ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐỒ ÁN MÔN HỌC

Lập trình hướng đối tượng

Tên đề tài: Turn-Based RPG

Giảng viên hướng dẫn: PGS.TS. Lê Đfíc Hậu

Nhóm sinh viên thực hiện:

- Nguyễn Lưu Tùng Quân - 20225757
- Nguyễn Trí Nam - 20226093

HÀ NỘI, 12/2024

Contents

1 Giới thiệu đề tài **3**
1.1 Đặt vấn đề 3
1.2 Mục tiêu và phạm vi đề tài 4
1.2.1 Mục tiêu 4
1.2.2 Phạm vi 4

2 Khảo sát và phân tích yêu cầu **5**
2.1 Khảo sát hiện trạng 5
2.1.1 Đối tượng sử dụng 5
2.1.2 Hệ thống hiện có 5
2.1.3 Hệ thống tự phát triển 5
2.2 Tổng quan chức năng 5

3 Công nghệ sử dụng **7**
3.1 Java 7

4 Phân tích thiết kế hệ thống **8**
4.1 Thiết kế gói 8
4.1.1 Các Package 8
4.1.2 Mối quan hệ giữa các Package 8

5 Phân tích đặc tính hướng đối tượng **9**
5.1 Tính đóng gói 9
5.2 Tính kế thừa 9
5.3 Tính trừu tượng 10
5.4 Tính đa hình 10

6 Xây dựng chương trình minh hoạ **11**
6.1 Kết quả đạt được 11
6.2 Minh hoạ các chức năng chính 11
6.2.1 Quản lý bản đồ và ô (Tile) 11

6.2.2 Quản lý nhân vật và quái vật 11
6.2.3 Hệ thống vật phẩm (Item) 11
6.2.4 Quản lý giai đoạn chơi (GameStage) 12
6.2.5 Hệ thống tương tác 12
6.2.6 Giao diện và hiển thị 12
6.2.7 Cơ chế lưu và tải 12
6.2.8 Hệ thống khởi chạy (App) 12

1 Giới thiệu đề tài

1.1 Đặt vấn đề

Trong lĩnh vực trò chơi điện tử, thể loại Turn-Based RPG (Role-Playing Game) luôn giữ một vị trí đặc biệt trong lòng người chơi nhờ sự kết hợp giữa yếu tố chiến thuật, cốt truyện sâu sắc, và tính tương tác cao. Thể loại này không chỉ mang đến những giờ phút giải trí mà còn khơi gợi khả năng tư duy, phân tích tình huống, và sáng tạo trong cách xây dựng chiến lược.

Tuy nhiên, với sự phát triển không ngừng của ngành công nghiệp trò chơi, các tựa game Turn-Based RPG hiện nay đang phải đối mặt với nhiều thách thức. Người chơi ngày càng có yêu cầu cao hơn về chất lượng đồ họa, tính đa dạng trong lối chơi, cũng như sự hấp dẫn của cốt truyện. Đặc biệt, việc giữ được sự cân bằng giữa yếu tố truyền thống và đổi mới trong thiết kế trò chơi là một bài toán khó đối với các nhà phát triển. Một trò chơi quá bám sát khuôn mẫu cũ có thể gây nhàm chán, trong khi việc thay đổi quá nhiều có nguy cơ làm mất đi bản sắc vốn có của thể loại này.

Trong bối cảnh đó, nghiên cứu và phát triển một tựa game Turn-Based RPG không chỉ dừng lại ở việc tạo ra một trò chơi mới, mà còn cần đảm bảo mang đến trải nghiệm hấp dẫn, phù hợp với nhu cầu và xu hướng hiện đại. Đồng thời, sản phẩm phải giữ được sự lôi cuốn từ cốt truyện, sự tinh tế trong chiến thuật, và khả năng tương tác cao – những yếu tố làm nên sức hút đặc trưng của dòng game này.

Với những lý do trên, việc tìm hiểu và xây dựng một tựa game Turn- Based RPG mới không chỉ mang tính ứng dụng cao mà còn góp phần làm phong phú thêm thị trường trò chơi điện tử, đáp ứng nhu cầu ngày càng đa dạng của cộng đồng game thủ.

1.2 Mục tiêu và phạm vi đề tài

1.2.1 Mục tiêu

Xây dựng một trò chơi Turn-Based RPG với các chức năng chính, bao gồm cơ chế chiến đấu theo lượt, nhặt vật phẩm, đánh boss.

1.2.2 Phạm vi

Dự án bao gồm việc phát triển một hệ thống Turn-Based RPG với các chức năng sau:

- Hệ thống chiến đấu theo lượt: Thiết kế cơ chế chiến đấu chiến thuật, nơi người chơi phải đưa ra các quyết định chiến lược dựa trên tài nguyên và tình huống cụ thể trong mỗi lượt.
- Hỗ trợ lưu trữ và truy xuất tiến trình: Đảm bảo hệ thống lưu và tải game hoạt động ổn định, cho phép người chơi dễ dàng tiếp tục hành trình ở bất kỳ thời điểm nào.

2 Khảo sát và phân tích yêu cầu

2.1 Khảo sát hiện trạng

2.1.1 Đối tượng sử dụng

- Đối tượng: Người yêu thích thể loại Turn-Based RPG, bao gồm game thủ cá nhân từ 15 đến 35 tuổi, cả người chơi mới và kỳ cựu.

2.1.2 Hệ thống hiện có

- Lối chơi thiếu sáng tạo, dễ gây nhàm chán cho người chơi lâu năm.
- Ít tùy chọn ảnh hưởng đến diễn biến và kết cục của trò chơi.

