PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

THI THỰC HÀNH CUỐI KỲ

REPORT

Tên: Nguyễn Văn Đạt

MSSV: 20127132

Khoa Công nghệ Thông tin

học Khoa học Tự nhiên TP HCM

Tháng Thg12-21

MỤC LỤC

I. Tổng quan

Thông tin Sinh Viên

Họ tên	**Nguyễn Văn Đạt**
MSSV	20127132
Email	20127132@student.hcmus.edu.vn

II. Phân Tích

Bài 1:

*Mô tả mã nguồn

1. PaymentHistory

```
class PaymentHistory
{
    string SoTaiKhoan;
    double SoTienChuyenKhoan;
    string NoidungChuyenKhoan;
    string NgayGiaoDich;
public:
    PaymentHistory();
    PaymentHistory(const PaymentHistory&);
    PaymentHistory(string, double, string, string);
    ~PaymentHistory();
    double getSoTienChuyenKhoan();
    void NhapLichSu();
    void showHistory();
};
```

PaymentHistory là lớp chứa các thông tin lịch sử thanh toán. Lớp gồm:

-SoTaiKhoan: Lưu thông tin số tài khoản của thẻ thanh toán bằng kiểu string

-SoTienChuyenKhoan: Lưu thông tin số tiền chuyển khoản của thẻ thanh toán bằng kiểu double

-NoidungChuyenKhoan: Lưu thông tin nội dung chuyển khoản bằng kiểu string

-NgayGiaoDich: Lưu thông tin ngày giao dịch bằng kiểu string

-Có các phương thức Contructor, Detructor, Copycontructor

-Một số hàm trả về các thông tin thẻ

- NhapLichSu, showHistory: Hàm nhập xuất lịch sử thanh toán

2. CreditCardAccount

```
class CreditCardAccount
{
protected:
```

```
    string SoThe;
    string NgayPhatHanh;
    string TaiKhoanLienKet;
    //Hạn mức tín dụng
    double creditLimit;
    //Lãi suất
    double interestRate;
    //Thanh toán tối thiểu
    double minPayment;
    //Phí trả chậm
    double latePenalty;
    //Số dư hiện tại
    double balance;
    //Lịch Sử Giao Dịch
    vector<PaymentHistory> LichSuGiaoDich;
public:
    CreditCardAccount();
    CreditCardAccount(const CreditCardAccount&);
    CreditCardAccount(double, double, double, double, double);
    ~CreditCardAccount();
    string LaySoThe();
    double getBalance();
    double getInterestRate();
    double getMinPayment();
    double getLatePenalty();
    void setBalance(double);
    void setMinPayment(double);
    bool charge(double amount);
    void payment(double amount);
    double SoTienConNo();
    virtual void thanhtoan(double, string) = 0;
    virtual void setLichSuGiaoDich(PaymentHistory);
    virtual void showHistory();
    virtual void TaoTaiKhoan();
    virtual void XuatThongTin();
};
```

CreditCardAccount là lớp cha chứa các thông tin thẻ tín dụng được sử dụng để thanh toán các khoản phí trả sau cho tài khoản thanh toán. Lớp gồm:

-SoThe: Lưu thông tin số thẻ của thẻ tín dụng bằng kiểu string

-NgayPhatHanh: Lưu thông tin ngày phát hành thẻ tín dụng bằng kiểu string

-TaiKhoanLienKet: Lưu thông tin tài khoản liên kết với thẻ tín dụng bằng kiểu string

-creditLimit: Lưu thông tin hạn mức của thẻ tín dụng bằng kiểu double

-interestRate: Lưu thông tin lãi suất của thẻ tín dụng bằng kiểu double

-minPayment: Lưu thông tin thanh toán tối thiểu của thẻ tín dụng bằng kiểu double

-latePenalty: Lưu thông tin phí trả chậm của thẻ tín dụng bằng kiểu double

-balance: Lưu thông tin số dư hiện tại của thẻ tín dụng bằng kiểu double

-LichSuGiaoDich: Lưu thông tin lịch sử giao dịch của thẻ tín dụng bằng vector kiểu của lớp PaymentHistory

-Có các phương thức Contructor, Detructor, Copycontructor

-Một số hàm trả về các thông tin thẻ

-charge: Hàm kiểu bool. Kiểm tra tài khoản hiện tại có thể thanh toán số tiềnamount hay không. Tài khoản thực hiện được thanh toán khi amount + balance <= creditLimit. Nếu thanh toán được thì thực hiện cộng amount vào balance của tài khoản thẻ và trả về True. Nếu không thanh toán được thì trả về False

-payment: Là hàm thanh toán thẻ tín dụng, thực hiện trừ số tiền amount vào balance (lúc này balance có thể đạt giá trị âm).

-thanhtoan: Hàm ảo.

