**MỤC LỤC**

[**Lời cảm ơn** 2](#_Toc60089439)

[**CHƯƠNG 1: YÊU CẦU BÀI TOÁN** 3](#_Toc60089440)

[**CHƯƠNG 2: GIẢI QUYẾT BÀI TOÁN** 4](#_Toc60089441)

[**2.1 Định hướng giải pháp** 4](#_Toc60089442)

[**2.2 Thiết kế chi tiết các lớp trong chương trình** 4](#_Toc60089443)

[2.2.1 Biểu đồ lớp 4](#_Toc60089444)

[2.2.2 Chi tiết các lớp và các phương thức quan trọng 4](#_Toc60089445)

[**CHƯƠNG 3: KẾT QUẢ ĐẠT ĐƯỢC** 12](#_Toc60089446)

# **Lời cảm ơn**

Lời đầu tiên, em xin trân trọng cảm ơn và bày tỏ lòng biết ơn sâu sắc tới cô **Nguyễn Thị Thu Hương** – Giảng viên Bộ môn Khoa học máy tính, Viện Công nghệ thông tin & Truyền thông, giảng viên hướng dẫn Project I đã nhiệt tình hướng dẫn, chỉ bảo.

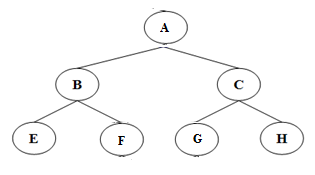
Em xin chân thành cảm ơn các thầy cô Viện Công nghệ thông tin & Truyền  
thông đã cung cấp tài liệu và đưa ra những ý kiến đóng góp quý báu.

Cuối cùng, em xin dành lời cảm ơn chân thành đến bạn bè đã động viên, giúp đỡ em hoàn thành tốt đề tài của mình.  
 Xin chân thành cảm ơn!

Hà Nội, tháng 12 năm 2020 Sinh viên  
 Nguyễn Đức Tuân

# **CHƯƠNG 1: YÊU CẦU BÀI TOÁN**

Một công ty bán hàng đa cấp tổ chức các nhà phân phối theo mô hình nhị phân:



Mô hình này cho phép mỗi nhà phân phối được và chỉ được tuyển mộ thêm hai nhà phân phối thuộc tầng 1 (**thế hệ thứ nhất**) và bắt buộc hai nhánh của mình phải luôn phát triển đồng đều (số sản phẩm bán được của nhánh ít phải >= 90% số sản phẩm bán được của nhánh nhiều). Nếu không thực hiện được điều này thì nhà phân phối sẽ chỉ được chi trả hoa hồng theo nhánh yếu hơn.

Hãy viết chương trình cho phép thực hiện chức năng sau:

* Nhập thông tin về nhà phân phối mới, bao gồm: họ tên, mã nhà phân phối cấp trên. Nếu nhà phân phối cấp trên đã đủ 2 nhà phân phối ở tầng 1, nhà phân phối mới sẽ được thêm vào cấp dưới tiếp theo. Giả sử chỉ lập 1 cây phân phối, nhà cung cấp được nhập đầu tiên là nhà cung cấp ở cấp cao nhất. Không giới hạn số tầng của cây. Chú ý thông tin các nhà cung cấp được lưu vào tệp văn bản, hoặc tệp CSDL.
* Xóa thông tin về nhà phân phối. Nhà phân phối cấp ngay dưới có mức hoa hồng cao hơn trong đợt phân phối gần nhất sẽ được chuyển lên cấp trên. Nếu hai nhà phân phối cấp dưới lượng hoa hồng ngang nhau thì chọn người có số lượng hàng bán nhiều hơn, nếu tất cả các tiêu chí của các nhà phân phối cấp dưới hoàn toàn giống nhau thì chọn ngẫu nhiên.
* Nhập thông tin về số sản phẩm bán được trong từng đợt, bao gồm (mã NCC, số luợng SP). Tính hoa hồng cần trả cho tất cả các nhà phân phối. Biết tỷ lệ hoa hồng cho nhà phân phối bán trực tiếp là 10%, mỗi mức cấp trên tỷ lệ là 10% của mức cấp ngay dưới. Thông tin về hoa hồng được lưu vào một tệp văn bản hoặc tệp CSDL, bao gồm (Mã NCC, số SP, số tiền hoa hồng).