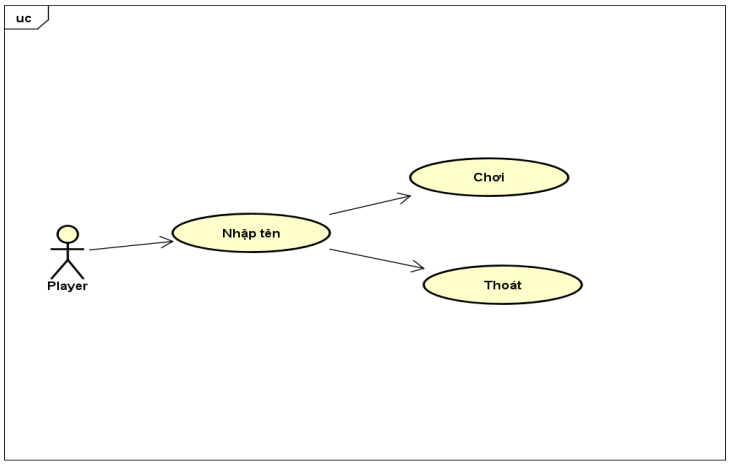
2.1.3 Hệ thống tự phát triển

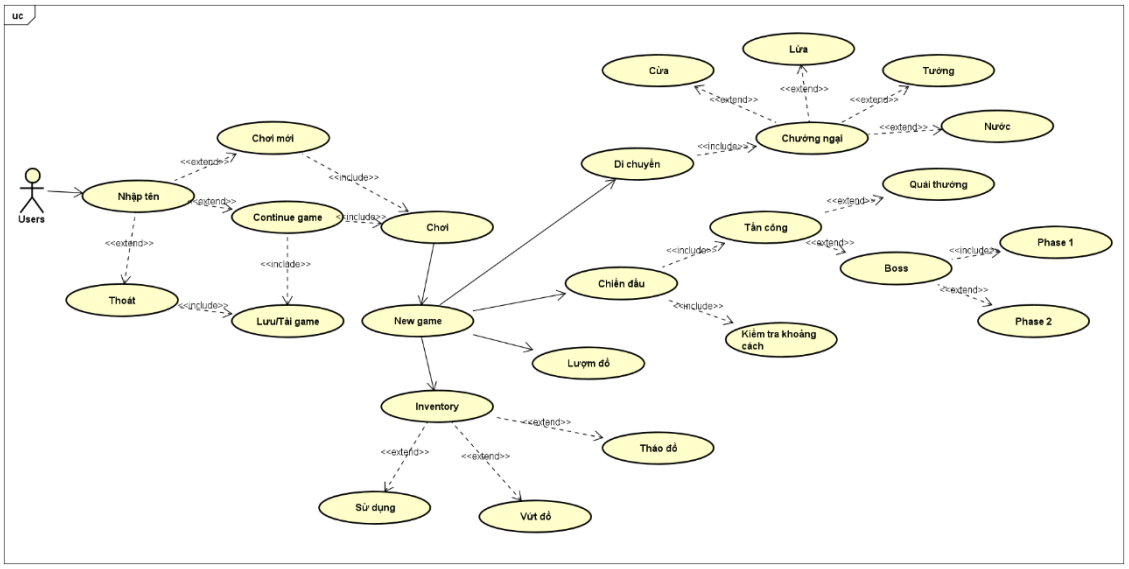
- Quản lý thông tin nhân vật trực quan: Cung cấp các tính năng dễ dàng xem, chỉnh sửa, hoặc cập nhật thông tin nhân vật, bao gồm chỉ số, kỹ năng và trang bị.
- Hệ thống chiến đấu linh hoạt: Thiết kế cơ chế chiến đấu theo lượt dễ tiếp cận, đồng thời cho phép tùy chỉnh chiến thuật đa dạng.
- Lưu trữ tiến trình lâu dài: Cho phép lưu và tải trò chơi nhanh chóng, hỗ trợ người chơi tiếp tục trải nghiệm bất cứ lúc nào.

2.2 Tổng quan chức năng

- Nhập tên: Người chơi nhập tên nhân vật của mình để bắt đầu trò chơi.

- Chơi: Sau khi nhập tên, người chơi chọn bắt đầu trải nghiệm trò chơi.
- Thoát: Người chơi có thể chọn thoát khỏi trò chơi.





3 Công nghệ sử dụng

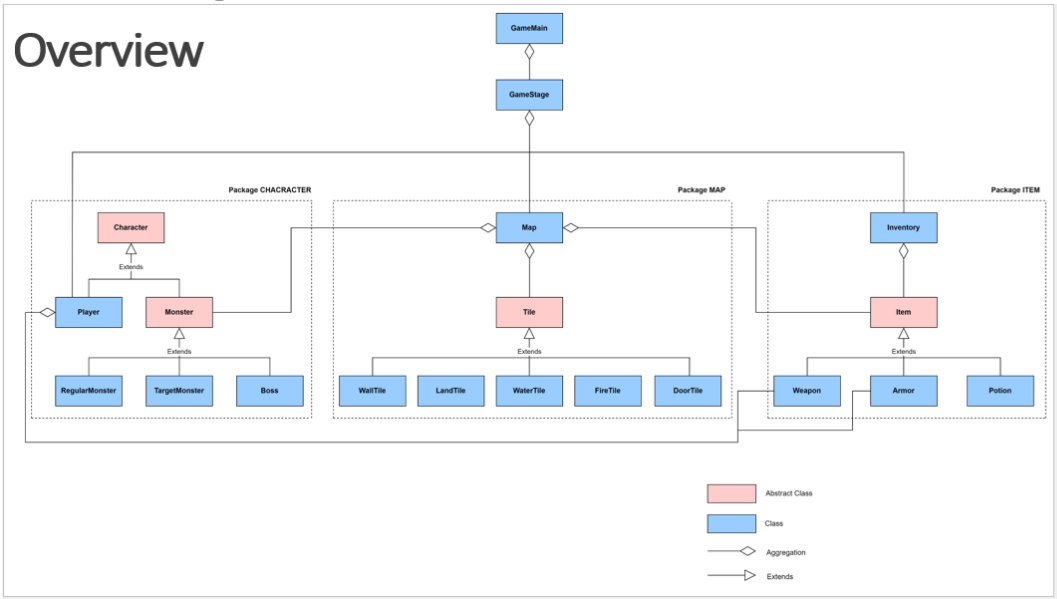
3.1 Java

Java là một ngôn ngữ lập trình hướng đối tượng, mạnh mẽ và phổ biến, được thiết kế để chạy trên nhiều nền tảng khác nhau (Write once, run anywhere), và được sử dụng rộng rãi trong phát triển ứng dụng web, ứng dụng di động, phần mềm doanh nghiệp và nhiều hơn nữa.