-setLichSuGiaoDich: Hàm ảo, Hàm để thêm phần tử vào vector LichSuGiaoDich

-showHistory: Hàm ảo, Xuất thông tin LichSuGiaoDich

-TaoTaiKhoan, XuatThongTin: Hàm ảo, Để tạo và xuất thông tin tài khoản

3. CashBackCardAccount

```
class CashBackCardAccount : public CreditCardAccount
{
    float cashBackRate;
    int currentCashBack;
public:
    CashBackCardAccount();
    CashBackCardAccount(const CashBackCardAccount&);
    CashBackCardAccount(double, double, double, double, double, float, int);
    ~CashBackCardAccount();
    double getCurrentCashBack();
```

```
    bool charge(double);
    void redeemCashBack();
    void TaoTaiKhoan();
    void XuatThongTin();
    void thanhtoan(double, string);
};
```

CashBackCardAccount là lớp con của CreditCardAccount chứa các thông tin thẻ tín dụng kế thừa từ lớp cha được sử dụng để thanh toán các khoản phí trả sau cho tài khoản thanh toán. Lớp gồm:

-cashbackRate: Lưu tỉ lệ hoàn tiền bằng kiểu double

-currentCashBack: Lưu số tiền hiện tại trong tài khoản bằng kiểu double

-Có các phương thức Contructor, Detructor, Copycontructor

-Một số hàm trả về các thông tin thẻ

-charge: Hàm kiểu bool. Xử lý tương tự CreditCardAccount, thêm vào đó là khi phương thức trả về True cần phải tính số tiền hoàn lại được nhận là amount nhân với cashBackRatevà thêm số tiền này vào currentCashBack.

-redeemCashBack: để tiến hành hoàn tiền, phương thức sẽ giảm số tiền currentCashBack trực tiếp vào trong balance và đặt currentCashBack lại mặc định bằng 0.

-TaoTaiKhoan, XuatTaiKhoan: được kế thừa từ lớp cha.

4. RewardCardAccount

```
class RewardCardAccount :public CreditCardAccount
{
    float rewardRate;
    int currentPoints;
public:
    RewardCardAccount();
    RewardCardAccount(const RewardCardAccount&);
    RewardCardAccount(double, double, double, double, double, float, int);
    ~RewardCardAccount();
    double getCurrentPoints();
    bool charge(double);
    void payWithPoints(int);
    void TaoTaiKhoan();
```

```
    void XuatThongTin();
    void thanhtoan(double, string);
};
```

RewardCardAccount là lớp con của CreditCardAccount chứa các thông tin thẻ tín dụng kế thừa từ lớp cha được sử dụng để thanh toán các khoản phí trả sau cho tài khoản thanh toán. Lớp gồm:

-rewardRate: Lưu tỉ lệ hoàn điểm bằng kiểu double

-currentPoints: Lưu số điểm hiện tại trong tài khoản bằng kiểu double

-Có các phương thức Contructor, Detructor, Copycontructor

-Một số hàm trả về các thông tin thẻ

-charge: Hàm kiểu bool. Xử lý tương tự CreditCardAccount, thêm vào đó là khi phương thứctrả về True cần phải tính số điểm được nhận là amount nhân với rewardRate, chuyển số điểm về kiểu int (làm tròn xuống) và thêm số điểm này vào currentPoints.

-payWithPoints: để thanh toán giao dịch với số điểm là pAmount. Nếu pAmount lớn hơn curentPoints, phương thức không cần thực hiện thao tác nào. Ngược lại, giảm curentPoints đi một lượng pAmount.

-TaoTaiKhoan, XuatTaiKhoan: được kế thừa từ lớp cha.

5. PaymentAccount

```
class PaymentAccount
{
    string SoTaiKhoan;
    double SoDu;
    string NgayPhatHanh;
    vector<PaymentHistory> LichSuGiaoDich;
    //Nợ kỳ trước
    double SoNo;
    vector<CreditCardAccount*> m_creditCard;
public:
    PaymentAccount();
    PaymentAccount(const PaymentAccount&);
    PaymentAccount(string, double, string, vector<PaymentHistory>);
    ~PaymentAccount();
    int lenght();
    int getNumber(char);
    double getBalance();
```

```
    string GetNgayPhatHanh();
    bool transferTo(double amount);
    void showHistory();
    void showListCard();
    void thanhtoan(int, double, string);
    void addCreditCardAccount(CreditCardAccount*);
    void payment(int, string);
    friend istream& operator>> (istream&, PaymentAccount&);
    friend ostream& operator<< (ostream&, PaymentAccount&);
};
```

PaymentAccount là lớp chứa các thông tin thẻ thanh toán được sử dụng để thanh toán các khoản phí từ nhiều nguồn. Lớp gồm:

-SoTaiKhoan: Lưu thông tin số tài khoản của thẻ bằng kiểu string

-SoDu: Lưu thông tin số dư tài khoản của thẻ bằng kiểu double

-NgayPhatHanh: Lưu thông tin ngày phát hành thẻ bằng kiểu string

-SoNo: Lưu thông tin số nợ tài khoản của thẻ bằng kiểu double

-LichSuGiaoDich: Lưu thông tin lịch sử giao dịch bằng vector kiểu của lớp PaymentHistory

-m_creditCard: Lưu thông tin các thẻ tín dụng liên kết bằng vector kiểu của lớp CreditCardAccount

-Có các phương thức Contructor, Detructor, Copycontructor

-Một số hàm trả về các thông tin thẻ

-payment: Là hàm thanh toán thẻ nợ và các thẻ tín dụng.