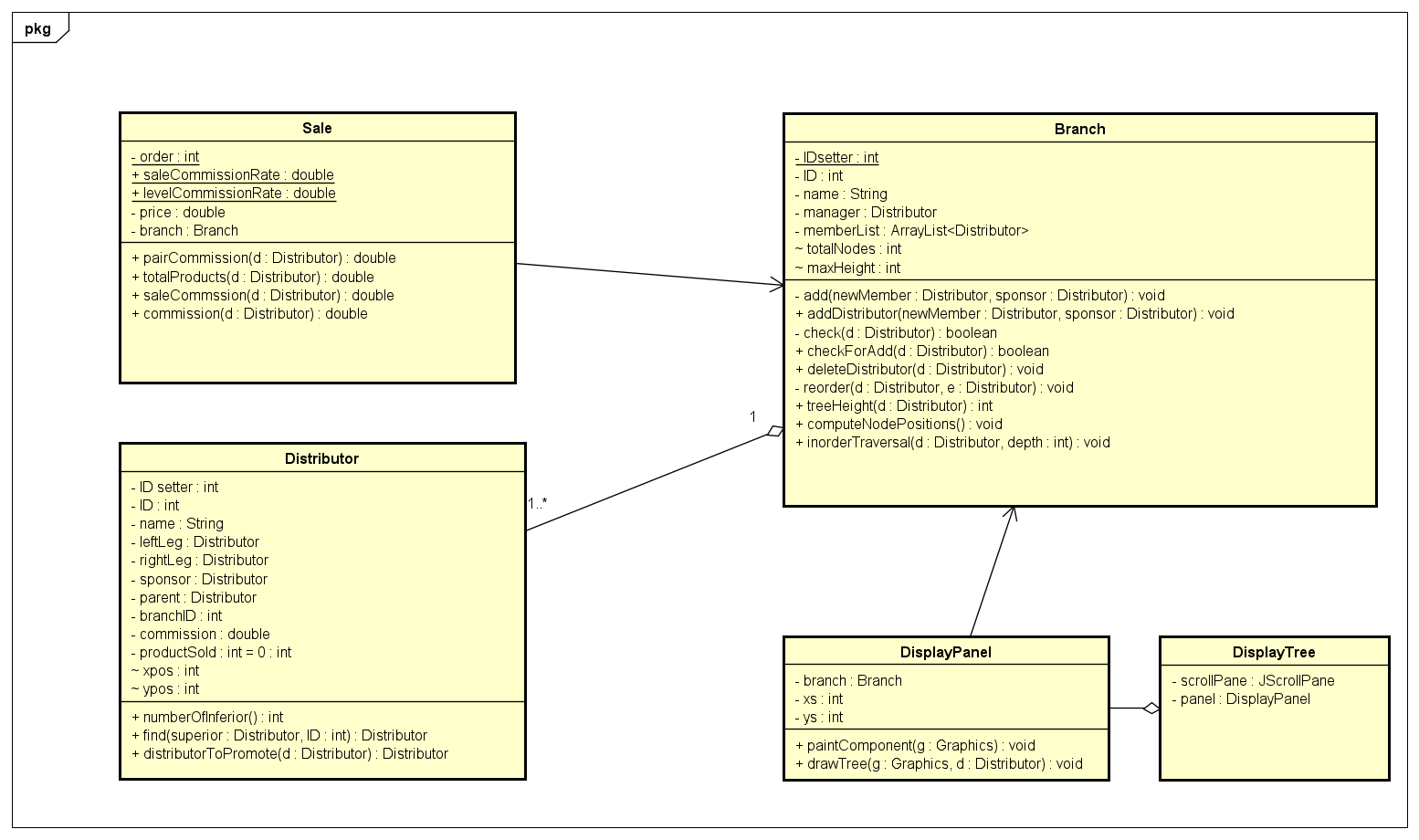
# **CHƯƠNG 2: GIẢI QUYẾT BÀI TOÁN**

## **2.1 Định hướng giải pháp**

* Ngôn ngữ lập trình sử dụng: Java
* Sử dụng phương pháp lập trình hướng đối tượng
* Sử dụng cấu trúc dữ liệu cây nhị phân và các thuật toán liên quan
* Kết nối với SQL server để lưu dữ liệu

## **2.2 Thiết kế chi tiết các lớp trong chương trình**

### 2.2.1 Biểu đồ lớp



### 2.2.2 Chi tiết các lớp và các phương thức quan trọng

1. Class **Distributor:** lưu trữ thông tin của các nhà phân phối

* Một số thuộc tính:

- ID, name: tên và ID của nhà cung cấp

- leftLeg, rightLeg: 2 nhà cung cấp ngay dưới

- sponsor: nhà cung cấp tài trợ cho nhà cung cấp này

- parent : nhà cung cấp ở vị trí node cha của nhà cung cấp hiện tại trong cây nhị phân (parent và sponsor của 1 nhà cung cấp có thể khác nhau, trong trường hợp thêm nhà cung cấp mới mà sponsor đã đủ 2 chân trái và phải)

- commission: tiền hoa hồng trong đợt phân phối gần nhất

- productsSold: số sản phẩm đã bán được trong đợt phân phối gần nhất

* Một số phương thức quan trọng:

**numberOfInferiors()**: trả về tổng số nhà phân phối con của nhà phân phối đang xét

public int numberOfInferiors() {

int count = 0;

if(leftLeg != null) count += 1 + leftLeg.numberOfInferiors();

if(rightLeg != null) count += 1 + rightLeg.numberOfInferiors();

return count;

}

**distributorToPromote(Distributor d)**: tìm nhà phân phối để thay thế nhà phân phối d, trong trường hợp xóa d khỏi cây nhị phân

public static Distributor distributorToPromote(Distributor d) {

Distributor distributorToPromote = null;

if(d.isFullOfLeg()) {

if(d.getRightLeg().getCommission() > d.getLeftLeg().getCommission()){

distributorToPromote = d.getRightLeg();

}

else if(d.getRightLeg().getCommission()

== d.getLeftLeg().getCommission()){

if(d.getRightLeg().productsSold > d.getLeftLeg().productsSold) {

distributorToPromote = d.getRightLeg();

} else

distributorToPromote = d.getLeftLeg();

}

else distributorToPromote = d.getLeftLeg();

} else {

if(d.getLeftLeg() == null && d.getRightLeg() != null)

distributorToPromote = d.getRightLeg();

if(d.getRightLeg() == null && d.getLeftLeg() != null)

distributorToPromote = d.getLeftLeg();

}

return distributorToPromote;

}

1. Class **Branch**: mô phỏng một cây nhị phân gồm các nhà phân phối

* Một số thuộc tính:

- manager: nhà phân phối là gốc của cây nhị phân

- memberList: danh sách các nhà phân phối trên cây hiện tại

* Một số phương thức quan trọng:

- Các phương thức để thêm một nhà phân phối mới vào cây nhị phân:

**checkForAdd(Distributor sponsor)**: kiểm tra xem có thể thêm nhà phân phối mới làm cấp dưới(trực tiếp hoặc gián tiếp) của nhà phân phối sponsor hay không

public boolean checkForAdd(Distributor sponsor) {

while(sponsor != null) {

boolean check;

if(sponsor.getParent() == null) {

sponsor = null;

continue;

}

Distributor p = sponsor.getParent();

int a1 = sponsor.numberOfInferiors();

int a2;

if(p.getRightLeg() == null || p.getLeftLeg() == null)

a2 = 0;

else if(p.getRightLeg() != sponsor)

a2 = p.getRightLeg().numberOfInferiors();

else

a2 = p.getLeftLeg().numberOfInferiors();

final int MAX\_DISPARITY = 2;

check = (a1 - a2 < MAX\_DISPARITY);

if(!check)

return false;

sponsor = sponsor.getParent();

}

return true;

}

**add(Distributor newMember, Distributor sponsor):** thêm nhà phân phối mới vào cây nhị phân và nhà phân phối mới là cấp dưới trực tiếp(hoặc gián tiếp) của nhà phân phối sponsor

private void add(Distributor newMember, Distributor sponsor) {

if(sponsor.getParent() != null) {

Distributor p = sponsor.getParent();

if(p.getLeftLeg() == null) {

p.setLeftLeg(newMember);

newMember.setParent(p);

return;

}

if(p.getRightLeg() == null) {

p.setRightLeg(newMember);

newMember.setParent(p);

return;

}

}

if(newMember.getSponsor() == null)

newMember.setSponsor(sponsor);

if(sponsor.getLeftLeg() == null) {

sponsor.setLeftLeg(newMember);

newMember.setParent(sponsor);

}

else if(sponsor.getRightLeg() == null) {

sponsor.setRightLeg((newMember));

newMember.setParent(sponsor);

}

else {

if(sponsor.getLeftLeg().numberOfInferiors() > sponsor.getRightLeg().numberOfInferiors())

add(newMember, sponsor.getRightLeg());

else {

add(newMember, sponsor.getLeftLeg());

}

}

}

- Các phương thức để xóa nhà phân phối khỏi cây nhị phân

**deleteDistributor(Distributor d):** xóa nhà phân phối d

public void deleteDistributor(Distributor d) {

memberList.remove(d);