- Đơn giản và dễ học: Cú pháp rõ ràng, dễ hiểu.
- Hướng đối tượng: Tổ chức code hiệu quả, dễ bảo trì.
- An toàn: Cơ chế quản lý bộ nhớ tự động, giảm thiểu lỗi.
- Đa nền tảng: Chạy trên nhiều hệ điều hành khác nhau.
- Cộng đồng lớn: Có nhiều tài liệu, thư viện hỗ trợ.

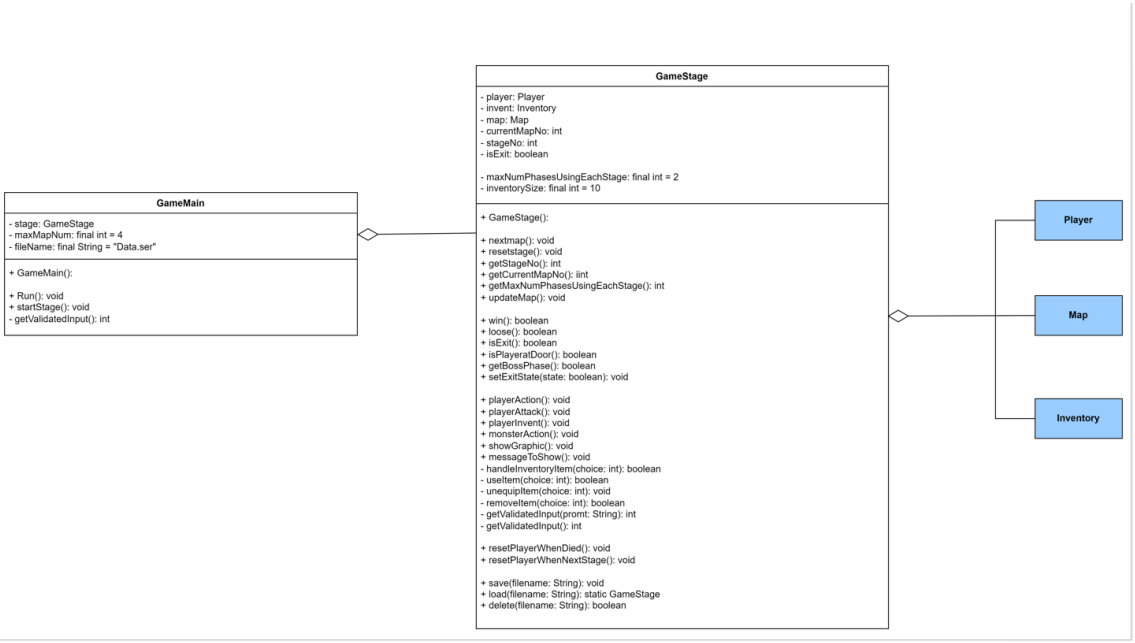
4 Phân tích thiết kế hệ thống

4.1 Thiết kế gói

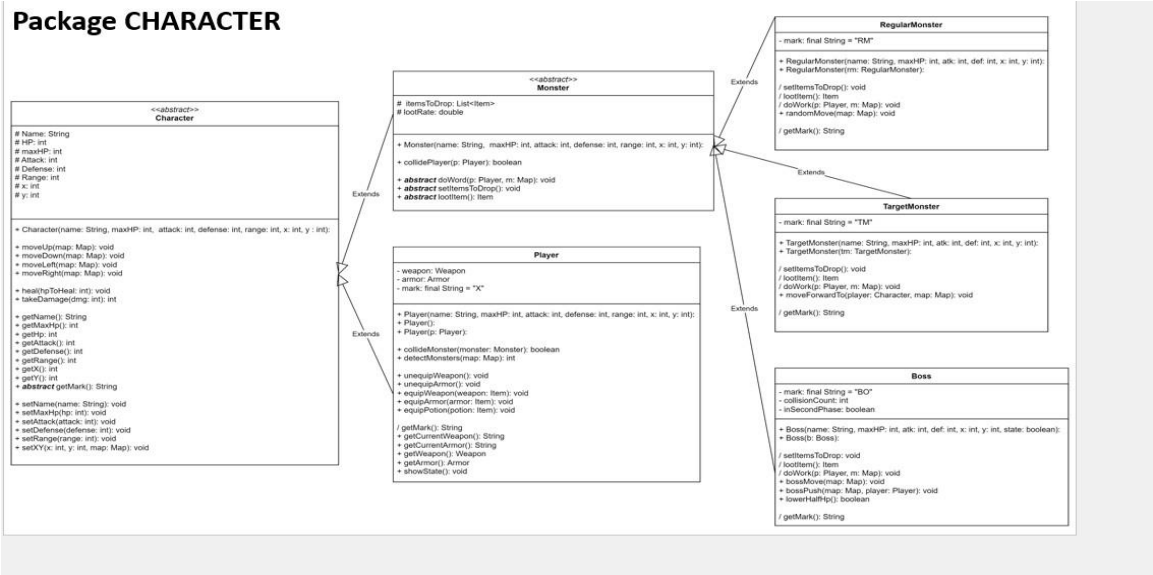


4.1.1 Các Package

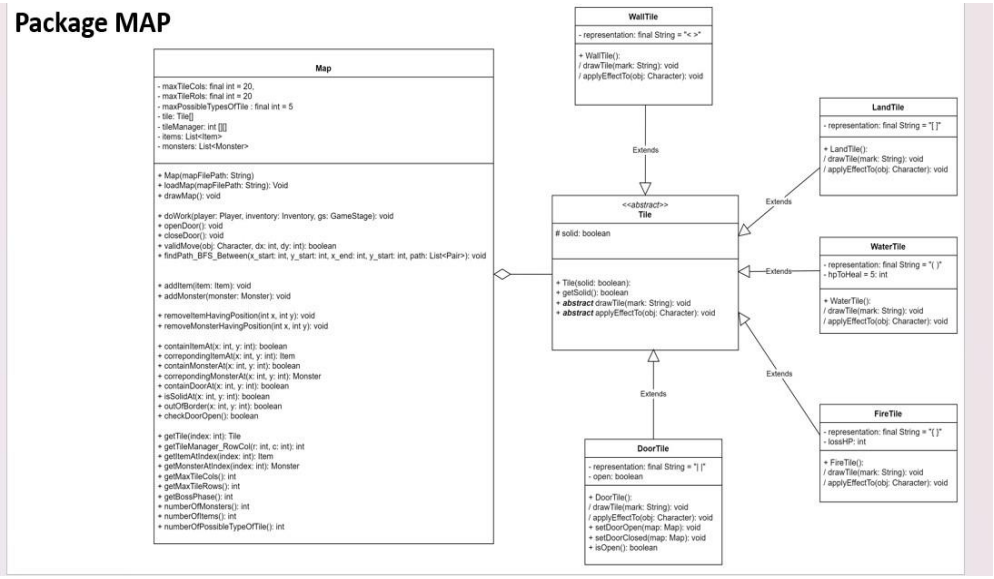
- **GameMain**: Chứa lớp chính quản lý trò chơi và các giai đoạn.