-thanhtoan: Hàm trung gian để thanh toán các thẻ tín dụng

-addCreditCardAccount: Hàm để thêm phần tử vào vector m_creditCard

-showHistory: Xuất thông tin LichSuGiaoDich

- showListCard: Xuất thông tin toàn bộ thẻ tín dụng

-TaoTaiKhoan, XuatThongTin: Hàm ảo, Để tạo và xuất thông tin tài khoản

-Hàm friend với các thông tin nhập xuất thẻ thanh toán

6. Run

```
void run()
{
    int count1 = 0, count2 = 0;
    int choose;
    //Lấy ngày phát hành làm bắt đầu
    string temp = account.GetNgayPhatHanh();
    int day = account.getNumber(temp[0]) * 10 + account.getNumber(temp[1]);
    int month = account.getNumber(temp[3]) * 10 + account.getNumber(temp[4]);
    int year = account.getNumber(temp[6]) * 1000 + account.getNumber(temp[7]) * 100 +
        account.getNumber(temp[8]) * 10 + account.getNumber(temp[9]);
    while (1)
    {
        if (account.lenght() > 0)
        {
            if (count1 == 30 || count2 == 15)
            {
                //trả nợ
                cout << "Ngay Den Han: ";
                cout << day << "/" << month << "/" << year << endl;
                cout << "Ban Co Muon Thanh Toan So Tien The Tin Dung Khong(1: Co, 2:
Khong): ";
                cin >> choose;
                while (choose != 1 && choose != 2)
                {
                    cout << "Nhap Lai: ";
                    cin >> choose;
                }
                if (choose == 1)
                {
                    cout << "Ban Muon Thanh Toan Toan Bo Hay Toi Thieu (1: Toan Bo, 2:
Toi Thieu): ";
                    cin >> choose;
                    while (choose != 1 && choose != 2)
                    {
                        cout << "Nhap Lai: ";
                        cin >> choose;
                    }
                    account.payment(choose, to_string(day) + "/" + to_string(month) +
"/" + to_string(year));
                    count1 = count2 = -1;
                }
            }
```

```
            if (day == 15)
            {
                //sao kê
                cout << "Ngay Sao Ke: ";
                cout << day << "/" << month << "/" << year << endl;
                account.showHistory();
            }
        }
        NgayTiepTheo(day, month, year);
        if (count2 != 0)
            count2++;
        if (count1 == 30)
        {
            //mỗi 30 ngày hỏi có muốn tạo thêm tài khoản không
            count1 = -1;
            count2++;
            cout << "Ban Co Muon Them The Tin Dung (1: Co, 2: Khong): ";
            cin >> choose;
            while (choose != 1 && choose != 2)
            {
                cout << "Nhap Lai: ";
                cin >> choose;
            }
            if (choose == 1)
            {
                cout << "1. The Tich Diem\n";
                cout << "2. The Hoan Tien\n";
                cout << "Chon: ";
                cin >> choose;
                CreditCardAccount* acc;
                if (choose == 1)
                    acc = new RewardCardAccount();
                else
                    acc = new CashBackCardAccount();
                acc->TaoTaiKhoan();
                account.addCreditCardAccount(acc);
            }
        }
        if (count1 % 4 == 0)
        {
            //mỗi 4 ngày thì sử dụng thẻ 1 lần
            if (account.lenght() > 0 && account.lenght() <= 6)
            {
```

```
                cout << "Ban Co Muon Su Dung Tin Dung (1: Co, 2: Khong): ";
                cin >> choose;
                while (choose != 1 && choose != 2)
                {
                    cout << "Nhap Lai: ";
                    cin >> choose;
                }
                if (choose == 1)
                {
                    double amount;
                    account.showListCard();
                    cout << "Chon Su Dung The So: ";
                    cin >> choose;
                    cout << "So Tien: ";
                    cin >> amount;
                    account.thanhtoan(choose, amount, to_string(day) + "/" +
to_string(month) + "/" + to_string(year));
                    account.showHistory();
                }
            }
        }
        count1++;
    }
}
int main()
{
    int choose = 1;
    while (1)
    {
        cout << "Tao Tai Khoan Thanh Toan\n";
        cin >> account;
        run();
        return 0;
    }
}
```