Distributor toPromote = Distributor.distributorToPromote(d);

if(toPromote == null) {

if(d.getParent() != null) {

Distributor parent = d.getParent();

if(parent.getLeftLeg() == d)

parent.setLeftLeg(null);

else parent.setRightLeg(null);

}

return;

}

Distributor notPromote = (d.getLeftLeg() == toPromote) ? d.getRightLeg() : d.getLeftLeg();

Distributor parent = d.getParent();

if(parent != null) {

if(parent.getRightLeg() == d) {

parent.setRightLeg(toPromote);

} else {

parent.setLeftLeg(toPromote);

}

toPromote.setParent(parent);

} else {

manager = toPromote;

Main.updateManager(this);

toPromote.setParent(null);

}

Main.updateParent(toPromote);

if(notPromote != null) reorder(toPromote, notPromote);

}

1. Class **Sale:** lưu trữ dữ liệu cho 1 đợt phân phối hàng hóa

* Các thuộc tính:

- price: giá 1 đơn vị mặt hàng trong đợt phân phối hiện tại

- company: công ty phân phối trong đợt hiện tại

* Một số phương thức quan trọng:

**saleCommission(Distributor d)**: tính hoa hồng của nhà phân phối d thu được trực tiếp từ việc bán hàng

public double saleCommission(Distributor d) {

return d.getProductsSold() \* price \* saleCommissionRate;

}

**pairCommission(Distributor d)**: tính hoa hồng của nhà phân phối d thu được từ việc bán hàng của cấp dưới

public double pairCommission(Distributor d) {

if(d.getLeftLeg() == null || d.getRightLeg() == null) return 0;

if(totalProducts(d.getLeftLeg()) > totalProducts(d.getRightLeg())) {

return levelCommissionRate \* d.getRightLeg().getCommission();

} else return levelCommissionRate \* d.getLeftLeg().getCommission();

}

**commission(Distributor d)**: tổng hoa hồng hiện tại của nhà phân phối d

public void commission(Distributor d) {

d.setCommission(saleCommission(d) + pairCommission(d));

}

1. Class **DisplayPanel:** là 1 JPanel hiển thị trực quan cây nhị phân chứa các nhà phân phối hiện tại

* Các thuộc tính

- branch: cây cần hiển thị

- xs, ys: vị trí bắt đầu vẽ cây trên panel

* Các phương thức để vẽ cây nhị phân:

protected void paintComponent(Graphics g) {

g.setColor(getBackground()); //colors the window

g.fillRect(0, 0, getWidth(), getHeight());

g.setColor(getForeground()); //set color and fonts

Font MyFont = new Font("Arial",Font.BOLD,15);

g.setFont(MyFont);

xs = 20; //where to start printing on the panel

ys = 20;

g.drawString(branch.toString(), xs, ys);

MyFont = new Font("Arial",Font.BOLD,12);

g.setFont(MyFont);

this.drawTree(g, branch.getManager()); // draw the tree

revalidate(); //update the component panel

}

public void drawTree(Graphics g, Distributor d) {

int dx, dy, dx2, dy2;

int SCREEN\_WIDTH = 500;

int SCREEN\_HEIGHT = 500;

int X\_SCALE, Y\_SCALE;

X\_SCALE = SCREEN\_WIDTH / branch.totalNodes;

Y\_SCALE = (SCREEN\_HEIGHT - ys) / (branch.treeHeight(branch.getManager()) + 1);

if (d != null) {

drawTree(g, d.getLeftLeg());

dx = d.xpos \* X\_SCALE;

dy = d.ypos \* Y\_SCALE + ys;

String s = d.getName() + " ID:" + d.getID();

g.drawString(s, dx, dy);

s = d.getCommission() + " VND";

g.drawString(s, dx, dy + 12);

if (d.getLeftLeg() != null) {

dx2 = d.getLeftLeg().xpos \* X\_SCALE;

dy2 = d.getLeftLeg().ypos \* Y\_SCALE + ys;

g.drawLine(dx, dy, dx2, dy2);

}

if (d.getRightLeg() != null) {

dx2 = d.getRightLeg().xpos \* X\_SCALE;

dy2 = d.getRightLeg().ypos \* Y\_SCALE + ys;

g.drawLine(dx, dy, dx2, dy2);

}

drawTree(g, d.getRightLeg());