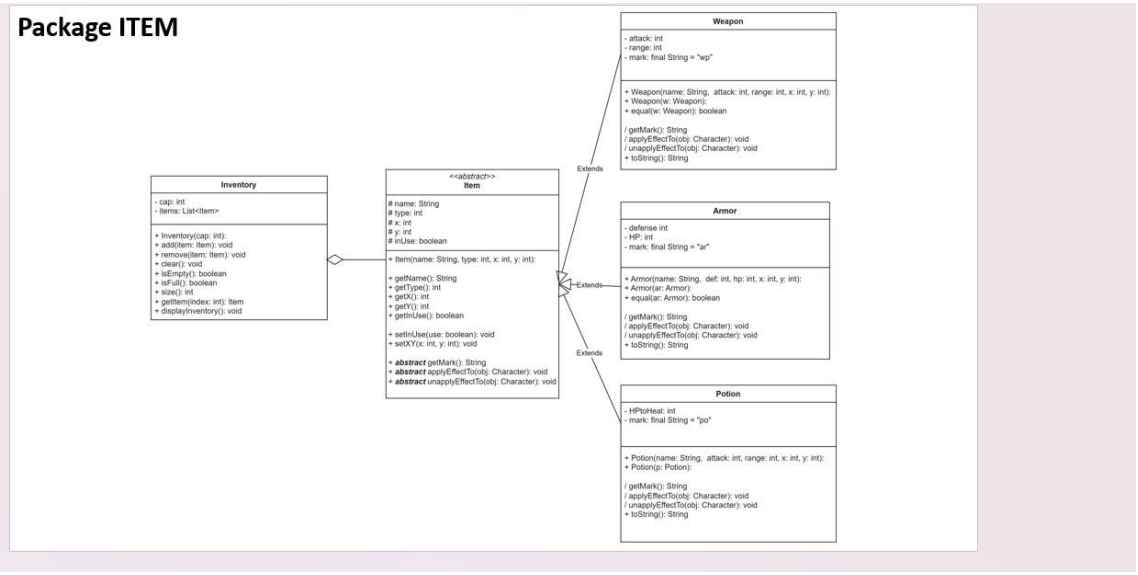
- **CHARACTER**: Quản lý nhân vật, bao gồm Player, Monster và các lớp kế thừa.



- **MAP**: Quản lý bản đồ và các loại ô (Tile).



- **ITEM**: Quản lý vật phẩm và kho đồ.



4.1.2 Mối quan hệ giữa các Package

- GameMain kết nối các Package để điều phối hoạt động.
- CHARACTER tương tác với ITEM để quản lý trang bị.
- MAP quản lý trạng thái và tương tác giữa nhân vật và bản đồ.

5 Phân tích đặc tính hướng đối tượng

5.1 Tính đóng gói

```
public abstract class Character implements Serializable
{
    protected String Name;
    protected int HP, maxHP;
    protected int Attack;
    protected int Defense;
    protected int Range;
    protected int x;
    protected int y;
```

Các thuộc tính trong tất cả các lớp đều được khai báo là *protected*. Mỗi lớp đều đóng gói dữ liệu của nó bằng cách ẩn các thuộc tính bên trong và chỉ cung cấp các phương thức *public* để truy cập và thay đổi dữ liệu.

Ví dụ: Trong lớp Character, các thuộc tính Name, HP đều là *protected*. Để truy cập các thuộc tính này, cần sử dụng các phương thức getter và setter tương ứng (ví dụ: getName(), setHP()).

Lợi ích:

- *Bảo vệ dữ liệu:* Ngăn chặn truy cập trực tiếp và sửa đổi dữ liệu từ bên ngoài, đảm bảo tính toàn vẹn của dữ liệu.
- *Kiểm soát truy cập:* Kiểm soát cách thức truy cập và thay đổi dữ liệu thông qua các phương thức *public*, tăng tính bảo mật.
- *Giảm sự phụ thuộc:* Các lớp khác không phụ thuộc vào cấu trúc dữ liệu bên trong, chỉ cần biết cách sử dụng các phương thức *public*.
- *Dễ bảo trì:* Khi cần thay đổi cấu trúc dữ liệu bên trong, chỉ cần sửa đổi lớp tương ứng mà không ảnh hưởng đến các lớp khác.