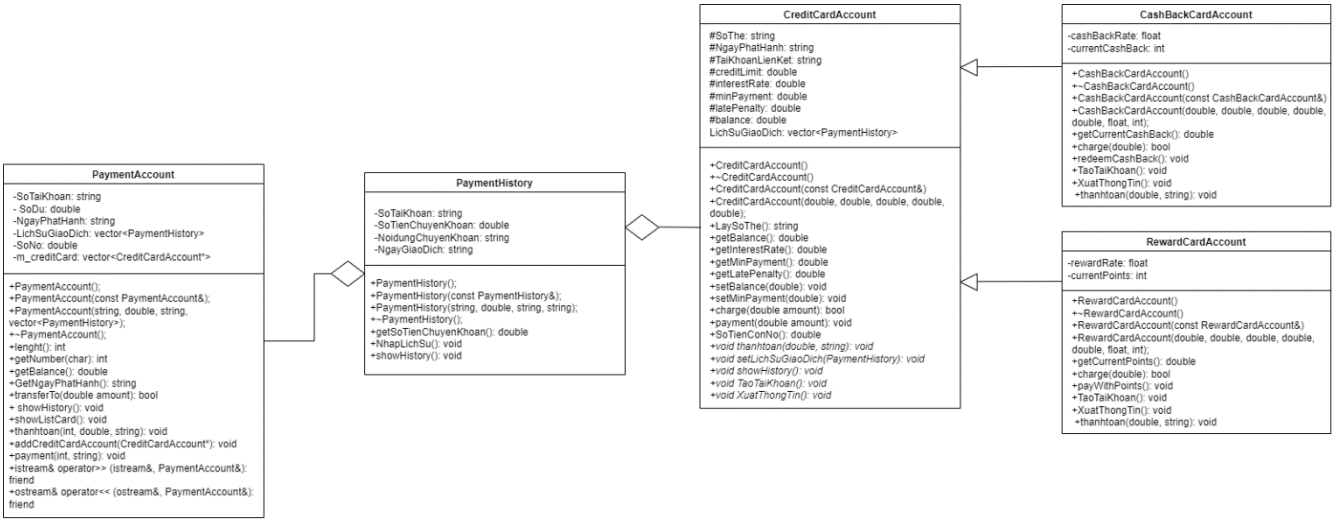
Khi vừa bắt đầu chạy hàm sẽ tạo tài khoản rồi chạy hàm run().

Trong hàm run, khi bắt đầu chạy, khởi tạo ngày tạo tài khoản là ngày bắt đầu, cứ đến sau 30 ngày sẽ có thông báo đóng tiền, nếu không đóng thì sau 15 ngày có có thông báo tiếp, nếu vẫn không đóng thì số tiền đó sẽ cộng vào tiền nợ và tính cho tháng sau. Ngày 15 hàng tháng sẽ tự xuất ra bảng sao kê lịch sử thanh toán của thẻ thanh toán và thẻ tín

dụng. Cứ 30 ngày, hệ thống hỏi có muốn tạo thêm thẻ tín dụng không, nếu đã đạt số lượng tối đa sẽ không hỏi. Cứ sau 4 ngày sẽ có dòng hỏi có muốn sử dụng thẻ hay thông.

*Đoạn mã tự đề xuất: Thêm ngày vào hàm để cho trải nghiệm sử dụng thẻ bằng chương trình hiện thực hơn.

***UML:**



Bài 2:

*Ý tưởng:

-Thực hiện trò chơi "Đại chiến hai thế giới" bằng cách chia ra hai phe, một phe thiện và phe ác, phe thiện do ta điều khiển và chỉ huy tiến đánh đập tan phe ác. Khởi tạo sẽ có có các công trình thêm lính, công trình tăng cấp độ, nếu chưa lính thì công trình tăng cấp không có tác dụng, phải tạo ra lính bằng các đến công trình thêm lính. Ăn lính càng nhiều thì lực lượng càng mạnh đánh bại các quái vật, nếu ta không còn quân lính nào thì thất bại, nếu các quái vật bị hạ toàn bộ bị qua màn và tăng sức mạnh quái vật lên. Khi đạt được số màn chơi tối đa, người chơi sẽ chiến thắng trò chơi.

-Phe ta có 02 loại lính là lính thủ (lính cầm khiên) và lính tấn công (lính cầm cung). Lính thủ có sức phòng thủ cao nhưng đổi lại sức tấn công khá yếu. Lính tấn công thì ngược lại có sức phòng thủ thấp nhưng đổi lại sức tấn công khá cao.

-Phe địch có nhiều quái vật có nhiều máu và sức tấn công rất cao, có thể tấn công các quân lính phe ta.

-Tạo bàn cờ 4x4, chứa các tên các loại lính, công trình, quái vật gồm 02 kí tự, nhân vật được thay thế bằng "Me".

