My game

```
# Base class for all entities in the game
class Entity:
    def __init__(self, name, hp, magic):
        self.name = name
        self.hp = hp
        self.magic = magic

    def is_alive(self):
        return self.hp > 0

# Race classes
class Race:
    def __init__(self, name, hp_bonus, magic_bonus):
        self.name = name
        self.hp_bonus = hp_bonus
        self.magic_bonus = magic_bonus

# Define some races
human = Race("Human", 10, 5)
elf = Race("Elf", 5, 10)
dwarf = Race("Dwarf", 15, 0)

class Race:
    def __init__(self, name, hp_bonus, magic_bonus, strength_bonus, agility_bonus, 
intelligence_bonus):
        self.name = name
        self.hp_bonus = hp_bonus
        self.magic_bonus = magic_bonus
        self.strength_bonus = strength_bonus
        self.agility_bonus = agility_bonus
        self.intelligence_bonus = intelligence_bonus

    def __str__(self):
        return (f"Race: {self.name}\n"
                f"HP Bonus: {self.hp_bonus}\n"
                f"Magic Bonus: {self.magic_bonus}\n"
                f"Strength Bonus: {self.strength_bonus}\n"
                f"Agility Bonus: {self.agility_bonus}\n"
                f"Intelligence Bonus: {self.intelligence_bonus}\n")

# Define some races
human = Race(
    name="Human",
    hp_bonus=10,
    magic_bonus=5,
    strength_bonus=5,
    agility_bonus=5,
    intelligence_bonus=5
)

elf = Race(
    name="Elf",
    hp_bonus=5,
    magic_bonus=15,
    strength_bonus=3,
    agility_bonus=10,
    intelligence_bonus=7
)

dwarf = Race(
    name="Dwarf",
    hp_bonus=20,
    magic_bonus=0,
    strength_bonus=10,
    agility_bonus=3,
    intelligence_bonus=2
)

orc = Race(
    name="Orc",
    hp_bonus=15,
    magic_bonus=0,
    strength_bonus=15,
    agility_bonus=2,
    intelligence_bonus=1
)

# Test the race definitions
if __name__ == "__main__":
    races = [human, elf, dwarf, orc]
    for race in races:
        print(race)
        print("=" * 30)


# Base class for player characters
class Character(Entity):
    def __init__(self, name, race, character_class):
        self.race = race
        self.character_class = character_class
        hp = 100 + race.hp_bonus + character_class.hp_bonus
        magic = 50 + race.magic_bonus + character_class.magic_bonus
        super().__init__(name, hp, magic)
        self.level = 1
        self.experience = 0
        self.skills = character_class.skills

    def level_up(self):
        self.level += 1
        self.hp += 10 + self.race.hp_bonus + self.character_class.hp_bonus
        self.magic += 5 + self.race.magic_bonus + self.character_class.magic_bonus
        print(f"{self.name} leveled up to level {self.level}!")

    def gain_experience(self, amount):
        self.experience += amount
        if self.experience >= self.level * 100:
            self.experience -= self.level * 100
            self.level_up()

    def use_skill(self, skill_name, target):
        if skill_name in self.skills:
            skill = self.skills[skill_name]
            if self.magic >= skill['magic_cost']:
                self.magic -= skill['magic_cost']
                target.hp -= skill['damage']
                print(f"{self.name} uses {skill_name} on {target.name} for {skill['damage']} damage!")
            else:
                print(f"Not enough magic to use {skill_name}!")
        else:
            print(f"{self.name} does not know {skill_name}!")

# Class for character classes
class CharacterClass:
    def __init__(self, name, hp_bonus, magic_bonus, skills):
        self.name = name
        self.hp_bonus = hp_bonus
        self.magic_bonus = magic_bonus
        self.skills = skills

# Define some character classes
warrior_skills = {
    "Slash": {"damage": 15, "magic_cost": 0},
    "Power Strike": {"damage": 30, "magic_cost": 5}, strong slam damage 15 magic cost 0  berserker
mode damage 45 last 25 seconds magic 30
}

mage_skills = {
    "Fireball": {"damage": 20, "magic_cost": 10},
    "Ice Shard": {"damage": 25, "magic_cost": 15},
Earth smash damage 10 magic cost 5
Shield damage 0 magic cost 0 defense 20
Healing damage 0 magic cost 20 magic over ride last 15 seconds damage 25 magic cost 20



# Create a character
player = Character("Aragorn", human, warrior) mage
enemy = Character("Goblin", dwarf,dragon zombie
# Example gameplay
player.use_skill("Slash", enemy)
enemy.use_skill("Fireball", player)
player.gain_experience(150) 300 450 500 650 700 850 900 1050 1220   # Should cause a level up

print(f"Player HP: {player.hp}, Player Magic: {player.magic}")
print(f"Enemy HP: {enemy.hp}, Enemy Magic: {enemy.magic}").
```