5.2 Tính kế thừa

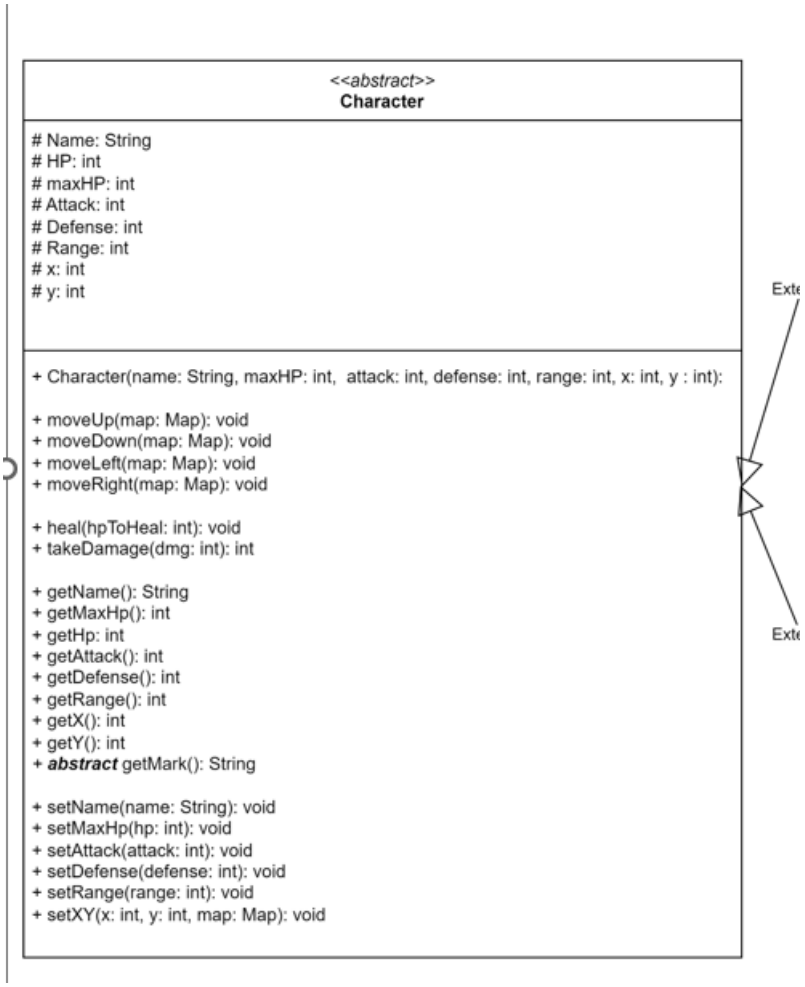
```
//------------------------------------ Getter Methods ------------------------------------------------
    public int getX()
    {return this.x;}
    public int getY()
    {return this.y;}
    public int getMaxHp()
    {return this.maxHP;}
    public int getHP()
    {return this.HP;}
    public int getRange()
    {return this.Range;}
    public int getAttack()
    {return this.Attack;}
    public int getDefense()
    {return this.Defense;}
    public String getName()
    {return this.Name;}

//------------------------------------ Setter Methods ------------------------------------------------

    public void setXY(int x, int y, Map map)
    {
        if(!map.isSolidAt(x, y) && !map.containMonsterAt(x, y) && !map.outOfBorder(x, y))
        {
            this.x = x;
            this.y = y;
        }
    }
```

- Lớp Player và Monster kế thừa từ lớp Character.
- Lớp RegularMonster, TargetMonster, và Boss kế thừa từ lớp Monster.

Lợi ích:

- *Tái sử dụng mã:* Tránh lặp lại mã, giúp mã ngắn gọn và dễ bảo trì hơn.
- *Mở rộng dễ dàng:* Dễ dàng thêm các loại người dùng mới bằng cách kế thừa từ Character hoặc các lớp con hiện có.
- *Tạo hệ thống phân cấp rõ ràng:* Thể hiện mối quan hệ giữa các đối tượng trong hệ thống.

5.3 Tính trừu tượng



Lớp Character định nghĩa một phương thức trừu tượng getMark() mà không cung cấp chi tiết triển khai.

Lợi ích:

- *Trừu tượng hóa:* Ẩn giấu chi tiết triển khai cụ thể của từng thao tác.
- *Tạo sự thống nhất:* Cung cấp một giao diện chung cho tất cả các lớp kế thừa, giúp mã dễ quản lý và hiểu hơn.

5.4 Tính đa hình

- Phương thức getMark() trong Character được ghi đè trong các lớp con như Player, Monster, với cách triển khai khác nhau.

```
//------------------------------------ Abstract Methods ---------------------------------
    public abstract String getMark();
```

```
@Override
public String getMark()
{return this.mark;}
```

Lợi ích:

- *Tính linh hoạt:* Có thể xử lý các đối tượng Character khác nhau (Player, Monster, ...) thông qua cùng một phương thức trừu tượng.
- *Khả năng mở rộng:* Dễ dàng thêm các loại Character mới mà không cần sửa đổi mã hiện có.
- *Tái sử dụng mã:* Player kế thừa các thuộc tính và phương thức cơ bản từ Character.
- *Bảo trì mã:* Dễ dàng sửa đổi logic cụ thể cho từng lớp con của Character mà không ảnh hưởng đến các lớp khác.

6 Xây dựng chương trình minh hoạ

6.1 Kết quả đạt được

Chương trình đã hoàn thiện với đầy đủ chức năng, giao diện đồ họa và các cơ chế chiến đấu, lưu trữ tiến trình, tương tác linh hoạt.