*Mô tả mã nguồn:

1. CongTrinh

```
class CongTrinh
{
protected:
    string name;
    int heal;
    int dame;
    int def;
    int x, y;
public:
    CongTrinh();
    ~CongTrinh();
    CongTrinh(const CongTrinh&);
    CongTrinh(string, int, int, int);
    string GetName();
    bool LinhDef();
    int X();
```

```
    int Y();
    int GetHeal();
    int GetDame();
    int GetDef();
    void setPos(int, int);
    virtual void Draw() = 0;
    virtual bool LevelUp() = 0;
    virtual void ThemCongTrinh() = 0;
};
```

```
class CongTrinhNangCap :public CongTrinh
{
public:
    CongTrinhNangCap();
    CongTrinhNangCap(const CongTrinhNangCap&);
    CongTrinhNangCap(string, int, int, int);
    void Draw();
    bool LevelUp();
    void ThemCongTrinh();
};
```

```
class CongTrinhThemLinh :public CongTrinh
{
public:
    CongTrinhThemLinh();
    CongTrinhThemLinh(const CongTrinhThemLinh&);
    CongTrinhThemLinh(string, int, int, int);
    void Draw();
    bool LevelUp();
    void ThemCongTrinh();
};
```

CongTrinh là lớp cha của lớp CongTrinhNangCap và CongTrinhThemLinh chứa thông tin của các công trình nâng cấp và công trình thêm lính, các thuộc tính của công trình sẽ là thuộc tính của lính khi lính được chiêu mộ trong công trình thêm lính:

-name: Tên của công trình (kiểu string)

-heal, dame, def, x, y: lần lượt là máu, sức mạnh, phòng thủ, vị trí của công trình. (kiểu int)

-Một số làm lấy thuộc tính trong lớp.

-Detructor, contructor, Copy Contructor

-Các hàm ảo: Draw, LevelUp, ThemCongTrinh lần lượt để vẽ công trình, nếu là công trình nâng cấp thì LevelUp sẽ trả về True, ngược lại trả về false, và nhập thông tin công trình khi khởi tạo lại trò chơi.

2. Linh

```
class Linh
{
protected:
    string name;
    int heal;
    int dame;
    int def;
    int x, y;
public:
    Linh();
    ~Linh();
    Linh(const Linh&);
    Linh(string, int, int, int, int, int);
    string GetName();
    int GetDame();
    int X();
    int Y();
    bool battle(int);
    void setPos(int, int);
    virtual void Draw() = 0;
    virtual void LevelUp() = 0;
};
```

```
class LinhKhien :public Linh
{
public:
    LinhKhien();
    ~LinhKhien() {}
    LinhKhien(const LinhKhien&);
    LinhKhien(string, int, int, int, int, int);
    void Draw();
    void LevelUp();
};
```

```
class CungThu : public Linh
{
public:
    CungThu();
    ~CungThu() {}
    CungThu(const CungThu&);
    CungThu(string, int, int, int, int, int);
    void Draw();
    void LevelUp();
};
```

Linh là lớp cha của lớp CungThu và LinhKhien chứa thông tin của Linh được kế thừa lại từ lớp CongTrinhThemLinh:

-name: Tên của lính (kiểu string)

-heal, dame, def, x, y: lần lượt là máu, sức mạnh, phòng thủ, vị trí của lính. (kiểu int)

-Một số làm lấy thuộc tính trong lớp.

-Detructor, contructor, Copy Contructor

-Các hàm ảo: Draw, LevelUp lần lượt để vẽ lính, LevelUp khác nhau giữa lính tấn công (tấn công + 5, phòng thủ + 1) và phòng thủ (tấn công + 1, phòng thủ + 5)

3. Player

```
class Player
{
    int x, y;
    bool dead;
    vector<Linh*> m_Linh;
public:
    Player();
    ~Player();
    int X();
    int Y();
    int length();
    bool IsDead();
    bool IsImpact(vector<Linh*>&);
    bool IsImpact(vector<QuaiVat*>&);
    bool IsImpact(CongTrinh*);
    void Draw();
```

```
    void move(char);
    void setPos(int, int);
};
```

Player là lớp chứa thông tin của người chơi:

-dead: tình trạng người chơi (kiểu bool)

- x, y: vị trí của người chơi. (kiểu int)

- m_Linh: số lính người chơi hiện có (kiểu vector)

-Một số làm lấy thuộc tính trong lớp.

-Detructor, contructor

- IsImpact: Xử lý va chạm khi đến các đối tượng (lính, quái vật, công trình)

-Các hàm: Draw, move, setPos lần lượt để vẽ người, di chuyển sau khi nhập và thiết lập vị trí cho người chơi.

4. Game

```
class Game
{
    Player* m_Player;
    CongTrinhNangCap* m_CongTrinhNangCap;
    CongTrinhThemLinh* m_CongTrinhThemLinh;
    vector<QuaiVat*> m_QuaiVat;
    vector<Linh*> m_dsLinh;
    int level;
    bool nextLevel;
public:
    Game();
    ~Game();
    void Draw();
    void move(char);
    void resetGame();
    void IsImpact();
    void clearGame();
    void findPos(int&, int&);
    int curLev();
    bool IsNextLevel();
    bool IsDead();
};
```

Game là lớp chính để thực hiện trò chơi:

- m_Player: thông tin người chơi (lớp Player)

- m_CongTrinhNangCap: thông tin công trình nâng cấp (lớp CongTrinhNangCap)

- m_CongTrinhThemLinh: thông tin công trình thêm lính (lớp CongTrinhThemLinh)

- m_QuaiVat: thông tin các quái vật (lớp QuaiVat)

- m_dsLinh: thông tin lính đang có trên bản đồ (lớp Linh)

- level: level hiện tại của trò chơi. (kiểu int)

- nextLevel: kiểm tra xem có hợp lệ để tăng level không, nếu có trả về True, không thì False (kiểu bool)

-Một số làm lấy thuộc tính trong lớp.

