viên chỉ đọc hiểu và không chỉnh sửa lớp này):
    - *productId*: mã sản phẩm
    - *productName*: tên sản phẩm
    - *productType*: loại sản phẩm

RatingQuery
- edges: ArrayList<Rating> - productList: ArrayList<Product> - customerList: ArrayList<Customer>
+ RatingQuery(edges: ArrayList<Rating>, productList: ArrayList<Product>, customerList: ArrayList<Customer>) + printGraph(): void + readCustomerFile(filePath: String): boolean + readProductFile(filePath: String): boolean + addEdge(u: Customer, v: Product, w: int): void + buildGraph(filePath: String): boolean + query1(customerId: String): ArrayList<Product> + query2(productId: int): Integer + query3(): ArrayList<Product> + writeFile(filePath: String, list: ArrayList<E>): boolean + writeFile(filePath: String, data: <E>): boolean

- Giải thích một số thuộc tính và phương thức như sau:
  - Lớp **RatingQuery**:
    - *edges*: danh sách cạnh biểu diễn đồ thị
    - *productList*: danh sách sản phẩm
    - *customerList*: danh sách khách hàng
    - **Phương thức khởi tạo**: gọi đến các phương thức đọc file và xây dựng đồ thị.
    - **printGraph()**: in danh sách cạnh.
    - **addEdge(u: Customer, v: Product, w: int)**: thêm một cạnh vào danh sách cạnh với đỉnh *u* là một khách hàng, đỉnh *v* là một sản phẩm và trọng số *w* là số sao mà khách hàng đánh giá cho sản phẩm.
    - **buildGraph(filePath: String)**: xây dựng đồ thị từ file *rating.txt*, phương thức này đã được định nghĩa sẵn. Sinh viên chỉ cần định nghĩa đúng phương thức **addEdge** thì phương thức này sẽ hoạt động.

- **query1, query2, query3:** là các phương thức ứng với Yêu cầu 3, 4 và 5.
- **Hai phương thức writeFile:** phương thức dùng để ghi file và được dùng trong phương thức **main** để ghi kết quả các Yêu cầu 3, 4 và 5.

## V. Mô tả file input và file output

- Trong thư mục **data** gồm 04 file input.
- File input có tên *receipt.txt* chứa danh sách các hóa đơn, mỗi dòng ứng với thuộc tính của một hóa đơn được cách nhau bằng dấu “,” theo định dạng:

Mã hóa đơn, Danh sách mã sản phẩm (cách nhau bằng dấu “\_”), Mã khách hàng

```
1    1,[1_2],Q102
2    2,[1_4],Q902
3    3,[1_7_10],Q701
4    4,[2_3_4_9],Q702
```

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

Mã khách hàng, Tên khách hàng, Giới tính (“M” là nam và “F” là nữ)

```
1    Q101,Vu Thi Hanh,F
2    Q102,Nguyen Quang Duy,M
3    Q1101,Tran Tuan Kiet,M
4    Q801,Pham Thi Nhu,F
5    Q802,Nguyen Duy An,M
```

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

Mã sản phẩm, Tên sản phẩm, Loại sản phẩm

```
1    1,Vinamilk,Milk
2    2,Chicken,Meat
3    3,Pork,Meat
4    4,Beef,Meat
5    5,Coca,Beverage
```

Mã sản phẩm sẽ luôn bắt đầu từ 1 và được đánh số liên tiếp nhau theo thứ tự tăng dần.

- File input có tên *rating.txt* chứa danh sách các đánh giá của khách hàng cho sản phẩm với mỗi dòng ứng với mỗi sản phẩm được khách hàng đánh giá, dữ liệu được cách nhau bằng dấu “,” theo định dạng:

Mã khách hàng, Mã sản phẩm, Số sao đánh giá

1	Q102, 1, 3
2	Q102, 2, 4
3	Q902, 1, 5
4	Q902, 4, 1
5	Q701, 1, 2

Ví dụ: dòng dữ liệu 1 là khách hàng mã Q102 đánh giá sản phẩm mã 1 với số sao là 3.

- Thư mục output gồm có 5 file ứng với kết quả mẫu cho các yêu cầu:
  - o *Req1.txt*: kết quả duyệt cây AVL chứa hóa đơn theo thứ tự cách duyệt NLR.
  - o *Req2.txt*: kết quả tìm kiếm mã hóa đơn là 3 trên cây AVL đã xây dựng.
  - o *Req3.txt*: kết quả các sản phẩm được khách hàng mã Q601 đánh giá trên 3 sao.
  - o *Req4.txt*: kết quả số lượng người dùng là nam đánh giá sản phẩm mã 1 trên 3 sao.
  - o *Req5.txt*: kết quả các sản phẩm có hơn 50% số lượng khách hàng là nữ đánh giá.