Other races

```
class Race:
    def __init__(self, name, hp_bonus, magic_bonus, strength_bonus, agility_bonus,
intelligence_bonus, skills, side_effects):
        self.name = name
        self.hp_bonus = hp_bonus
        self.magic_bonus = magic_bonus
        self.strength_bonus = strength_bonus
        self.agility_bonus = agility_bonus
        self.intelligence_bonus = intelligence_bonus
        self.skills = skills
        self.side_effects = side_effects

    def __str__(self):
        skills_str = "\n".join([f"{skill}: {desc}" for skill, desc in self.skills.items()])
        side_effects_str = "\n".join([f"{effect}: {desc}" for effect, desc in self.side_effects.items()])
        return (f"Race: {self.name}\n"
                f"HP Bonus: {self.hp_bonus}\n"
                f"Magic Bonus: {self.magic_bonus}\n"
                f"Strength Bonus: {self.strength_bonus}\n"
                f"Agility Bonus: {self.agility_bonus}\n"
                f"Intelligence Bonus: {self.intelligence_bonus}\n"
                f"Skills:\n{skills_str}\n"
                f"Side Effects:\n{side_effects_str}\n")
```

detailed races with skills and side effects

```
dragon_skills = {
    "Fire Breath": "Deals 50 damage to all enemies in range.",
    "Flight": "Allows to avoid ground-based attacks for 1 turn.",  shape shift cost 50 magic last 1 day
}

dragon_side_effects = {
    "Treasure Hoarding": "Reduces ability to carry non-treasure items by 50%.",
    "Large Size": "Difficulty in navigating narrow spaces, reducing agility by 2.",
}shape shift becomes extremely tired and loses most of the power and strength for  3 hours

dragon = Race(
    name="Dragon",
    hp_bonus=50,
    magic_bonus=30,
    strength_bonus=40,
    agility_bonus=-5,
    intelligence_bonus=10,
    skills=dragon_skills,
    side_effects=dragon_side_effects
)

zombie_skills = {
    "Infect": "Has a chance to turn an enemy into a zombie upon death.",
    "Undead Resilience": "Reduces damage taken by 20%.",
}

zombie_side_effects = {
    "Rotting Flesh": "Slowly loses 1 HP per turn due to decay.",
    "Low Intelligence": "Reduces intelligence-based skill effectiveness by 50%.",
}

zombie = Race(
    name="Zombie",
    hp_bonus=30,
    magic_bonus=0,
    strength_bonus=10,
    agility_bonus=-10,
    intelligence_bonus=-5,
    skills=zombie_skills,
    side_effects=zombie_side_effects
```

```
)

fish_skills = {
    "Water Jet": "Deals 20 damage and pushes the enemy back.",
    "Swim": "Increases speed in water environments by 50%.",
}

fish_side_effects = {
    "Water Dependency": "Loses 5 HP per turn when out of water.",
    "Fragile": "Takes 10% more damage from physical attacks.",
}
```

1. Dragon
- HP Bonus: +50
- Magic Bonus: +30
- Strength Bonus: +40
- Agility Bonus: -5
- Intelligence Bonus: +10
- Skills:
- Fire Breath: Deals 50 damage to all enemies in range.
- Flight: Allows to avoid ground-based attacks for 1 turn.
- Side Effects:
- Treasure Hoarding: Reduces ability to carry non-treasure items by 50%.
- Large Size: Difficulty in navigating narrow spaces, reducing agility by 2.

2. Zombie
- HP Bonus: +30
- Magic Bonus: 0
- Strength Bonus: +10
- Agility Bonus: -10
- Intelligence Bonus: -5
- Skills:
- Infect: Has a chance to turn an enemy into a zombie upon death.
- Undead Resilience: Reduces damage taken by 20%.
- Side Effects:

- Rotting Flesh: Slowly loses 1 HP per turn due to decay.
- Low Intelligence: Reduces intelligence-based skill effectiveness by 50%.

3. Fish

- HP Bonus: +5
- Magic Bonus: +10
- Strength Bonus: +5
- Agility Bonus: +15
- Intelligence Bonus: +10
- Skills:

- Water Jet: Deals 20 damage and pushes the enemy back.
- Swim: Increases speed in water environments by 50%.
- Side Effects:
- Water Dependency: Loses 5 HP per turn when out of water.
- Fragile: Takes 10% more damage from physical attacks.

4. Canton

- HP Bonus: +20
- Magic Bonus: +10
- Strength Bonus: +15
- Agility Bonus: +10
- Intelligence Bonus: +30
- Skills:
- Technomancy: Allows usage of advanced technology for combat, dealing 25 damage.
- Shield Generator: Reduces incoming damage by 20% for 3 turns.
- Side Effects:
- Overreliance on Technology: Magic abilities are 50% less effective.
- Energy Dependency: Requires an energy source to maintain peak performance; loses 10% effectiveness without it.

Explanation

- Each race is defined with specific attribute bonuses, unique skills, and side effects.
- Skills provide special abilities that can be used in combat or other situations.
- Side Effects introduce drawbacks or weaknesses for each race, balancing their strengths.
- The __str__ method in the Race class ensures that the race details are printed in a readable format for testing and debugging.

This setup allows for creating varied and interesting characters in an RPG game, each with their own strengths and weaknesses based on their race.

Name for the game

Prime rpg