-Detructor, contructor

-IsImpact: Xử lý va chạm khi đến các đối tượng (lính, quái vật, công trình), khi đến vị trí của đối tượng nào thì sẽ gọi hàm IsImpact trong lớp Player ra để thực hiện xử lý, riêng khi va chạm công trình thêm lính sẽ có thêm xử lý là nếu lính đó tấn công lớn hơn phòng thủ thì trả về cung thủ và ngược lại. Sau khi thực hiện xử lý va chạm, đối tượng đó sẽ bị xóa và thêm bằng đối tượng mới có bị trí khác. Nếu qua màn thì trả về nextLevel là True, nếu thua thì trả về dead là True.

-clearGame: Xóa tất cả đối tượng trên bản đồ

-findPos: tìm vị trí trống để thêm các đối tượng (lính, quái vật, công trình)

*Mã nguồn tâm đắc nhất: IsImpact trong lớp Game và Player

```
bool Player::IsImpact(vector<Linh*>& linh)
{
    for (int i = 0; i < linh.size(); i++)
    {
        if (linh[i]->X() == x && linh[i]->Y() == y)
        {
            m_Linh.push_back(linh[i]);
            linh.erase(linh.begin() + i);
            return true;
        }
    }
    return false;
}
```

```
bool Player::IsImpact(vector<QuaiVat*>& quai)
{
    for (int i = 0; i < quai.size(); i++)
    {
        if (quai[i]->X() == x && quai[i]->Y() == y)
        {
            for (int j = 0; j < m_Linh.size(); j++)
            {
                //nếu lính chết -> xóa
                if (m_Linh[j]->battle(quai[i]->GetDame()))
                    m_Linh.erase(m_Linh.begin() + j);
                //trừ máu quái
                quai[i]->setHeal(m_Linh[j]->GetDame());
            }
            if (quai[i]->dead())
                quai.erase(quai.begin() + i);
            else // Hết lính mà quái chưa chết -> thua
                dead = true;
            return true;
        }
    }
    return false;
}
bool Player::IsImpact(CongTrinh* congtrinh)
{
    if (congtrinh->X() == x && congtrinh->Y() == y)
    {
        //công trình nâng cấp
        if (congtrinh->LevelUp())
            for (int i = 0; i < m_Linh.size(); i++)
                m_Linh[i]->LevelUp();
        //công trình thêm lính xử lý ở class Game
        return true;
    }
    return false;
}
```

```
void Game::IsImpact()
{
    int x, y;
    //đến công trình nâng cấp
    if (m_Player->IsImpact(m_CongTrinhNangCap))
```

```
    {
        //thêm vào vị trí mới
        findPos(x, y);
        m_CongTrinhNangCap->setPos(x, y);
    }
    //đến công trình thêm lính
    else if (m_Player->IsImpact(m_CongTrinhThemLinh))
    {
        findPos(x, y);
        //nếu là lính phòng thủ thì def > dame
        if (m_CongTrinhThemLinh->LinhDef())
        {
            m_dsLinh.push_back(new LinhKhien(m_CongTrinhThemLinh->GetName(),
m_CongTrinhThemLinh->GetHeal(),
                m_CongTrinhThemLinh->GetDame(), m_CongTrinhThemLinh->GetDef(), x, y));
        }
        //nếu là lính cung thủ thì def < dame
        else
            m_dsLinh.push_back(new CungThu(m_CongTrinhThemLinh->GetName(),
m_CongTrinhThemLinh->GetHeal(),
                m_CongTrinhThemLinh->GetDame(), m_CongTrinhThemLinh->GetDef(), x, y));
        //thêm vào vị trí mới
        findPos(x, y);
        m_CongTrinhThemLinh->setPos(x, y);
        m_CongTrinhThemLinh->ThemCongTrinh();
    }
    //đến gặp lính thì thêm lính đó vào player, xóa trong danh sách lính
    else if (m_Player->IsImpact(m_dsLinh)){}
    //đến gặp quái vật, nếu đánh quái vật không chết -> thua
    //nếu đánh thắng thì trừ máu
    //nếu quên lính hết máu thì xóa
    else if (m_Player->IsImpact(m_QuaiVat))
    {
        if (m_QuaiVat.size() == 0)
            nextLevel = true;
    }
    Draw();
}
```

Hai mã nguồn này là một nhưng khi đưa vào lớp Game, một số xử lý trong IsImpact (lớp Player) không thể thực hiện như xử lý thắng khi hết quái vật, thu khi không còn lính, hay thêm lính mà đối tượng công trình được khởi tạo trong lớp Game.