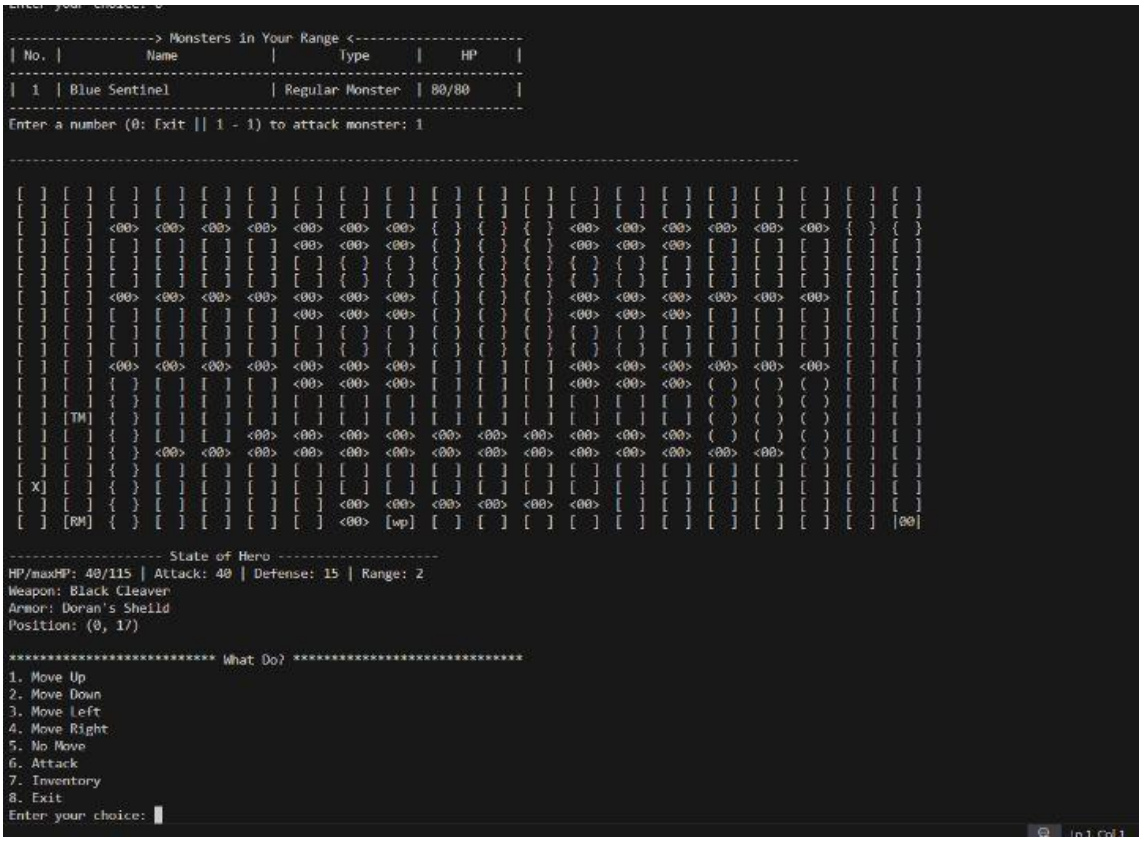
6.2 Minh hoạ các chức năng chính

6.2.1 Quản lí bản đồ và ô (Tile)

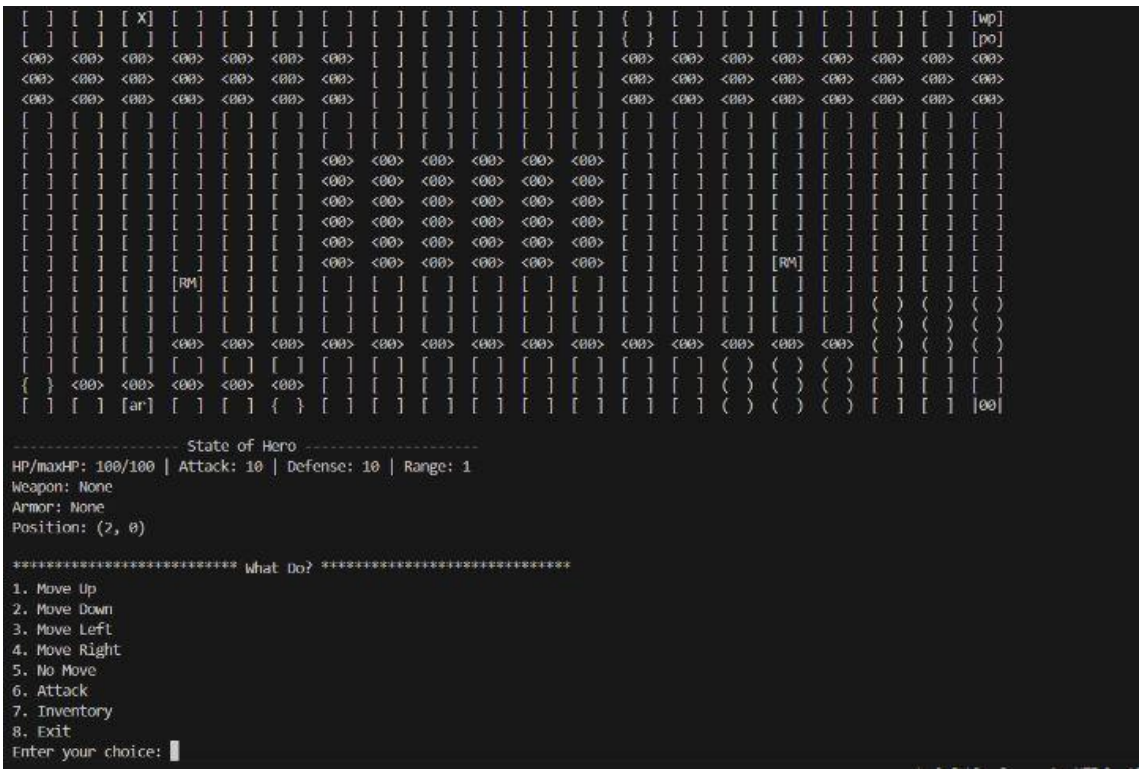
- Hiển thị trạng thái bản đồ, quái vật, vật phẩm.
- Tương tác giữa các loại ô như WallTile, FireTile, WaterTile.

6.2.2 Quản lí nhân vật và quái vật

- Player: Di chuyển, chiến đấu, sử dụng vật phẩm.

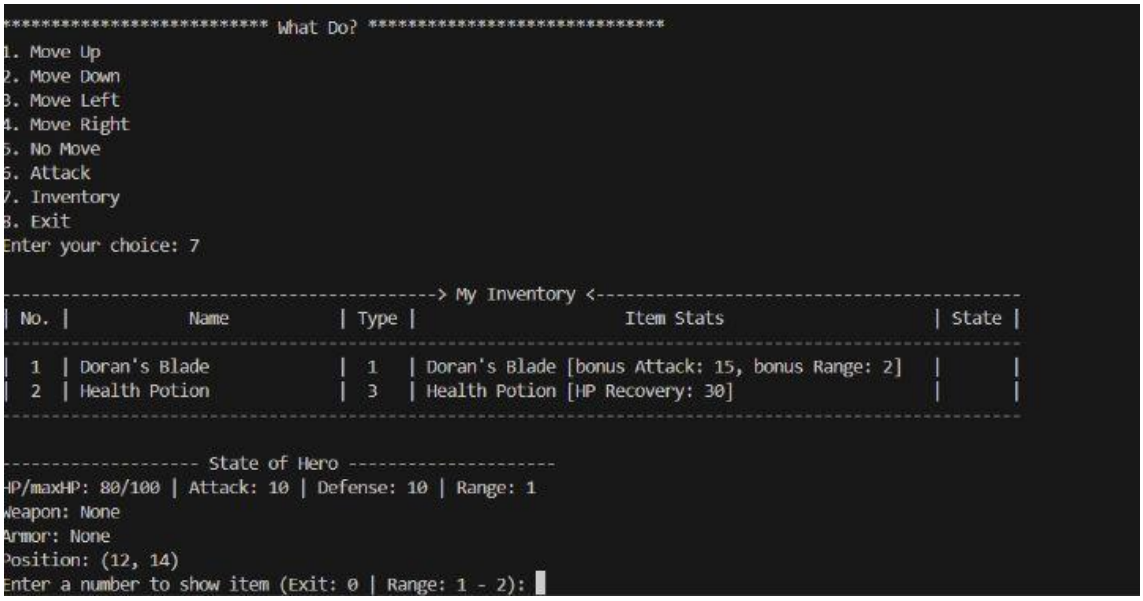


- Monster: Gồm RegularMonster, TargetMonster, Boss với hành vi khác nhau.