### **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 lớp 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 *Main.java* đã được cung cấp sẵn.
- **Tuyệt đối không chỉnh sửa tên lớp hoặc tên của các phương thức đã có sẵn (tuân theo đúng sơ đồ lớp phía trên).**
- **Sinh viên không thực hiện chỉnh sửa trên những file không yêu cầu nộp lại.**

## **VI. Hướng dẫn chạy phương thức main**

- Phương thức **main** được cung cấp sẵn trong file *Main.java* được tạo sẵn các đối tượng và gọi để kiểm thử các yêu cầu.



- Để kiểm thử các yêu cầu, sinh viên phải định nghĩa các lớp đủ và đúng theo các yêu cầu, sau khi biên dịch thành công, sinh viên gọi theo cú pháp để in ra kết quả của yêu cầu tương ứng:

*java Main X Y*

- o YC1: X = 1, Y = 1
- o YC2: X = 1, Y = 2
- o YC3: X = 2, Y = 1
- o YC4: X = 2, Y = 2
- o YC5: X = 2, Y = 3

Ví dụ, sinh viên muốn gọi để kiểm thử YC1, sinh viên nhập *java Main 1 1*, kết quả sẽ được ghi ra file *Req1.txt*.

- Sau khi đã kiểm thử thành công với **main** cung cấp sẵn, sinh viên có thể thay đổi tham số để kiểm tra với các trường hợp khác.

## VII. Yêu cầu

Sinh viên không được phép sử dụng thư viện bên ngoài.

Sinh viên nên thực hiện bài tập lớn trên Java 8.

### 1. YÊU CẦU 1 (3 điểm)

Sinh viên hiện thực lớp **Node** theo sơ đồ lớp bên trên và hiện thực các phương thức trong lớp **AVL** để tạo cây AVL từ nút (Node) đã định nghĩa.

Sinh viên hiện thực phương thức **private Node insert(Node node, Receipt receipt)** để thực hiện đệ quy thêm một hóa đơn mới vào cây AVL. Lưu ý thêm sinh viên phải cân bằng cây mỗi khi thêm một nút mới vào. (Sinh viên tự định nghĩa thêm phương thức **rotateLeft**, **rotateRight** và hoàn thiện phương thức **balance**)

Phương thức duyệt NLR và đọc file *receipt.txt* lấy danh sách hóa đơn để thêm vào cây đã được hiện thực và gọi sẵn trong lớp **ReceiptManagement**.

Sau khi định nghĩa thành công phương thức **insert**, sinh viên kiểm thử bằng phương thức **main** đã cung cấp bằng cách gọi câu lệnh *java Main 1 1* thì chương trình sẽ cho ra file *Req1.txt*. Sinh viên xem kết quả file này và file *Req1.txt* trong folder output mẫu được cung cấp sẵn để so sánh kết quả.

**Lưu ý:** Đây là phương thức sinh viên phải định nghĩa được thì Yêu cầu 2 mới được tính điểm.

### 2. YÊU CẦU 2 (1 điểm)

Trong lớp AVL, sinh viên hiện thực phương thức:



**private Node search(Node x, int receiptId)**

để thực hiện đệ quy tìm một nút có theo mã hóa đơn **receiptId** truyền vào.

Lớp **ReceiptManagement** đã hiện thực sẵn lời gọi đến phương thức **search**. Sinh viên kiểm thử bằng phương thức **main** đã cung cấp bằng cách gọi câu lệnh *java Main 1 2* thì chương trình sẽ cho ra file *Req2.txt*. Sinh viên xem kết quả file này và file *Req2.txt* trong folder output mẫu được cung cấp sẵn để so sánh kết quả.

**3. YÊU CẦU 3 (3 điểm)**

Để thực hiện Yêu cầu 3, 4, 5 sinh viên phải xây dựng đồ thị hai phía, vô hướng, có trọng số dựa vào file *rating.txt*. Đồ thị sẽ chứa các đỉnh là khách hàng và sản phẩm, cạnh có trọng số đại diện cho số sao mà khách hàng đánh giá sản phẩm.

Ví dụ:

Với các dòng dữ liệu:

Q102,1,3

Q102,2,4

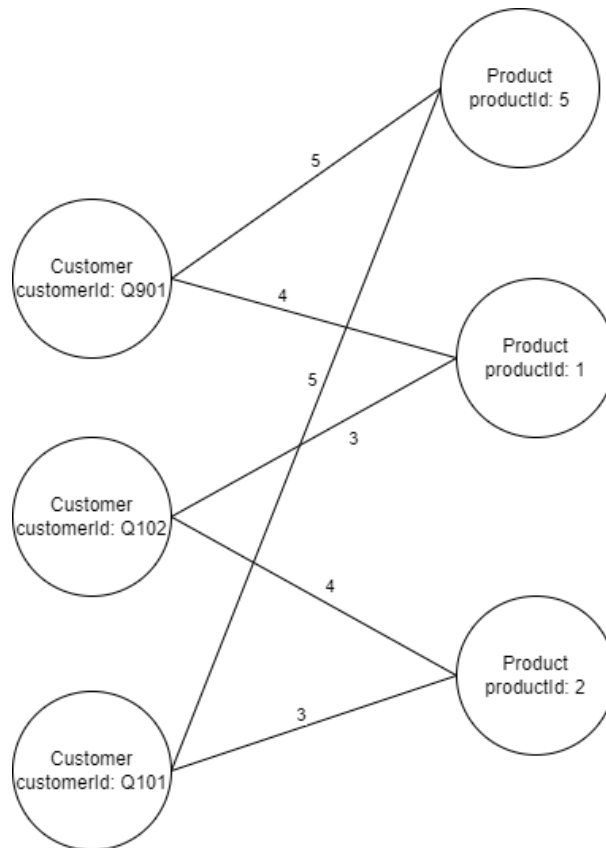
Q101,2,3

Q901,1,4

Q101,5,5

Q901,5,5

Ta sẽ xây dựng được đồ thị như sau:



Để lưu trữ đồ thị dạng trên, sinh viên sẽ sử dụng danh sách cạnh (edge list) với mỗi đối tượng cạnh có đỉnh  $u$  là Customer, đỉnh  $v$  là Product và trọng số  $w$  là số sao.

Sinh viên xây dựng đồ thị theo từng bước sau:

1. Tạo file **Rating.java** để hiện thực lớp **Rating**. Lớp **Rating** sẽ có 03 thuộc tính là Customer, Product và số sao. Mỗi đối tượng Rating sẽ là thể hiện của hai đỉnh được nối bởi một cạnh trong đồ thị.
2. Trong file *RatingQuery.java* sinh viên hiện thực phương thức **public void addEdge(Customer u, Product v, int w)** để thêm cạnh vào danh sách **edges** có sẵn.
3. Trong lớp **RatingQuery**, phương thức **public boolean buildGraph(String filePath)** đã được định nghĩa sẵn để đọc file *rating.txt*. Nếu sinh viên định nghĩa đúng phương thức **addEdge** thì phương thức **buildGraph** này sẽ hoạt động để tạo đồ thị bằng danh sách cạnh.
4. Sinh viên có danh sách cạnh **edges** để thực hiện các yêu cầu.

Sinh viên hiện thực phương thức:

**public ArrayList<Product> query1(String customerId)**

trả về danh sách những sản phẩm được khách hàng có mã là **customerId** đánh giá trên 3 sao.

Sinh viên hiện thực phương thức trên, biên dịch và chạy lệnh *java Main 2 1* để ghi ra kết quả file *Req3.txt*. Sinh viên xem kết quả file này và file *Req3.txt* trong folder output mẫu được cung cấp sẵn để so sánh kết quả.

#### 4. YÊU CẦU 4 (2 điểm)

Hiện thực phương thức

**public Integer query2(int productId)**

trả về số lượng người dùng là nam đánh giá sản phẩm có mã là **productId** trên 3 sao.

Sinh viên hiện thực phương thức trên, biên dịch và chạy lệnh *java Main 2 2* để ghi ra kết quả file *Req4.txt*. Sinh viên xem kết quả file này và file *Req4.txt* trong folder output mẫu được cung cấp sẵn để so sánh kết quả.

#### 5. YÊU CẦU 5 (1 điểm)

Hiện thực phương thức

























**public ArrayList<Product> query3()**

trả về danh sách những sản phẩm có từ 50% trở lên khách hàng đánh giá là nữ.

Sinh viên hiện thực phương thức trên, biên dịch và chạy lệnh *java Main 2 3* để ghi ra kết quả file *Req5.txt*. Sinh viên xem kết quả file này và file *Req5.txt* trong folder output mẫu được cung cấp sẵn để so sánh kết quả.

### VIII. 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 ( $X = \{1,2,3,4,5\}$ ) 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:

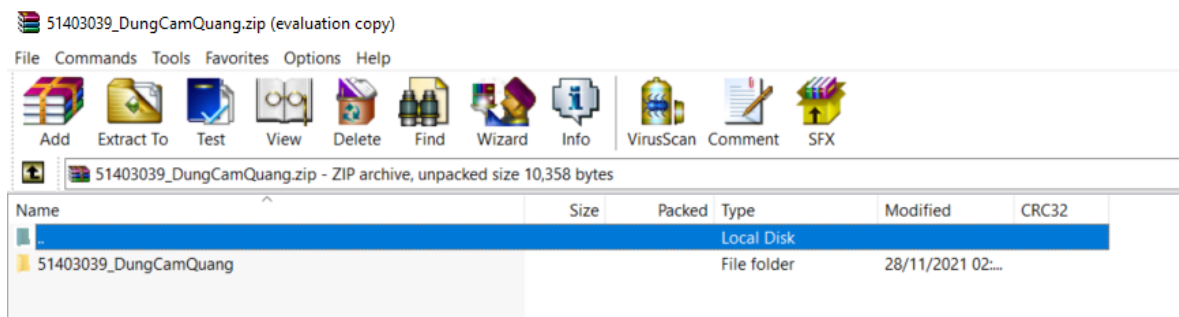
 data	22/11/2021 22:49	File folder	
 AVL.class	23/11/2021 03:22	CLASS File	3 KB
 AVL.java	22/11/2021 03:35	JAVA File	3 KB
 Customer.class	23/11/2021 03:22	CLASS File	2 KB
 Customer.java	23/11/2021 02:31	JAVA File	2 KB
 Main.class	23/11/2021 03:22	CLASS File	2 KB
 Main.java	23/11/2021 03:22	JAVA File	2 KB
 Node.class	23/11/2021 03:22	CLASS File	2 KB
 Node.java	21/11/2021 02:44	JAVA File	1 KB
 Product.class	23/11/2021 03:22	CLASS File	2 KB
 Product.java	23/11/2021 02:46	JAVA File	2 KB
 Rating.class	23/11/2021 03:22	CLASS File	2 KB
 Rating.java	23/11/2021 02:24	JAVA File	1 KB
 RatingQuery.class	23/11/2021 03:22	CLASS File	6 KB
 RatingQuery.java	23/11/2021 03:08	JAVA File	6 KB
 Receipt.class	23/11/2021 03:22	CLASS File	2 KB
 Receipt.java	22/11/2021 03:28	JAVA File	2 KB
 ReceiptManagement.class	23/11/2021 03:22	CLASS File	4 KB
 ReceiptManagement.java	22/11/2021 03:35	JAVA File	3 KB
 Req1.txt	23/11/2021 03:22	Text Document	2 KB
 Req2.txt	23/11/2021 03:22	Text Document	1 KB
 Req3.txt	23/11/2021 03:22	Text Document	1 KB
 Req4.txt	23/11/2021 03:22	Text Document	1 KB
 Req5.txt	23/11/2021 03:23	Text Document	1 KB

## IX. Hướng dẫn nộp bài

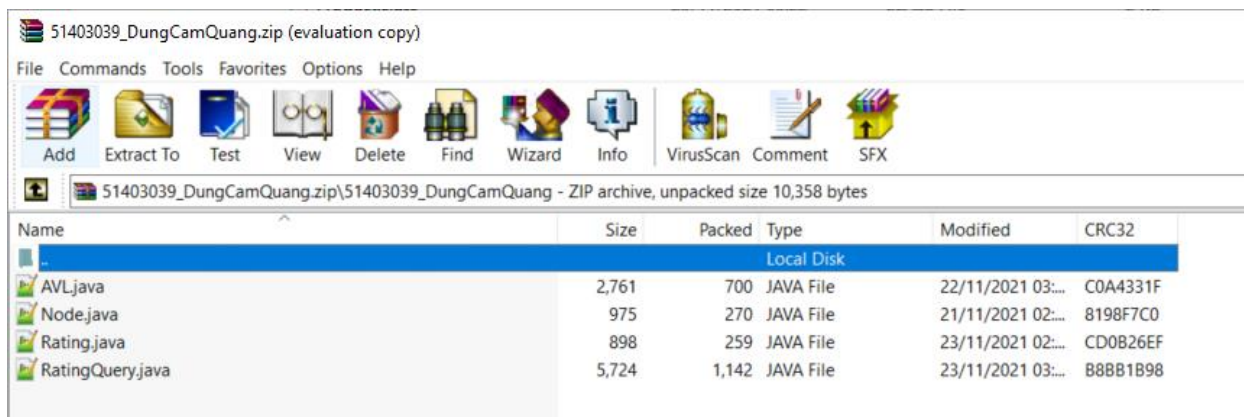
- Khi nộp bài sinh viên nộp lại file *Node.java*, *AVL.java*, *Rating.java* và file *RatingQuery.java*, **không nộp kèm bất cứ file nào khác và tuyệt đối không được sửa tên 4 file này.**
- **Sinh viên đặt 4 file bài làm vào thư mục *MSSV\_HoTen*** (HoTen viết liền, không dấu) và nén lại với định dạng **.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 28/11/2021 02:47 WinRAR ZIP archive 4 KB

- o Bên trong file nén:



- o Bên trong thư mục:



## X. 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 các file đầu vào có cùng định dạng nhưng khác nội dung 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 phỏng vấn code để chứng minh bài làm là của mình.
- **Hạn chót nộp bài: 23h00 ngày 18/12/2021.**

-- HẾT --