Hai mã nguồn này làm mất rất nhiều thời gian và công sức nhưng nó vẫn chưa được hoàn thiện hoàn toàn về khía cạnh của người làm trò chơi như thêm âm thanh, các hiệu ứng, hay trả về các dòng thông báo số lượng lính, quái rõ ràng. Chưa hoàn thiện được kiểm tra tên khi người dùng nhập trùng để cho có sự lẫn lộn nhiều phía người chơi.

-Về mã nguồn của lớp Player:

+Linh: Người chơi đến vị trí lính lớp Game thì sẽ thêm quân lính đó vào lớp Player, sau đó xóa lính đó ở lớp Game

+QuaiVat: Người chơi đến vị trí quái vật lần lượt các quân lính khiên của người chơi xông lên tấn công trước, nếu quái vật chết, sẽ trừ máu quân lính, nếu quân lính mất hết máu thì xóa lính đó đi, quái vật sẽ bị trừ máu, lính tiếp theo sẽ tấn công, đến khi hết quân lính mà quái vật chưa chết thì người chơi thua.

+CongTrinh: Khi đến được công trình, nếu là công trình nâng cấp thì sẽ nâng cấp cho toàn bị quân lính của người chơi, không thì trả về true và sẽ xử lý tiếp trong lớp Game là công trình thêm lính, nếu sức mạnh lớn hơn phòng thì thêm vào cung thủ, và ngược lại.

-Về mã nguồn của lớp Game:

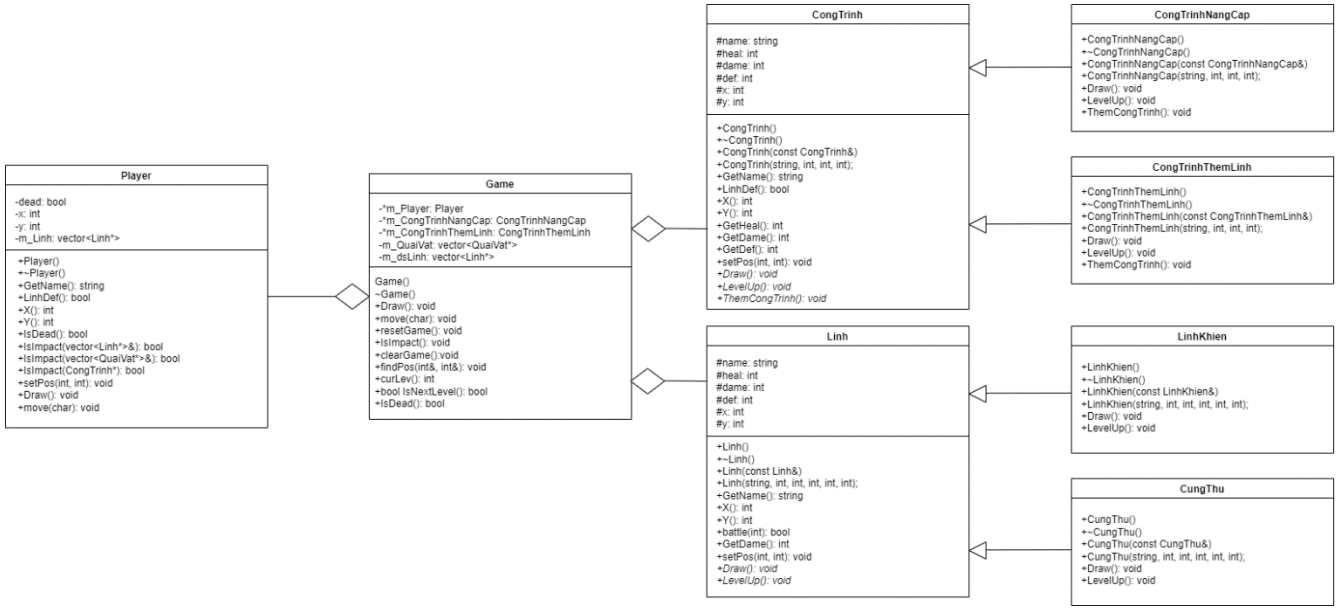
+CongTrinhNangCap: Không thay đổi tên, chỉ thay đổi vị trí

+ CongTrinhThemLinh: Như đã mô tả ở trên, sau khi hoàn thành sẽ tạo lại công trình và thay đổi vị trí công trình.

+Linh: Không xử lý, chỉ xử lý trong lớp Player.

+QuaiVat: Nếu quái vật không còn thì qua màn.

