IN THE NAME OF GOD

name = AmirAliAhmadi
stu_id = 40213260281802

---

# Project Report: Analysis of the Game Code

## Table of Contents

---

## 1. Introduction
This project simulates a simple turn-based game where players and enemies take turns to perform actions. The main objective of the game is for the players to defeat the enemies using attacks, spells, and items before the enemies defeat them. This report provides a detailed analysis of the code in the `main.py` and `game.py` files.

---

## 2. Analysis of game.py

### 2.1 Class bcolors
This class is used to define different colors for console output. Each color is represented by a specific code.

```python code

class bcolors:
    HEADER='\033[95m'  # Purple for headers
    OKBLUE='\033[94m'  # Blue for positive messages
    OKGREEN='\033[92m' # Green for success messages
    WARNING='\033[93m' # Yellow for warnings
    FAIL='\033[91m'    # Red for error messages
    ENDC='\033[0m'     # Reset to default color
    BOLD='\033[1m'     # Bold text
    UNDERLINE='\033[4m' # Underlined text
```

### 2.2 Class `Person`
This class is used to define players and enemies. Each character has attributes and methods that control their behavior in the game.

```python code

class Person:
    def __init__(self, name, hp, mp, atk, magic, items):
        self.maxhp = hp  # Maximum health points
        self.hp = hp     # Current health points
        self.maxmp = mp  # Maximum magic points
        self.mp = mp     # Current magic points
        self.atkl = atk - 10  # Minimum physical attack damage
        self.atkh = atk + 10  # Maximum physical attack damage
        self.magic = magic    # List of available spells
        self.items = items    # List of available items
        self.actions = ["Attack", "Magic", "Items"]  # Available actions
        self.name = name      # Character name
```

### 2.3 Methods of Class `Person`
The methods of the `Person` class include:
- generate_damage(): Generates random damage.
- take_damage(dmg): Reduces health based on damage taken.
- heal(dmg): Increases health.
- get_hp(): Returns current health.
- get_max_hp(): Returns maximum health.
- get_mp(): Returns current magic points.
- get_max_mp(): Returns maximum magic points.
- reduce_mp(cost): Reduces magic points.
- choose_action(): Displays available actions.
- choose_magic(): Displays available spells.
- choose_item(): Displays available items.
- choose_target(enemies): Selects a target for attack.
- get_enemy_stats(): Displays enemy health status.
- get_stats(): Displays player health and magic status.

—----------------------------------------------------------------------------------------------------------------

## 3. Analysis of `main.py`

### 3.1 Importing Modules
The necessary modules for the game are imported in this section.

```python code

from game import Person, bcolors
```

from magic import Spell
from inventory import Item
import random

### 3.2 Creating Spells and Items
Different spells and items are created.

```python code

fire = Spell("Fire", 25, 600)
thunder = Spell("Thunder", 25, 600)
blizzard = Spell("Blizzard", 25, 600)
meteor = Spell("Meteor", 40, 1200)

potion = Item("Potion", "potion", "Heals 50 HP", 50)
elixer = Item("Elixer", "elixer", "Fully restores HP/MP of player", 9999)
grenade = Item("Grenade", "attack", "Deals 500 damage", 500)
```

### 3.3 Creating Players and Enemies
Players and enemies are created.

```python code

player1 = Person("Homayun", 3260, 132, 300, player_spells, player_items)
player2 = Person("AmirAli", 3260, 132, 300, player_spells, player_items)
player3 = Person("Hossein", 3260, 132, 300, player_spells, player_items)

enemy1 = Person("Enemy_1", 1250, 130, 560, [], [])
enemy2 = Person("Enemy_2", 1250, 130, 560, [], [])
enemy3 = Person("Enemy_3", 1250, 130, 560, [], [])
```

### 3.4 Main Game Loop
The main game loop is executed. In each turn, players perform their actions, followed by enemies attacking the players.

—-------------------------------------------------------------------------------------------------------------------

# Detailed Explanation of the Main Game Loop

The main game loop is the core of the game, controlling the flow of turns between players and enemies until the game ends. Below is a step-by-step breakdown of how the loop works:

—-------------------------------------------------------------------------------------------------------------------

## 1. **Start of the Loop (`while running`)**:

The main game loop begins with the condition `while running`. The variable `running` is initially set to `True`, and the loop continues until the game ends (i.e., one team wins).

python code
```
running = True
while running:

```

—------------------------------------------------------------------------------------------------------------------

## 2. **Display Player and Enemy Stats**:
At the start of each turn, the health points (HP) and magic points (MP) of all players and enemies are displayed. This is done using the `get_stats()` method for players and the `get_enemy_stats()` method for enemies.

python code
```
print("==============================")
print("\n\n")
print("NAME                    HP                                    MP")

for player in players:
    player.get_stats()

print("\n")

for enemy in enemies:
    enemy.get_enemy_stats()
```

