TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: XÂY DỰNG GAME NHẬP VAI SimpleRPG

NHÓM 21

Sinh viên:	Mã số sinh viên
Nguyễn Văn Túc	20145070
Ngô Quang Hòa	20141840
Hoàng Ngọc Dũng	20140782
Nguyễn Trần Nam	20143069
Nguyễn Văn Đức	20141168

Hà Nội, Ngày 7 tháng 12 năm 2016

Mục lục

Trang

CHƯƠNG I: TỔNG QUAN
GIỚI THIỆU ĐỀ TÀI

1. Game RPG

RPG, viết tắt của cụm từ Role_Playing Games, là thể loại game nhập vai chiến đấu theo lượt hoặc theo thời gian thực và cũng là thể loại game khá phổ biến trên thị trường gaem PC hay mobile hiện nay.
Trong game người chơi sẽ điều khiển một, hoặc một nhóm nhân vật trong thế giới ảo. có khả năng nâng cấp bản thân và chịu trách nhiệm cho hành động cũng như định hướng phát triên của chính nhân vật đó

2. Một sô yếu tố cấu thành nên dòng RPG

- Một vai trò trong game

  Là một game nhập vai chắc chắn nhân vật của bạn phải có một vai trò rõ ràng nào đó trong game. Bạn sẽ tự lựa mình lựa chọn vai trò của mình trong trờ chơi quyệt định tới sự phát triển và sự tương tác giữa các nhân vật

- Cốt truyện

  Một game RPG chuyên nghiệp không thể thiếu được một cốt truyện cuốn hút và có chiều sâu. Sự cuống hút người chơi của loại game này ấn sâu trong cốt truyện mà các trò chơi nhập vai mang lại. Tuy nhiên, trong khuôn khổ và mục đích chính của môn học nhóm đã không thực hiện tạo cốt truyện cho chương trình của mình

- Hệ thống chiên đấu

  Đây là một yêu tố khiến bạn tốn nhiều thời gian nhất vào một game RPG. Một game nhập vai RPG phải có một hệ thống chiến đấu hoang chỉnh, đây cũng là sự khác biệt lớn nhất giữa các game nhập vai phương Đông (đánh theo lượt) và phương Tây (chiến đấu trong thời gian thực)

- Nhân vật

  Vai trò và kĩ năng cho từng lớp nhân vật trong game là một trong những yếu tố thú vị và lôi cuốn nhất đối với người chơi. Hệ thống xây dựng chỉ số và phát triển nhân vật tạo nên nhiều dạng và phong cách chiến đấu khác nhau. Ban phải sử dụng ,hành động cho nhân vật của mình để kiếm điểm kinh nghiệm và lên cấp để có thêm những điểm sức mạnh, phòng thủ,...

CHƯƠNG II: PHÂN TÍCH YÊU CẦU

1. Yêu cầu của đề tài:

1. Người chơi điều khiển một hoặc nhiều nhân vật trong một bản đồ được lưu trong một cấu trúc dữ liệu tương tự như hình bên, trong đó mỗi ô tương ứng với một dạng bản đồ khác nhau (đất, cỏ,nước…)
2. Trên bản đồ có các quái vật có thể di chuyển được. Các nhân vật người chơi điều khiển.
3. Các nhân vật người chơi và quái vật có các chỉ số xác định tình trạng và thể lực (ví dụ HP, MP, Attack, Defense, Speed…).
4. Người chơi có thể tấn công quái vật và sử dụng các kỹ năng đặc biệt. Tương tự, quái vật cũng có thể tìm đến và tấn công người chơi.
5. Người chơi có thể di chuyển qua lại giữa các bản đồ khác nhau (ví dụ khi đi vào vùng M0, M1, M2… trên bản đồ) hoặc đi đến kết thúc của trò chơi (ví dụ khi đi vào vùng END trên bản đồ).
6. Ghi chú: Sinh viên có thể phát triển thêm các tính năng khác như hệ thống đồ (Inventory system), hệ thống cốt truyện v.v..

M1	1	1	1	2	2	0	0	0	**END**
1	1	1	0	0	0	0	0	0	0
1	3	3	3	0	0	0	0	0	5
1	3	3	3	3	0	0	0	5	5
1	3	3	3	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	0	0	1
M0	0	1	1	1	1	0	4	0	1
1	1	1	1	1	1	1	1	1	1

2. Yêu cầu chức năng tương đương với các yêu cầu của đề bài:

1. Xây dựng bản đồ có kết tập các thành phần, mỗi loại thành phần (bao gồm cả nhân vật và từng thành phần cố định của bản đồ) của bản đồ được biểu diễn bằng một số nguyên cụ thể, thông tin khởi tạo bản đồ được lưu trong một mảng 2 chiều dạng nguyên với mỗi phần tử của mảng chứa một số biểu diễn một loại thành phần của bản đồ.
Mỗi thành phần cố định của bản đồ đều có tính tương tác với nhân vật riêng. Ví dụ: nước chỉ cho phép phù thủy ở bên quái vật đi qua, các viên đá giúp nhân vật di chuyển không tốn năng lượng, bùn đất sẽ làm cho nhân vật mất quyền đi thêm một ô