*UML:


Player
-dead: bool
-x: int
-y: int
-m_Linh: vector<Linh*>
+Player()
+~Player()
+GetName(): string
+LinhDef(): bool
+X(): int
+Y(): int
+IsDead(): bool
+IsImpact(vector<Linh*>&): bool
+IsImpact(vector<QuaiVat*>&): bool
+IsImpact(CongTrinh*): bool
+setPos(int, int): void
+Draw(): void
+move(char): void
Game
-*m_Player: Player
-*m_CongTrinhNangCap: CongTrinhNangCap
-*m_CongTrinhThemLinh: CongTrinhThemLinh
-m_QuaiVat: vector<QuaiVat*>
-m_dsLinh: vector<Linh*>
Game()
~Game()
+Draw(): void
+move(char): void
+resetGame(): void
+IsImpact(): void
+clearGame():void
+findPos(int&, int&): void
+curLev(): int
+bool IsNextLevel(): bool
+IsDead(): bool
CongTrinh
#name: string
#heal: int
#dame: int
#def: int
#x: int
#y: int
+CongTrinh()
+~CongTrinh()
+CongTrinh(const CongTrinh&)
+CongTrinh(string, int, int, int);
+GetName(): string
+LinhDef(): bool
+X(): int
+Y(): int
+GetHeal(): int
+GetDame(): int
+GetDef(): int
+setPos(int, int): void
+Draw(): void
+LevelUp(): void
+ThemCongTrinh(): void
CongTrinhNangCap
+CongTrinhNangCap()
+~CongTrinhNangCap()
+CongTrinhNangCap(const CongTrinhNangCap&)
+CongTrinhNangCap(string, int, int, int);
+Draw(): void
+LevelUp(): void
+ThemCongTrinh(): void
CongTrinhThemLinh
+CongTrinhThemLinh()
+~CongTrinhThemLinh()
+CongTrinhThemLinh(const CongTrinhThemLinh&)
+CongTrinhThemLinh(string, int, int, int);
+Draw(): void
+LevelUp(): void
+ThemCongTrinh(): void
Linh
#name: string
#heal: int
#dame: int
#def: int
#x: int
#y: int
+Linh()
+~Linh()
+Linh(const Linh&)
+Linh(string, int, int, int, int, int);
+GetName(): string
+X(): int
+Y(): int
+battle(int): bool
+GetDame(): int
+setPos(int, int): void
+Draw(): void
+LevelUp(): void
LinhKhien
+LinhKhien()
+~LinhKhien()
+LinhKhien(const LinhKhien&)
+LinhKhien(string, int, int, int, int, int);
+Draw(): void
+LevelUp(): void
CungThu
+CungThu()
+~CungThu()
+CungThu(const CungThu&)
+CungThu(string, int, int, int, int, int);
+Draw(): void
+LevelUp(): void

III. Đánh giá

BẢNG TỰ ĐÁNH GIÁ CÁC YÊU CẦU

STT	Bài	Yêu cầu	Tỉ lệ hoàn thành
1	**1**	Xây dựng các lớp PaymentAccount, PaymentHistory, CreditCardAccount, RewardCardAccount, CashBackCardAccount với các thuộc tính và phương thức mô tả ở trên	100%
2		Thiết kế đúng các mối quan hệ giữa các lớp và viết đúng các phương thức được mô tả	80%
3		Hoàn thiện phần nhập xuất cho chương trình chính và từng lớp	80%
4		Loại tài khoản tín dụng có chức năng phát hành thẻ để sử dụng trực tiếp. Mỗi tài khoản có thể phát hành 1 thẻ chính và tối đa 5 thẻ phụ. Mỗi thẻ có các thông tin về số thẻ, ngày phát hành và tài khoản thẻ liên kết đến. Hãy thiết kế và viết thêm các thành phần để thực hiện chức năng này	70%
5		Thiết kế chương trình chính và các lớp để hỗ trợ cho việc thực hiện thanh toán thẻ tính dụng và sao kê thẻ tính dụng vào cuối kì (vào ngày 15 hằng tháng).	90%
6	**2**	Bạn hãy xây dựng chương trình C++ đề xuất phát triển hệ thống mã nguồn cho trò chơi "Đại chiến hai thế giới" sử dụng kiến thức lập trình hướng đối tượng với các kỹ thuật kế thừa, đa hình và áp dụng các mẫu thiết kế hướng đối tượng phù hợp dựa vào mô tả nêu trên.	90%
7		Bạn hãy viết báo cáo để mô tả lại hệ thống mã nguồn mà bạn đề xuất xây dựng, đồng thời đưa ra những nhận định, nhận xét và phân tích về các đoạn mã nguồn mà bạn tâm đắc trong đồ án (khả năng mở rộng, tùy biến linh hoạt, tính tổng quát hóa, …)	100%
8		Bạn hãy viết báo cáo để mô tả lại hệ thống mã nguồn mà bạn đề xuất xây dựng, đồng thời đưa ra những nhận định, nhận xét và phân tích về các đoạn mã nguồn mà bạn tâm đắc trong đồ án (khả năng mở rộng, tùy biến linh hoạt, tính tổng quát hóa, …)	100%