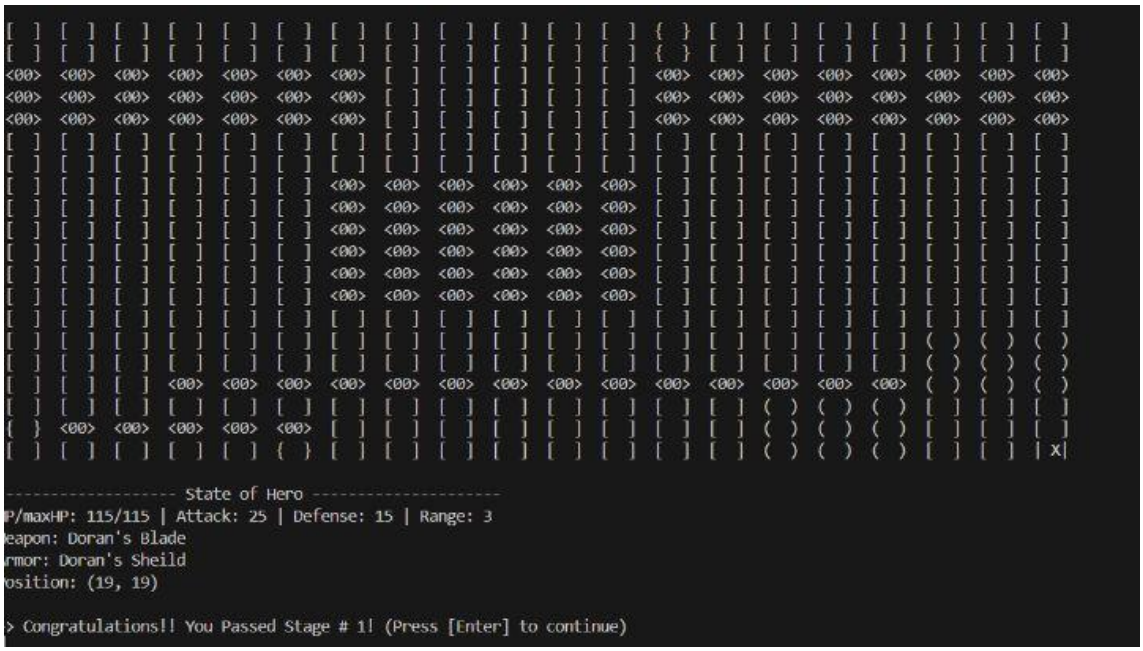
6.2.3 Hệ thống vật phẩm (Item)

- Weapon, Armor, Potion với hiệu ứng tăng sức mạnh.
- Quản lý kho đồ, sử dụng vật phẩm.



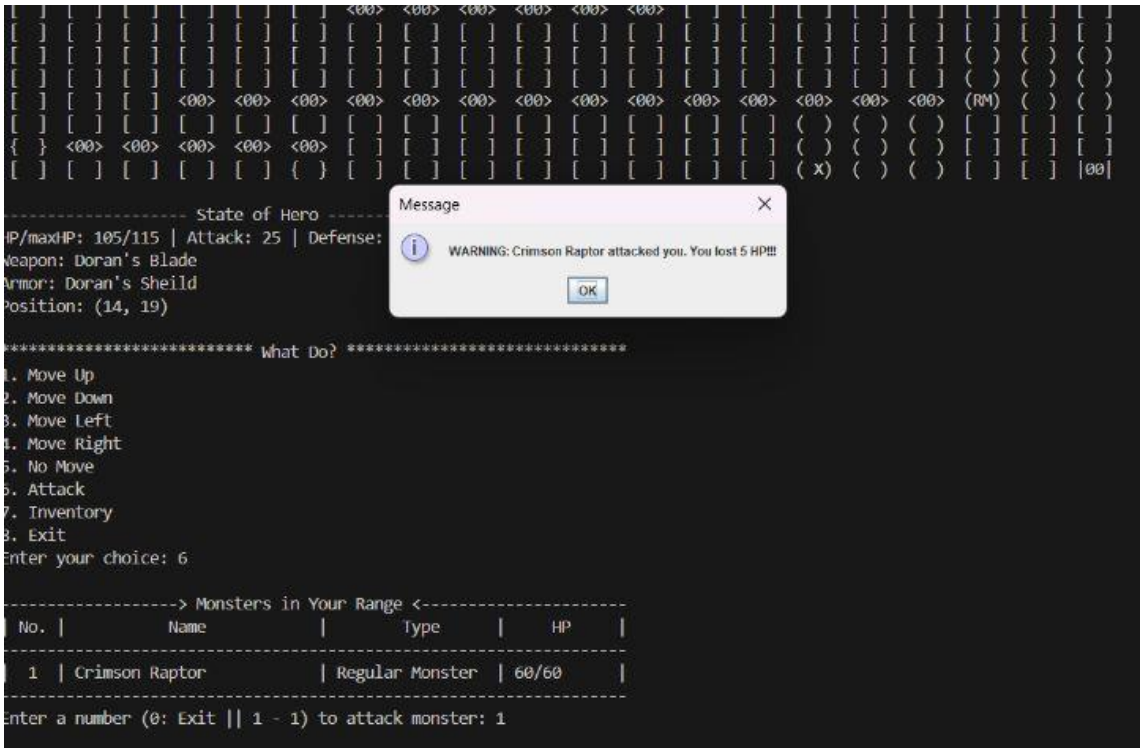
6.2.4 Quản lí giai đoạn chơi (GameStage)

- Điều khiển tiến trình trò chơi, bao gồm:
  - Chuyển đổi qua các bản đồ (giai đoạn mới).
  - Quản lý trạng thái thắng/thua và lưu/tải tiến trình chơi.
  - Xử lý hành động của người chơi và quái vật trong từng lượt.