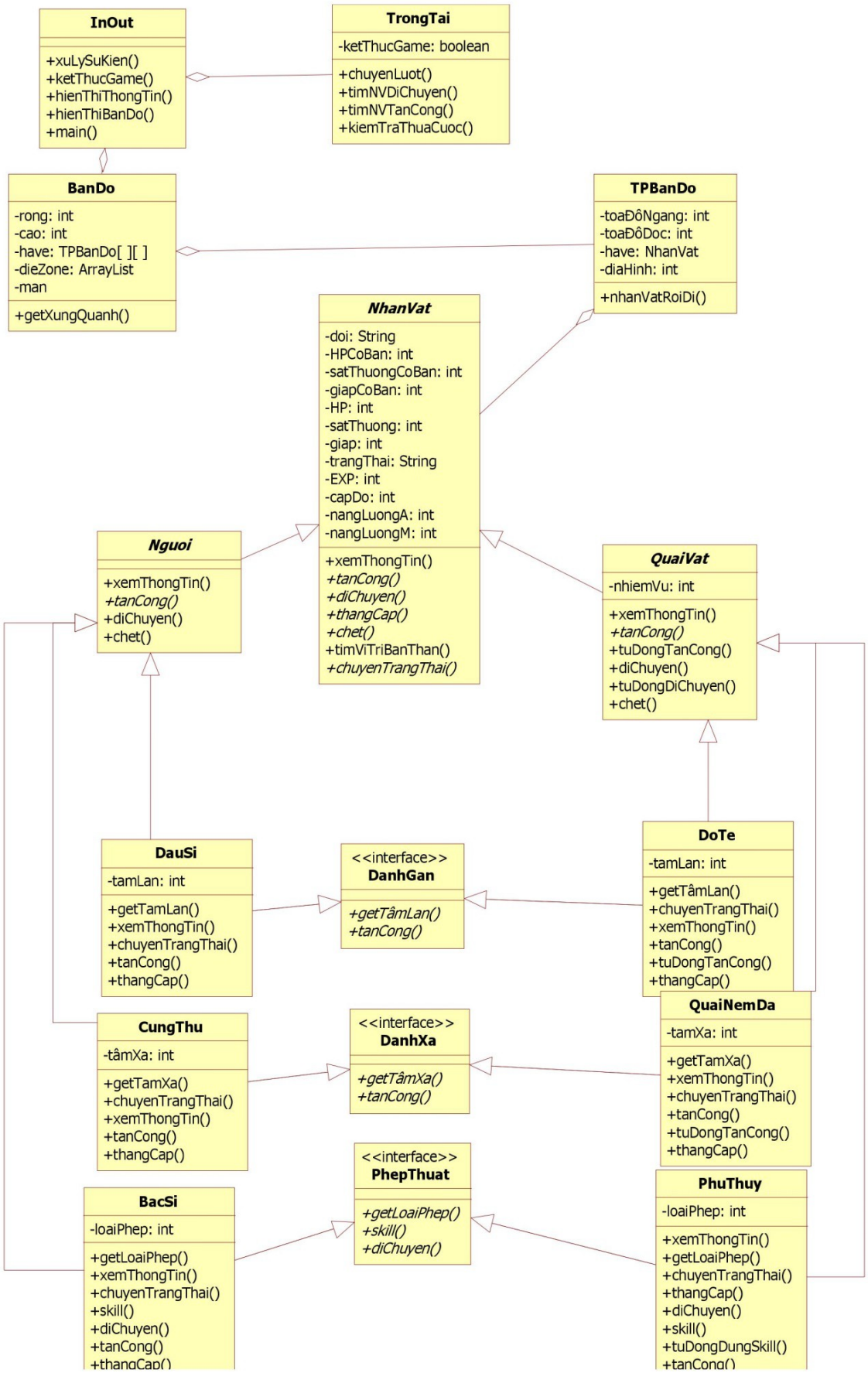
2. Các con quái vật có thể di chuyển tự động. Các nhân vật được người chơi điều khiển thông qua các nút bấm trên màn hình (attack: tấn công quái vật, skill: sử dụng kĩ năng đặc biệt, move: di chuyển trong tầm 8 ô xung quanh, end turn: kết thúc lượt di chuyển của bên người chơi nhường lượt cho bên quái vật).

3. Mỗi nhân vật (bao gồm cả các nhân vật người chơi và quái vật) có các thuộc tính như năng lượng di chuyển (tầm di chuyển của một nhân vật trong mỗi lượt), năng lượng tấn công, sát thương, sát thương cơ bản (khi nhân vật ở cấp độ đầu), lượng máu, lượng máu cơ bản, trạng thái nhân vật, cấp độ nhân vật, kinh nghiệm của nhân vật.

4. Người chơi có thể tấn công quái vật khi quái vật nằm trong tầm tấn công của người chơi và có thể sử dụng kĩ năng đặc biệt để trợ giúp đồng minh (tăng máu)
Khi nhân vật mà người chơi đảm nhận trong tầm quan sát của quái vật, quái vật sẽ tiếp cận con người và tấn công

5.Quy định thắng cuộc của trò chơi là khi nhân vật người chơi đảm nhận đi đến cánh cửa phía đối diện của bản đồ, từ đó sẽ chuyển người chơi sang một bản đồ mới. Thua khi người chơi hết nhân vật để điều khiển.

CHƯƠNG III: BIỂU ĐỒ UML



CHƯƠNG IV: THIẾT KẾ PHƯƠNG THỨC VÀ THUỘC TÍNH CÁC CLASS

1. **Class NhanVat**: lớp trừu tượng mà mọi nhân vật trong chương trình đều được coi là một đối tượng của lớp NhanVat

a. Thuộc tính:

+ *protected String doi:* Đội chơi của nhân vật, game quy định đội đỏ là đội của quái vật và đội đen là đội của người chơi, chỉ nhận 2 giá trị là "den" hoặc "do".

+*protected int HPCoBan:* Lượng máu ở cấp độ thấp nhất của mỗi nhân vật

+ *protected int satThuongCoBan:*Lượng sát thương ở cấp độ thấp nhất của mỗi nhân vật

+ *protected int giapCoBan:* Lượng giáp ở cấp độ thấp nhất của mỗi nhân vật.