—------------------------------------------------------------------------------------------------------------------

## 3. **Players' Turn**:
After displaying the stats, it's the players' turn. Each player takes their action one by one. If a player is defeated (`hp = 0`), their turn is skipped.

```python
for player in players:
    if player.get_hp() == 0:
        continue  # Skip defeated players
```

—------------------------------------------------------------------------------------------------------------------

## 4. **Player Action Selection**:

Each player can choose one of three actions:
- **Attack**: The player deals physical damage to a selected enemy.
- **Magic**: The player casts a spell, dealing magical damage to a selected enemy (if they have enough MP).
- **Items**: The player uses an item (e.g., a potion to heal or a grenade to deal damage).

```python
player.choose_action()
choice = input("    Choose action: ")
index = int(choice) - 1

if index == 0:  # Attack
    dmg = player.generate_damage()
    target = player.choose_target(enemies)
    enemies[target].take_damage(dmg)
    print("You attacked " + enemies[target].name + " for", dmg, "points of damage.")

elif index == 1:  # Magic
    player.choose_magic()
    magic_choice = int(input("    Choose magic: ")) - 1
    spell = player.magic[magic_choice]
    magic_dmg = spell.generate_damage()
    player.reduce_mp(spell.cost)
    target = player.choose_target(enemies)
    enemies[target].take_damage(magic_dmg)
    print(bcolors.OKBLUE + "\n" + spell.name + " deals", str(magic_dmg), "points of damage to " + enemies[target].name + bcolors.ENDC)

elif index == 2:  # Items
    player.choose_item()
    item_choice = int(input("    Choose item: ")) - 1
    item = player.items[item_choice]["item"]
    player.items[item_choice]["quantity"] -= 1

    if item.type == "potion":
        player.heal(item.prop)
        print(bcolors.OKGREEN + "\n" + item.name + " heals for", str(item.prop), "HP" + bcolors.ENDC)
    elif item.type == "elixer":
        player.hp = player.maxhp
        player.mp = player.maxmp
        print(bcolors.OKGREEN + "\n" + item.name + " fully restores HP/MP" + bcolors.ENDC)
    elif item.type == "attack":
        target = player.choose_target(enemies)
        enemies[target].take_damage(item.prop)
        print(bcolors.FAIL + "\n" + item.name + " deals", str(item.prop), "points of damage to " + enemies[target].name + bcolors.ENDC)
```

---

## 5. **Check for Defeated Enemies**:
After all players have taken their turns, the game checks if two or more enemies have been defeated (`hp = 0`). If so, the game ends with a "You win!" message, and the loop breaks.

```python
defeated_enemies = 0
for enemy in enemies:
    if enemy.get_hp() == 0:
        defeated_enemies += 1

if defeated_enemies >= 2:
    print(bcolors.OKGREEN + "You win!" + bcolors.ENDC)
    running = False
    break
```

---

## 6. **Enemies' Turn**:
If the game hasn't ended, it's the enemies' turn. Each enemy randomly selects a player to attack and deals damage. If an enemy is defeated (`hp = 0`), their turn is skipped.

```python
for enemy in enemies:
    if enemy.get_hp() == 0:
        continue  # Skip defeated enemies

    target = random.randrange(0, len(players))
    enemy_dmg = enemy.generate_damage()
    players[target].take_damage(enemy_dmg)
    print(bcolors.FAIL + enemy.name + " attacks " + players[target].name + " for",
enemy_dmg, "points of damage." + bcolors.ENDC)
```

---

## 7. **Check for Defeated Players**:
After all enemies have taken their turns, the game checks if two or more players have been defeated (`hp = 0`). If so, the game ends with a "Your enemies have defeated you!" message, and the loop breaks.

```python
defeated_players = 0
for player in players:
    if player.get_hp() == 0:
        defeated_players += 1

if defeated_players >= 2:
    print(bcolors.FAIL + "Your enemies have defeated you!" + bcolors.ENDC)
    running = False
    break
```

---

## 8. **Repeat the Loop**:
If none of the end conditions are met, the loop repeats, and a new turn begins. This continues until the game ends.

---

### Summary of the Game Loop:
1. **Display stats**: Show the HP and MP of all players and enemies.
2. **Players' turn**: Each player chooses an action (attack, magic, or item) and executes it.
3. **Check for defeated enemies**: If two or more enemies are defeated, the game ends with a victory message.
4. **Enemies' turn**: Each enemy attacks a random player.
5. **Check for defeated players**: If two or more players are defeated, the game ends with a defeat message.
6. **Repeat**: The loop continues until one of the end conditions is met.

---

## 4. Conclusion
This project implements a simple turn-based game where players and enemies take turns to perform actions. The `Person` class is the core of the game, containing attributes and methods that control the behavior of players and enemies. The game ends when two players or two enemies are defeated.

---

FINISH