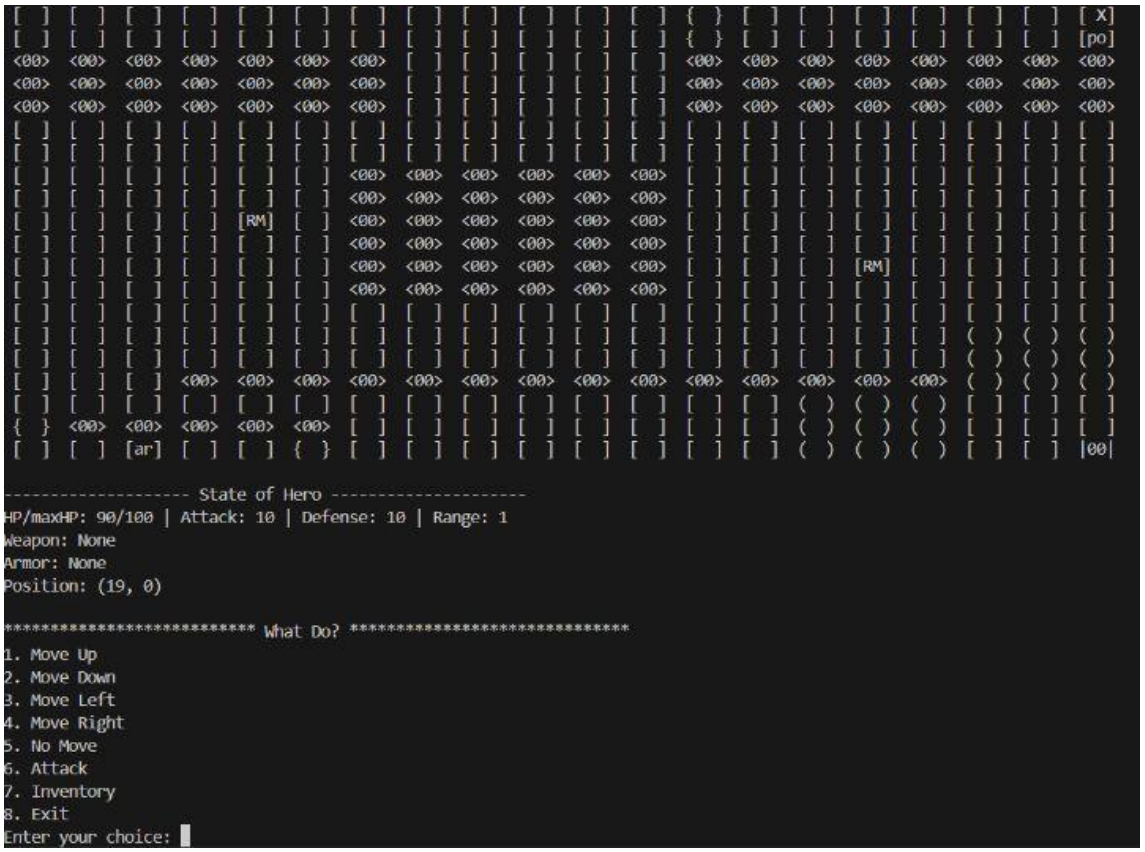


6.2.5 Hệ thống tương tác

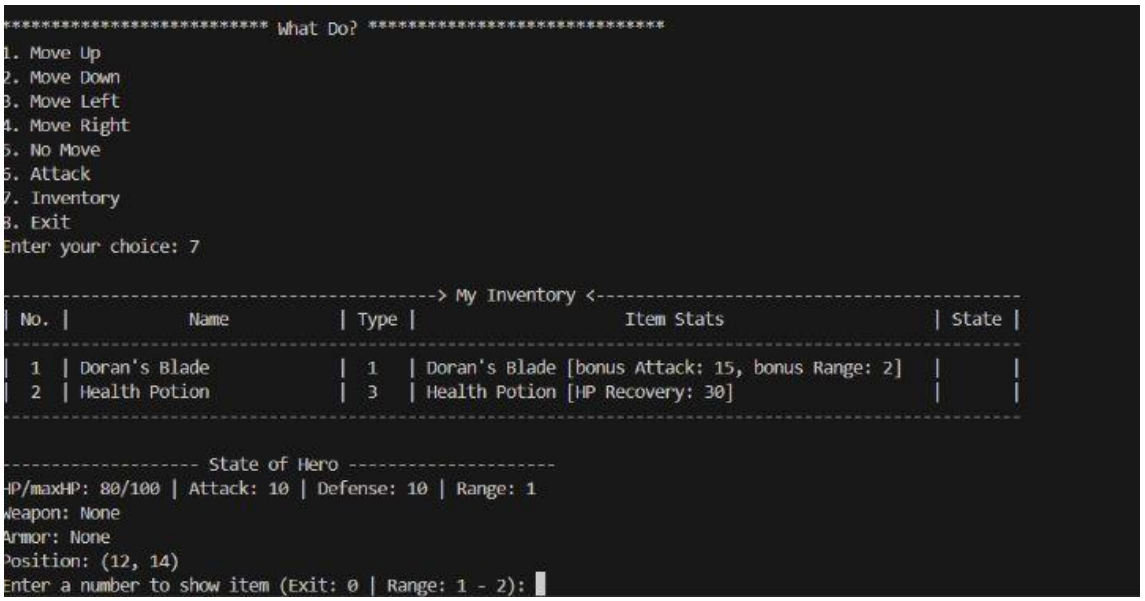
- **Chiến đấu:**
  - Tấn công quái vật trong phạm vi, với khả năng ưu tiên mục tiêu có máu thấp.



- **Thu thập vật phẩm:**
  - Tự động nhặt vật phẩm trên bản đồ nếu túi đồ không đầy.



- **Túi đồ (Inventory):**
  - Quản lý vật phẩm, cho phép xem thông tin, trang bị, hoặc xóa vật phẩm.



6.2.6 Giao diện và hiển thị

- Hiển thị toàn bộ trạng thái của bản đồ, nhân vật, quái vật, và vật phẩm theo dạng đồ họa ASCII.

6.2.7 Cơ chế lưu và tải

- Hỗ trợ lưu tiến trình chơi hiện tại và tiếp tục từ trạng thái đã lưu thông qua file dữ liệu.

6.2.8 Hệ thống khởi chạy (App)

- Chương trình khởi chạy game từ điểm bắt đầu, với khả năng chọn trò chơi mới hoặc tiếp tục từ dữ liệu đã lưu.