+*protected int HP:* Lượng máu ở cấp độ thấp nhật của mỗi nhân vật .

+*protected nt satThuong:*Lượng sát thương thực của nhân vật được sử dụng trong game bằng với lượng sát thương cơ bản nhân với cấp độ nhân vật

+ *protected int giap:* Lượng giáp thực của nhân vật được sử dụng trong game bằng với lượng giáp cơ bản nhân với cấp độ của nhân vật.

+*protected String trangThai:* Trạng thái của nhân vật bao gồm: "kiet suc" và "san sang". Ở trạng thái "kiet suc" nhân vật có năng lượng di chuyển và năng lượng tấn công bằng 0, các trường hợp một trong 2 thông số trên lớn hơn 0 nhân vật ở trạng thái "san sang".

+*protected int EXP:* Kinh nghiệm của nhân vật khi nhân vật đạt được 100 điểm kinh nghiệm sẽ tăng một cấp độ.

+ *protected int capDo:* Dựa vào thuộc tính này để từ các thuộc tính: satThuongCoBan, giapCoBan, HPCoBan có thể tính được satThuong, HP và giap.

+ *protected int nangLuongA:* Mỗi lượt chơi của một phe, các nhân vật sẽ có một lượt tấn công, khi tấn công xong nhân vật phải đợi lượt chơi xong để tấn công. Khi còn có thể tấn công nangLuongA = 1, hết lượt tấn công nangLuongA = 0.

+ *protected int nangLuongM:* Số lần di chuyển của nhân vật đó có thể sử dụng trong một lượt chơi, mỗi lần di chuyển chỉ được di chuyển 1 trong 8 ô xung quanh

b. Constructor:

+ *public NhanVat(int capDo) {}*

Constructor truyền vào tham số capDo để khởi tạo nhân vật với các thuộc tínhsatThuongCoBan, HPCoBan, giapCoBan . Các thuộc tính satThuong, HP, giap được gán phụ thuộc vào satThuongCoBan, HPCoBan, giapCoBan và cấp độ với hệ số ưu tiên (tùy theo đặc tính từng nhân vật) vừa sẽ được thực hiện ở các lớp con cụ thể

c. Phương thức (trừ getter và setter):

Các phương thức abstract được định nghĩa ở các lớp kế thừa từ lớp NhanVat

+ *public abstract String tanCong(int ngang, int doc, BanDo bando):*

+ *public abstract String diChuyen(int ngang, int doc, BanDo bando):*

+ *public abstract void chuyenTrangThai(String tt):*

+ *public abstract void thangCap*

+ *public abstract void chet(BanDo bando):*

+ *public abstract int getLuongHPToiDa():*

+ *public String xemThongTin() {}:* Trả về một chuỗi chứa các thông tin của nhân vật.

+ *public int[] timViTriBanThan(BanDo bando) { }:*Trả về một mảng nguyên gồm 2 phần tử: phần tử đầu chứa tọa độ ngang của nhân vật trên bản đồ, phần tử thứ 2 chứa tọa độ dọc của nhân vật trên bản đồ

d. Getter và Setter

+ *public int getNangLuongA() { }*
+ *public int getNangLuongM() { }*
+ *public void setNangLuongA(int nangLuong) { }*
+ *public void setNangLuongM(int nangLuong) { }*
+ *public int getCapDo() { }*
+ *public int getHPCoBan() { }*
+ *public void setHPCoBan(int HPCoBan) { }*
+ *public int getSatThuongCoBan() { }*
+ *public void setSatThuongCoBan(int satThuongCoBan) { }*
+ *public int getGiapCoBan() { }*
+ *public void setGiapCoBan(int giapCoBan) { }*
+ *public void setCapDo(int capDo) { }*
+ *public int getHP() { }*
+ *public void setHP(int HP) { }*
+ *public int getGiap() { }*
+ *public void setGiap(int giap) { }*
+ *public int getEXP() { }*
+ *public void setEXP(int EXP) { }*
+ *public String getDoi() { }*
+ *public void setDoi(String doi) { }*
+ *public int getSatThuong() { }*
+ *public String getTrangThai() { }*
+ *public void setTrangThai(String tt) { }*
+ *public void setSatThuong(int satThuong) { }*

Trong các phương thức trên có các phương thức setDoi(), setHP(), setTrangThai() được sử dụng để kiểm tra tính hợp lệ của dữ liệu vào.

setDoi() chỉ nhận 2 giá trị "den" và "do".

setHP() chỉ nhận giá trị nhỏ hơn lượng máu tối đa của từng nhân vật cụ thể, nếu tham số truyền vào vượt qua lượng máu tối đa thì HP sẽ được gán giá trị tối đa.

setTrangThai() chỉ nhận 2 giá trị "kiet suc" và "san sang".

2. Class QuaiVat: kế thừa từ lớp NhanVat

a. Constructor:

+ *public Quaivat(int capDo){}*

Phương thức khởi tạo có tham số gọi đến phương thức khởi tạo của lớp cha và gán giá trị của thuộc tính "doi" la "do".

b. Thuộc tính:

Lớp QuaIVat kế thừa các thuộc tính của lớp NhanVat, không thêm thuộc tính nào khác

c. Phương thức :

Các lớp trừu tượng được định nghĩa cụ thể ở các lớp con

+ *public abstract String tanCong(int ngang, int doc, BanDo bando){}*
+ *public abstract void tuDongTanCong(BanDo bando){}*
+ *public abstract int getLuongHPToiDa(){}*

+ *pulic String xemThongTin(){}:* Phương thức override phương thức ở lớp cha, gọi đến phương thức ở lớp cha