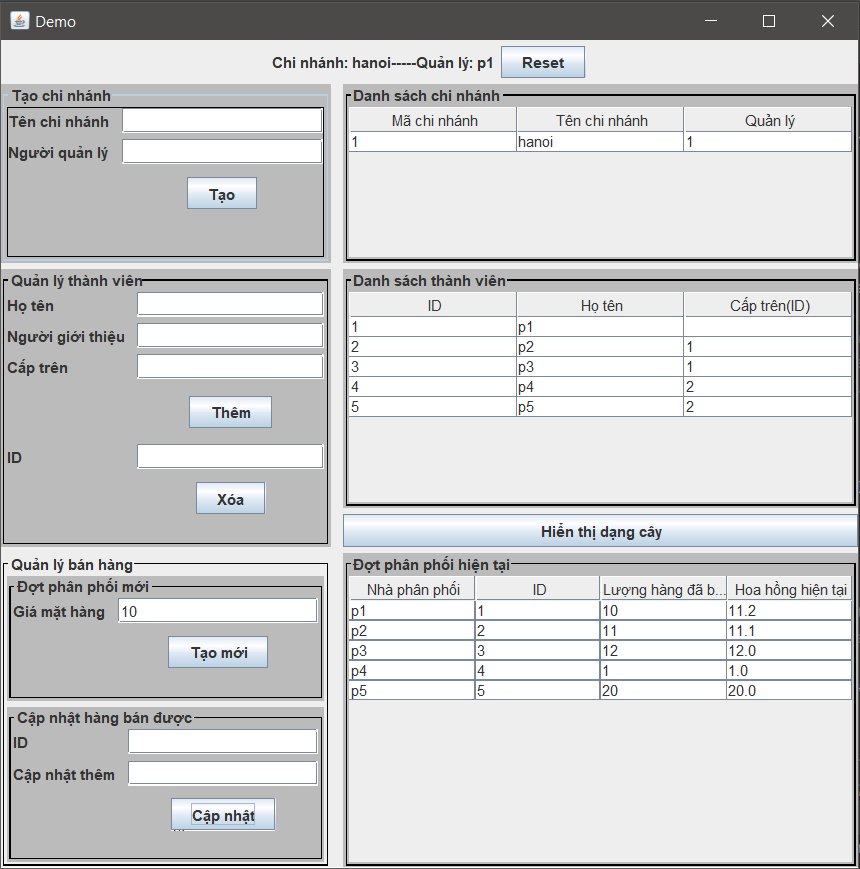
}

}

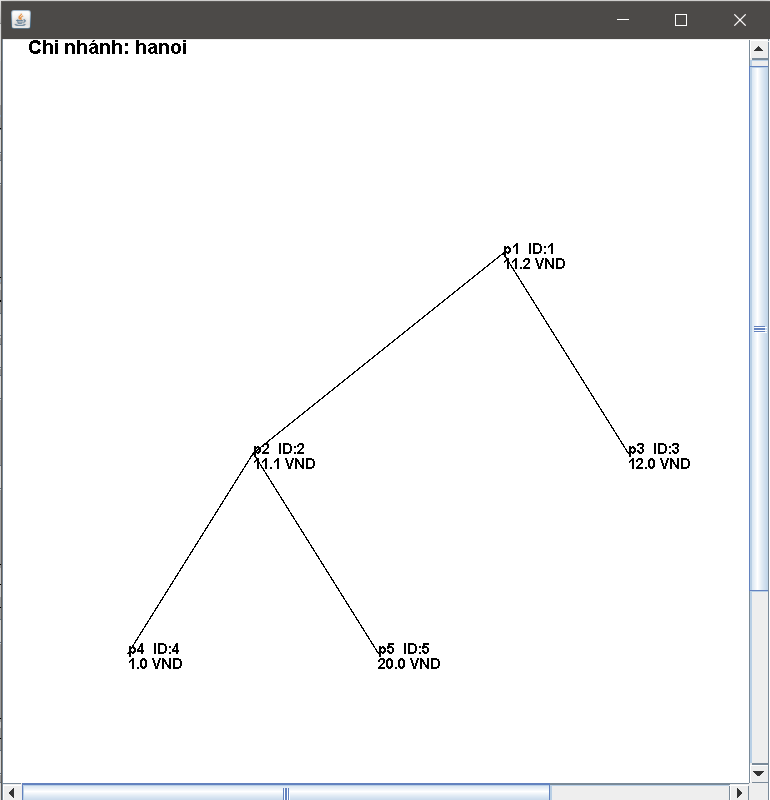
# **CHƯƠNG 3: KẾT QUẢ ĐẠT ĐƯỢC**

Chương trình được cài đặt với giao diện đơn giản bằng java swing để tiện sử dụng

**Giao diện chính:**

****

**Giao diện hiển thị trực quan dạng cây:**

****