```
import random
import re

class Character:
    """
    Represents a player character in the game.

    Attributes:
        name (str): The name of the character.
        character_class (str): The character's class (e.g., Warrior, Mage, Rogue).
        health (int): The current health points of the character.
        attack_power (int): The attack power of the character.
        inventory (list): A list containing the items the character has in their
inventory.
    """

    def __init__(self, name, character_class):
        """
        Initializes a new Character instance.

        Args:
            name (str): The name of the character.
            character_class (str): The character's class (e.g., Warrior, Mage,
Rogue).
        """
        self.name = name
        self.character_class = character_class
        self.health = 100
        self.attack_power = 20
        self.inventory = []

    def attack(self, enemy):
        """
        Attacks an enemy and reduces its health.

        Args:
            enemy (Enemy): The enemy instance to attack.
        """
        damage = random.randint(self.attack_power // 2, self.attack_power)
        enemy.health -= damage
        print(f"{self.name} attacks {enemy.name} and deals {damage} damage!")

    def use_item(self, item_name):
        """
        Uses an item from the character's inventory to heal.

        Args:
        item_name (str): The name of the item to use.
        """
        item_name = item_name.lower()  # Convert to lowercase for case-insensitive
comparison

        item_to_use = None
        for item in self.inventory:
            if item.name.lower() == item_name:
                item_to_use = item
                break

        if item_to_use:
```

```
            self.health += item_to_use.healing_power
            print(f"{self.name} uses {item_to_use.name} and gains
{item_to_use.healing_power} health.")
            self.inventory.remove(item_to_use)
            self.sort_inventory()
        else:
            print("Item not found in your inventory.")
    def sort_inventory(self):
        """
        Sorts the character's inventory alphabetically by item name
        """
        self.inventory.sort(key=lambda item: item.name.lower())
class Enemy:
    """
    Represents an enemy in the game.

    Attributes:
        name (str): The name of the enemy.
        health (int): The current health points of the enemy.
        attack_power (int): The attack power of the enemy.
    """

    def __init__(self, name, health, attack_power):
        """
        Initializes a new Enemy instance.

        Args:
            name (str): The name of the enemy.
            health (int): The initial health points of the enemy.
            attack_power (int): The attack power of the enemy.
        """
        self.name = name
        self.health = health
        self.attack_power = attack_power

class Item:
    """
    Represents an item that can be found in the game.

    Attributes:
        name (str): The name of the item.
        description (str): A description of the item.
        healing_power (int): The amount of health the item can restore.
    """

    def __init__(self, name, description, healing_power):
        """
        Initializes a new Item instance.

        Args:
            name (str): The name of the item.
            description (str): A description of the item.
            healing_power (int): The amount of health the item can restore.
        """
        self.name = name
        self.description = description
        self.healing_power = healing_power

class Location:
```

```
    """
    Represents a location in the game world.

    Attributes:
        name (str): The name of the location.
        description (str): A description of the location.
        items (list): A list of items found in the location.
        enemies (list): A list of enemies present in the location.
        directions (dict): A dictionary mapping direction names to adjacent
locations.
    """

    def __init__(self, name, description, items=None, enemies=None,
directions=None):
        """
        Initializes a new Location instance.

        Args:
            name (str): The name of the location.
            description (str): A description of the location.
            items (list, optional): A list of items found in the location. Defaults
to None.
            enemies (list, optional): A list of enemies present in the location.
Defaults to None.
            directions (dict, optional): A dictionary mapping direction names to
adjacent locations. Defaults to None.
        """
        self.name = name
        self.description = description
        self.items = items or []
        self.enemies = enemies or []
        self.directions = directions or {}

def create_character():
    """
    Creates a new character based on user input.

    Returns:
        Character: A new character instance.
    """
    name = input("Enter your character's name: ")
    character_class = input("Choose your character class (e.g., Warrior, Mage,
Rogue): ")
    new_character = Character(name, character_class)

    # Add items to the character's inventory
    items = create_items()
    new_character.inventory.extend(items)

    return new_character

def create_enemies():
    """
    Creates enemy instances.

    Returns:
        tuple: A tuple containing Enemy instances.
    """
    enemy1 = Enemy("Goblin", 50, 10)
```

```
    enemy2 = Enemy("Orc", 70, 15)
    return (enemy1, enemy2)

def create_items():
    """
    Creates item instances.

    Returns:
        list: A list containing Item instances.
    """
    health_potion = Item("Health Potion", "Restores 30 health points.", 30)
    magic_scroll = Item("Magic Scroll", "Unleashes a powerful spell.", 0)
    return [health_potion, magic_scroll]

def create_map():
    """
    Creates the game world map.

    Returns:
        dict: A dictionary containing location names as keys and Location instances
as values.
    """
    starting_location = Location("Starting Location", "You find yourself in a dark
forest.", items=[Item("Sword", "A basic weapon.", 0)])
    cave = Location("Cave", "A mysterious cave with glowing crystals.",
enemies=create_enemies())
    village = Location("Village", "A peaceful village with friendly villagers.",
items=create_items())

    starting_location.directions = {"north": cave, "east": village}
    cave.directions = {"south": starting_location}
    village.directions = {"west": starting_location}

    return {
        "Starting Location": starting_location,
        "Cave": cave,
        "Village": village
    }

def print_location_info(location):
    """
    Prints information about the current location.

    Args:
        location (Location): The current location to display information about.
    """
    print(location.name)
    print(location.description)
    print("Items in this location:")
    for item in location.items:
        print(f"- {item.name}: {item.description}")
    print("Enemies in this location:")
    for enemy in location.enemies:
        print(f"- {enemy.name} (Health: {enemy.health}, Attack Power:
{enemy.attack_power})")

def move_to_location(current_location, direction):
    """
    Moves the character to the specified location.
```

```
    Args:
        current_location (Location): The current location of the character.
        direction (str): The direction to move (e.g., north, south, east, west).

    Returns:
        Location: The new location after moving. If the direction is invalid,
returns the current location.
    """
    valid_directions = r"(north|south|east|west)"
    if re.match(valid_directions, direction.lower()):
        try:
            next_location = current_location.directions[direction]
            return next_location
        except KeyError:
            print("Invalid direction. You can't move in that direction.")
            return current_location
    else:
        print("Invalid direction. Please enter a valid direction.")
        return current_location

def main():
    """
    The main game loop.
    """
    print("Welcome to 'The Quest for Knowledge'!")
    player = create_character()
    map = create_map()

    current_location = map['Starting Location']
    while True:
        print_location_info(current_location)
        action = input("What do you want to do? (Type 'move' or 'attack' or 'use
item'): ").lower()

        if action == 'move':
            direction = input("Enter the direction (e.g., north, south, east,
west): ").lower()
            current_location = move_to_location(current_location, direction)
        elif action == 'attack':
            if current_location.enemies:
                enemy = random.choice(current_location.enemies)
                player.attack(enemy)
                if enemy.health <= 0:
                    print(f"{enemy.name} has been defeated!")
                    current_location.enemies.remove(enemy)
            else:
                print("There are no enemies in this location to attack.")
        elif action == 'use item':
            print("Your inventory:")
            for item in player.inventory:
                print(f"- {item.name}: {item.description}")
            item_name = input("Enter the name of the item you want to use: ")
            try:
                item = next(item for item in player.inventory if item.name.lower()
== item_name.lower())
                player.use_item(item_name)
            except StopIteration:
                print("Item not found in your inventory.")
```

```
        else:
            print("Invalid action. Try again.")

if __name__ == "__main__":
    main()
```