+ *public String diChuyen(int ngang, int doc, BanDo bando) {}:* Sau khi tìm được vị trí nhân vật bằng hàm timViTriBanThan() ở lớp cha, phương thức sẽ cập nhật lại bản đồ bằng cách xóa nhân vật ở vị trí cũ và chuyển sang vị trí mới, sau đó tùy vào thuộc tính của ô mới mà nhân vật di chuyển đến sẽ quyết định cách thức trừ năng lượng di chuyển của nhân vật

+ *public void tuDongDiChuyen(BanDo bando){}*: Phương thức giúp quái vật có thể tự động di chuyển, trong 8 ô xung quanh nếu có ô di chuyển được thì quái vật sẽ di chuyển đến, nếu không có ô nào có thể di chuyển được thì năng lượng di chuyển sẽ gán bằng 0

+ *public void chet(BanDo bando) {}:* Khi quái vật chết xóa nhân vật trên tọa độ đó của bản đồ.Xóa đối tượng bằng cách gán giá trị thuộc tính have trong TPBanDo (mà nó đứng trên) về null.

3. Class Nguoi:

a. Constructor:

+ *public Nguoi(int capDo){}:* Phương thức khởi tạo có tham số gọi đến phương thức khởi tạo của lớp cha và gán giá trị của thuộc tính "doi" la "den".

b. Thuộc tính:

Lớp Nguoi kế thừa các thuộc tính của lớp NhanVat, không thêm thuộc tính nào khác

c. Phương thức:

Các lớp trừu tượng được định nghĩa cụ thể ở các lớp con:

+ *public abstract String tanCong(int ngang, int doc, BanDo bando){}*
+ *public abstract int getLuongHPToiDa(){}*

+ *pulic String xemThongTin(){}:* Phương thức override phương thức ở lớp cha, gọi đến phương thức ở lớp cha

+ *public String diChuyen(int ngang, int doc, BanDo bando) {}:* Sau khi tìm được vị trí nhân vật bằng hàm timViTriBanThan() ở lớp cha, phương thức sẽ cập nhật lại bản đồ bằng cách đưa giá trị thuộc tính have ở vị trí cũ và gán nhân vật vào thuộc tính have ở vị trí mới, sau đó tùy vào thuộc tính của ô mới mà nhân vật di chuyển đến sẽ quyết định cách thức trừ năng lượng di chuyển của nhân vật

+ *public void chet(BanDo bando) {}:* Khi chết xóa nhân vật trên tọa độ đó của bản đồ.Xóa đối tượng bằng cách gán giá trị thuộc tính have trong TPBanDo (mà nó đứng trên) về null. Đồng thời trỏ dieZone của bản đồ vào nó.,nhằm tạo điều kiện để sau này có thể hồi sinh nhân vật (khi cần)

4. Interface DanhGan

Bao gồm 2 phương thức trừu tượng: sẽ được cài đặt chi tiết trong 2 lớp con DauSi và DoTe

+ *public abstract int getTamLan();*
+ *public abstract String tanCong(int ngang,int doc,BanDo bando) ;*

5. Class DauSi

a. Thuộc tính

Ngoài các thuộc tính được kế thừa từ lớp Nguoi, thì có thêm :

+ *private int tamLan;*

Tầm đánh lan của nhân vật khi nó thực hiện tấn công. Những kẻ thù trong phạm vi tâm lan sẽ bị tấn công.

b. Phương thức (được ghi đè từ class Nguoi và interface DanhGan)

+ *public DauSi(int capDo) {}* Khởi tạo DauSi

+ *public void chuyenTrangThai(String trangThai) {}*

Gọi đến khi chuyển lượt, mỗi trạng thái có 1 năng lượng A, M khác nhau tùy vào loại nhân vật.
Mỗi lượt chơi của một phe, các nhân vật sẽ có một lượt tấn công, khi tấn công xong nhân vật phải đợi lượt chơi mới để tấn công. Khi còn có thể tấn công nangLuongA = 1, hết lượt tấn công nangLuongA = 0.
Số lần di chuyển của nhân vật đó có thể sử dụng trong một lượt chơi, mỗi lần di chuyển chỉ được di chuyển 1 trong 8 ô xung quanh.

+ *public String xemThongTin() {}*

Cho ta các thông tin của DauSi bao gồn cả tamLan của DauSi

+ *public int getTamLan() {}*

Hiện thực hóa từ interface DanhGan, trả về phạm vi ảnh hưởng của kẻ thù khi nó nằm trong khoảng tâm lan của DauSi.

+ *public String tanCong(int ngang,int doc,BanDo bando) {}*

Giảm máu của địch được chọn theo sát thương của nhân vật và giáp của địch,những kẻ thù trong phạm vi tâm lan sẽbị tấn công..Với tham số truyền vào là tọa độ cần tấn công của bản đồ và BanDo(dùng để xác định vị trí nhân vật).

+ *public void thangCap() {}*

Thăng cấp cho nhân vật khi đủ EXP (>=100) cấp độ được tăng lên 1 đơn vị. Từ đó HP, SatThuong,Giap sẽ được tăng lên tương ứng.

+ *public int getLuongHPToiDa() {}*

Trả về lương HP tối đa mà DauSi có thể có tương ứng với cấp độ này.

6. Class DoTe

a. Thuộc tính

Ngoài các thuộc tính được kế thừa từ lớp QuaiVat, thì có thêm :

+ *private int tamLan;*

b. Phương thức

Bao gồm các phương thức được kế thừa từ lớp QuaiVat và DanhGan, tuy nhiên DoTe có phương thức tuDongTanCong.

- Các phương thức không đổi:

  + *public DoTe(int capDo) {}* → khởi tạo
  + *public void chuyenTrangThai(String trangThai) {}* → chuyển trạng thái (di chuyển ,tấn công)
  + *public String xemThongTin() {}* → xem các thông tin của DoTe
  + *public int getTamLan() {}*
  + *public void thangCap() {}*
  + *public int getLuongHPToiDa() {}*

- Phương thức tự động tấn công:

  public void tuDongTanCong(BanDo bando) {}

  Là phương thức giúp nhân vật tự động tấn công.Quét các ô trong tầm di chuyển(8 ô xung quanh) :
  nếu có các ô tấn công được thì tấn công ngẫu nhiên đến 1 trong các ô đó.giảm năng lượng a đi 1 +exp
  nếu không có ô tấn công được thì đưa năng lượng a về 0.

7. Interface DanhXa :

+ *public abstract int getTamXa();*
+ *public abstract String tanCong(int ngang,int doc,BanDo bando) ;*

Phương thức abstract sẽ được cài đặt chi tiết trong class implement interface này (bao gồm CungThu và QuaiNemDa)

8. Class CungThu :

a. Thuộc Tính

Ngoài các thuộc tính được kế thừa từ lớp Nguoi, thì có thêm :

+ *private int tamXa;* Tầm xa tấn công của nhân vật.

b. Phương thức (được ghi đè từ class Nguoi và interface DanhXa)

+ *public CungThu(int capDo) {} ;* Truyền vào tham số capDo để khởi tạo nhân vật với cấp độ truyền vào.
+ *public void chuyenTrangThai(String trangThai) {};* tương tự như chuyển trạng thái của Đấu Sĩ
+ *public String xemThongTin() {};* Cho ta các thông tin của DauSi bao gồn cả tamXa cua CungThu
+ *public int getTamXa(){};* Hiện thực hóa từ interface DanhXa, trả về phạm vi ảnh hưởng của kẻ thù khi nó nằm trong khoảng tâm xa tấn công của CungThu
+ *public String tanCong(int ngang,int doc,BanDo bando) {};* Giảm máu của địch được chọn theo sát thương của nhân vật và giáp của địch .
+ *public void thangCap() {};*Thăng cấp cho nhân vật khi đủ EXP (>=100) cấp độ được tăng lên 1 đơn vị. Từ đó HP, SatThuong,Giap sẽ được tăng lên tương ứng.
+ *public int getLuongHPToiDa() {};* Trả về lương HP tối đa mà DauSi có thể có tương ứng với cấp độ này.
+ *public void chuyenTrangThai(String trangThai) {}*

được gọi đến khi chuyển lượt, mỗi trạng thái có 1 năng lượng A, M khác nhau tùy vào nhân vật.

9. Class QuaiNemDa :

a. Thuộc Tính:

+ *private int tamXa;* Tầm xa tấn công của nhân vật.

b. Phương thức :

+ *public QuaiNemDa (int capDo) {} ;* Truyền vào tham số capDo để khởi tạo nhân vật với cấp độ truyền vào.
+ *public void chuyenTrangThai(String trangThai) {};* Gọi đến khi chuyển lượt, mỗi trạng thái có 1 năng lượng A, M khác nhau do vào đặc tính của QuaiNemDa quyết định
+ *public String xemThongTin() {}; Cho ta các thông tin của DauSi bao gồn cả tamXa cua QuaiNemDa*
+ *public int getTamXa() {};* Trả về tầm xa tấn công của nhân vật.
+ *public String tanCong(int ngang,int doc,BanDo bando) {};* Giảm máu của địch được chọn theo sát thương của nhân vật và giáp của địch .
+ *public void tuDongTanCong(BanDo bando) {}; Là phương thức giúp nhân vật tự động tấn công.Quét các ô trong tầm xa tấn công* nếu có các ô tấn công được thì tấn công ngẫu nhiên đến 1 trong các ô đó.giảm năng lượng a đi 1 và +exp

nếu không có ô tấn công được thì đưa năng lượng a về 0.

+*public void thangCap() {};* Tương tự CungThu
+ *public int getLuongHPToiDa() {};* Trả về lượng máu tối đa của nhân vật tùy theo cấp độ

10. Interface PhepThuat :

+ public String getLoaiPhep();
+ public String skill(int ngang, int doc, BanDo bando);
+ public abstract String diChuyen(int ngang, int doc, BanDo bando)

Phương thức abstract sẽ được cài đặt chi tiết trong class implement interface này (bao gồm BacSi và PhuThuy)

11 . Class BacSi :

a.Thuộc Tính:

+ private String loaiPhep; Loại phép sử dụng của nhân vật .(ở đây chỉ cài đặt 1 loại phép là hồi máu)
+ private int tamSD; Tầm sử dụng của phép.
+private int luongMauHoi; Lượng máu hồi lại cho đồng minh .

b.Phương thức :

+ public BacSi(int capDo) {}; Constructor truyền vào tham số capDo để khởi tạo nhân vật với cấp độ truyền vào.

+ public void chuyenTrangThai(String trangThai) {}; Được gọi đến khi chuyển lượt.mỗi trạng thái có 1 năng lượng A, M khác nhau tùy vào nhân vật.

+ public String xemThongTin() {}; Trả về một chuỗi chứa các thông tin của nhân vật.
+ public String getLoaiPhep() {}; Trả về loại phép của nhân vật.
+ public String diChuyen(int ngang, int doc, BanDo bando) {}; ghi đè lại phương thức di chuyển của Nguoi, để có thể di chuyển trên mặt nước, bằng cách lấy địa hình nước là một trong các nơi di chuyển tới được
+ public String tanCong(int ngang, int doc, BanDo bando){}; Bác Sĩ không thể tấn công, chỉ trả về thông báo khi gọi đến
+ public String skill(int ngang, int doc, BanDo bando) {}; Tăng máu cho đồng minh được chọn theo luongMauHoi., truyền vào tọa độ ngang, dọc của đồng minh đã được người chơi click

+ public void thangCap() {}; Thăng cấp cho nhân vật khi đủ EXP (>=100) cấp độ được tăng lên 1 đơn vị. Từ đó HP, SatThuong,Giap sẽ được tăng lên tương ứng.
+ public int getLuongHPToiDa() {}; Trả về lượng máu tối đa của nhân vật tùy theo cấp độ

12 . Class PhuThuy :

a. Thông tin :

+ private String loaiPhep; Loại phép sử dụng của nhân vật .
+ private int tamSD; Tầm sử dụng của phép.
+ private int luongMauHoi; Lượng máu hồi lại cho đồng minh.

b. Phương thức :

+ public BacSi(int capDo) {}; Constructor truyền vào tham số capDo để khởi tạo nhân vật theo cấp độ truyền vào.
+ public void chuyenTrangThai(String trangThai) {}; Được gọi đến khi chuyển lượt(xem TrongTai). mỗi trạng thái có 1 năng lượng A, M khác nhau tùy vào nhân vật.

+ public void tuDongDungSkill(BanDo bando) {};
+ public String xemThongTin() {}; Trả về một chuỗi chứa các thông tin của nhân vật.
+ public String getLoaiPhep() {}; Trả về loại phép của nhân vật.
+ public String diChuyen(int ngang, int doc, BanDo bando) {}; Thay đổi tọa độ của nhân vật trên bản đồ.
+ public String tanCong(int ngang, int doc, BanDo bando){}; Phù thủy không thể tấn công, chỉ trả về thông báo khi gọi đến
+ public String skill(int ngang, int doc, BanDo bando) {}; Tăng máu cho đồng minh được chọn theo luongMauHoi.
+ public void tuDongDiChuyen(BanDo bando) {}; Ghi đè lại phương thức di chuyển của QuaiVat sao cho nó có khả năng đi trên mặt nước
+ public void tuDongTanCong(BanDo bando) {}; Phù thủy không thể tấn công.
+public void thangCap() {}; tương tự BacSi
+ public int getLuongHPToiDa() {}; Trả về lượng máu tối đa của nhân vật tùy theo cấp độ

13. Class InOut:

Nhiệm vụ: kiểm soát dữ liệu vào, xuất dữ liệu ra màn hình.
điều khiển luồng chương trình.

a. Thuộc tính:

private JLabel[ ][ ] o :	Hiển thị hình ảnh các ô của game
private int clickROW ;	Dùng trong quá trình bắt sự kiện
private int clickCOL ;	
private int clickROWT ;	
private int clickCOLT ;	
private String clickButton ;	
private boolean sukien;	
private ArrayList dataCacBanDo;	Lưu dữ liệu khởi tạo của tất cả các màn
private int soLuongMan;	Lưu số lượng màn
private BanDo bando;	Kết tập bando
private TrongTai trongtai;	Kết tập trongtai

b. Constructor:
Public InOut(ArrayList dataCacBanDo,int manchoi,TrongTai trongtai){}
truyền vào các dữ liệu ban đầu của game (từ trong main)
khởi tạo bando từ dataCacBanDo và manchoi bắt đầu (màn 1)

c . Phương thức:
+ *private void xuLySuKien(){}* khi các điều kiện thông tin của 1 sự kiện được bắt đầy đủ. Hàm này được gọi tới.
trong đó đọc:
clickROW, click COL: tọa độ đối tượng nguồn tác động
clickROW, click COL: tọa độ đối tượng đích bị tác động
clickButton: hành động mà đối tượng nguồn tác động lên đối tượng đích.
để gọi phương thức tương ứng của đối tượng tác động

+ *public void hienThiBanDo(){}* đọc bando và hiển thị toàn bộ hình ảnh game(trong các label)
được gọi đến sau mỗi hành động của nhân vật.
+ *public void hienThiThongtin(){}* đọc bando và hiển thị thông tin nhân vật dưới dạng TipTex của các label
+ *private void ketThucGame() {}* hiện thông báo kết thúc. Thoát chương trình.

b. Getter và Setter

Các dữ liệu trong InOut không nên bị thay đổi trong class khác. Nên không có các phương thức get, set ở đây.

14. Class TrongTai:

Nhiệm vụ:
xem xét điều kiện thắng thua của game.

a. Thuộc tính:
+ *private boolean ketThucGame;*

b. Constructor:

+ *public TrongTai(){}*: khởi tạo ban đầu ketThucGame bằng false

c. Phương thức:

+ public void chuyenLuot(String doiHienTai, String doiNhanLuot,BanDo bando){} :nạp năng lượng cho cách nhân vật đội nhận lượt rút năng lượng về 0 đối với nhân vật thuộc đội hiện tại

+ *public NhanVat timNVDiChuyen(String doiHienTai, BanDo bando){}*

+ *public NhanVat timTanCong(String doiHienTai, BanDo bando){}*

tìm nhân vật còn khả năng di chuyển, tấn công của đội hiện tại trả về NhanVat đầu tiên tìm được

+ *public boolean kiemTraThuaCuoc(String doiHienTai, Bando bando){}*

kiểm tra xem trong bản đồ còn nhân vật của doiHienTai không, nếu còn thì đội hiện tại chưa thua (ketThucGame =false) và ngược lại.

d. Getter và Setter

+ *public boolean* getKetThucGame(){}.

Trả về game kết thúc hay chưa.

15. Class Bando:

1. Thuộc tính:

+ *private int man* : thông tin bản đồ của màn mấy
+ *private int rong* : chiều cao của bàn đồ
+ *private int cao* : chiều rộng của bản đồ
+ *private TPBanDo[][] have* : mảng have lưu thành phần của bản đồ
+ *private ArrayList dieZone* : ArrayList dieZone lưu các nhân vật đã chết trong màn chơi

2. Khởi tạo:

+ *public BanDo(int manchoi,ArrayList dataCacBanDo)* : truyền vào tham số màn chơi và dữ liệu bản đồ

3. Phương thức:

+ *public ArrayList getXungQuanh(int ngang,int doc,int xa)* : truyền vào tọa độ đang xét, trả về ArrayList<TPBanDo> là các ô xung quanh ô đang xét

4. Getter & Setter:

+ *public int getMan()* : trả về giá trị của màn chơi
+ *public void setMan(int manmoi)* : gán giá trị màn chơi
+ *public ArrayList getdieZone()* : trả về các nhân vật đã chết trong màn chơi
+ *public int getRong()* : trả về chiều rộng màn màn chơi
+ *public void setRong()* : gán chiều rồng màn chơi
+ *public int getCao()* : trả về chiều cao màn chơi
+ *public void setCao()* : gán chiều cao màn chơi
+ *public TPBanDo getHave(int row,int col)* : trả về thành phần bản đồ

16. Class TPBanDo:

1. Thuộc tính:

+ *private int toaDoNgang* : tọa độ ngang của ô
+ *private int toaDoDoc* : tọa độ dọc của ô
+ *private NhanVat have* : nhân vật trong ô
+ *private String diaHinh* : địa hình của ô

2. Khởi tạo:

   + *public TPBanDo(int indexRow, int indexCol, int numberDiaHinh, int numberNhanVat,int manchoi) :* truyền vào tham số về hàng, côt, nhân vật, địa hình trong ô và màn chơi
3. Phương thức:

   + *public void nhanVatRoiDi():* Chỉ số Have của ô trỏ về null sau khi nhân vật rời đi
4. Getter & Setter:

   + *public String getDiaHinh :* trả về loại địa hình của ô
   + *public NhanVat getHave :* trả về nhân vật trong ô
   + *public int getToaDoNgang() :* trả về tọa độ ngang ô
   + *public int getToaDoDoc() :* trả về tọa đọ dọc của ô
   + *public void setDiaHinh() :* gán địa hình cho ô
   + *public void setHave() :* gán nhân vật cho ô

CHƯƠNG V: CÁC KĨ THUẬT OOP ĐƯỢC SỬ DỤNG

1. Kế thừa, kết tập:

Như đã thể hiện trong UML:

Class Nguoi, QuaiVat kế thừa từ class NhanVat

Class DauSi, CungThu, BacSi kế thừa từ class Nguoi

Class Dote, QuaiNemDa, PhuThuy kế thừa từ class QuaiVat

NhanVat là một phần của TPBanDo (thành phần bản đồ)

TPBanDo là một phần của BanDo

BanDo, TrongTai là một phần trong các đối tượng trao đổi dữ liệu với người dùng, nên cũng kết tập trong InOut

2. Override:

Các phương thức abstract ở lớp cha trừu tượng đều được hiện thực ở lớp con cụ thể.

PhuThuy, BacSi do có khả năng đi trên mặt nước nên ghi đè lại phương thức diChuyen của lớp cha

Phương thức tanCong của mỗi lớp con là khác nhau (có nhân vật đánh lan, đánh xa) nên cũng hiện thực cách tấn công của riêng mình.

3. Overload:

Ở class PhuThuy:

có 2 phương thức trùng tên:

(a)public String skill(int ngang, int doc, BanDo bando) ;

(b)public void skill(BanDo bando);

Phương thức a , b cùng thực hiện hồi máu cho đồng minh. Nhưng ở a, tọa độ ngang dọc của điểm đích đã được truyền vào rõ ràng, nhân vật hồi máu cho đích.

Còn ở b, Nhân vật sẽ tự động tìm xung quanh xem có nhân vật nào máu thấp và dùng skill.

4. Đa Hình:

Khi in thông tin chỉ số , InOut xem các loại nhân vật như class cha NhanVat

Nhưng khi di chuyển và tấn công. InOut phải Dow-casting xuống QuaiVat để gọi được phương thức tấn công tự động và di chuyển tự động mà chỉ riêng QuaiVat mới có.

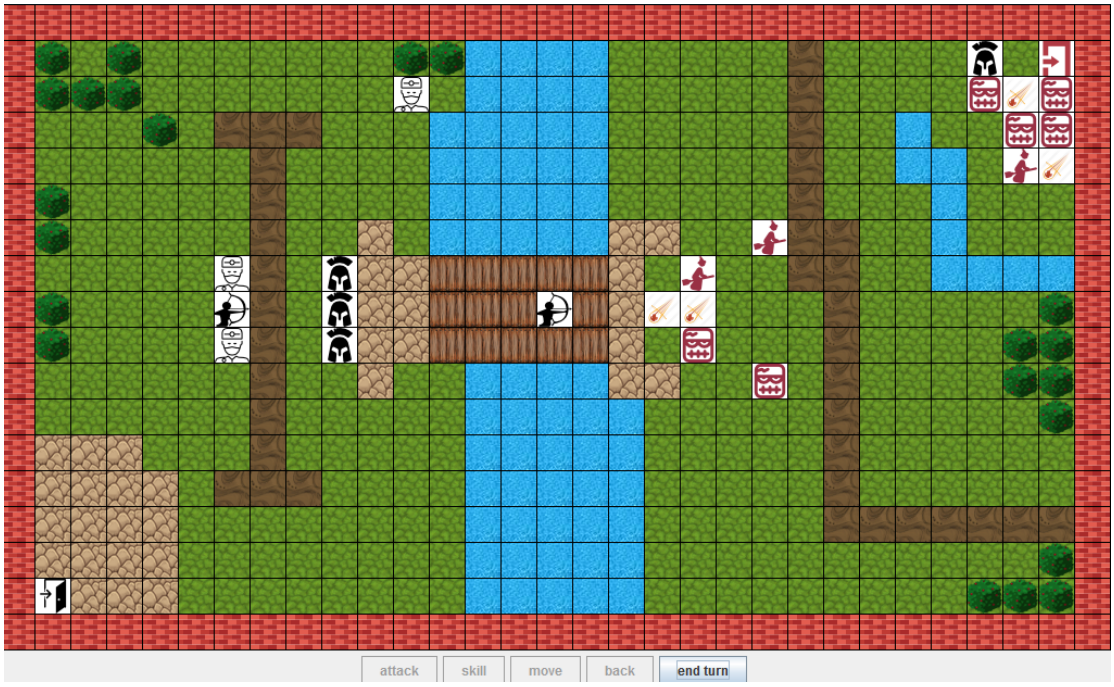
5.Che giấu và toàn vẹn giữ liệu:

Các thuộc tính đều là Protected . dùng get để truy cập dữ liệu. kiểm tra tính hợp lệ dữ liệu trước khi set.

CHƯƠNG VI: KẾT QUẢ

1. Cách thức hoạt động, chức năng của chương trình:

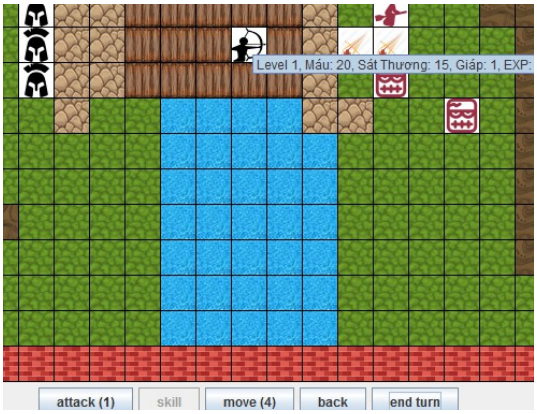
Phần giao diện chương trình:

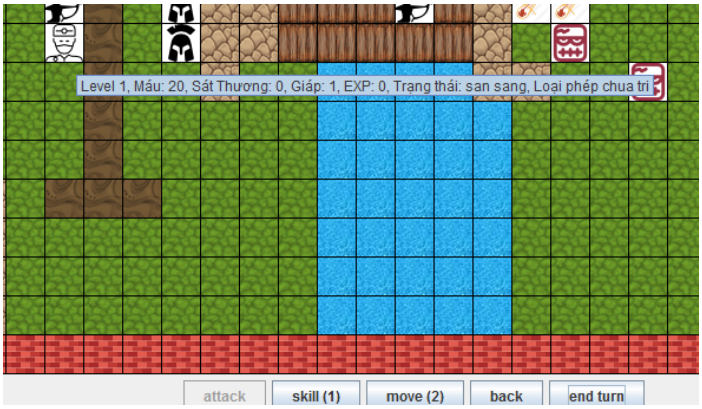


Người chơi điều khiển bên nhân vật màu đen với mục tiêu đi đến cửa của đội màu đỏ vượt qua sự cản trở của nhân vật đội đỏ, khi đến của màu đỏ sẽ đưa người chơi đến bàn mới hoặc kết thúc chương trình.

Xem thông tin nhân vật bằng cách trỏ chuột vào nhân vật cần xem thông tin.

Trong mỗi lượt chơi của mỗi bên, mỗi nhân vật có một lượt tấn công và một số lượt di chuyển tùy thuộc vào thuộc tính của từng nhân vật. Người chơi chọn nhân vật bằng cách nhấn vào nhân vật đó và các phím ở phía dưới màn hình tương ứng với các hành động mà nhân vật đó có thể thực hiện





2. Hướng phát triển của chương trình

- Hồi sinh nhân vật trong từng ván
- Tạo ra hệ thống các đồ vật của nhân vật, các loại cửa hàng và tiền
- Có thể tương tác thêm qua bàn phím, giảm bớt thời gian tương tác với chương trình
- Viết cốt truyện của game giúp game thu hút hơn
- Làm thêm nhiều animation tấn công và chuyển động cho mỗi nhân vật

3. Phân công công việc:

Công việc lập trình

- **Nguyễn Trần Nam:** các lớp xây dựng chính: NhanVat, QuaiVat, Nguoi.
- **Nguyễn Văn Túc:** các lớp xây dựng chính: DauSi, DanhGan, DoTe, PhepThuat
- **Nguyễn Văn Đức:** các lớp xây dựng chính: BanDo, TPBanDo.
- **Ngô Quang Hòa:** các lớp xây dựng chính: InOut, TrongTai.

- **Hoàng Ngọc Dũng:** các lớp xây dựng: CungThu, DanhXa, QuaiNemDa, BacSi, Phu Thuy

Công việc khác:

- **Nguyễn Trần Nam:** tạo bản đồ, tìm kiếm hình ảnh, viết báo cáo
- **Nguyễn Văn Túc:** tìm kiếm thiết kế các thuộc tính của các nhân vật trong game
- **Nguyễn Văn Đức:** kiểm thử game
- **Ngô Quang Hòa:** tạo bản đồ, thiết kê UML, thiết kế luồng
- **Hoàng Ngọc Dũng:** kiểm thử game