```
import pytest


from swarm_models import OpenAIChat

from swarms.structs.agent import Agent

from swarms.structs.company import Company


# Mock OpenAIChat instance

llm = OpenAIChat(openai_api_key="test_key", max_tokens=4000)


# Mock Agents

ceo = Agent(llm=llm, name="CEO")

dev = Agent(llm=llm, name="Developer")

va = Agent(llm=llm, name="VA")

hr = Agent(llm=llm, name="HR")

shared_instructions = "Listen to your boss"




def test_add_agent():

    company = Company(

        org_chart=[[ceo, [dev, va]]],

        shared_instructions=shared_instructions,

    )

    company.add(hr)

    assert hr in company.agents
```

```
def test_get_agent():
    company = Company(
        org_chart=[[ceo, [dev, va]]],
        shared_instructions=shared_instructions,
    )
    company.add(hr)
    assert company.get("HR") == hr


def test_remove_agent():
    company = Company(
        org_chart=[[ceo, [dev, va]]],
        shared_instructions=shared_instructions,
    )
    company.add(hr)
    company.remove(hr)
    assert hr not in company.agents


def test_add_existing_agent():
    company = Company(
        org_chart=[[ceo, [dev, va]]],
        shared_instructions=shared_instructions,
    )
    company.add(hr)
    with pytest.raises(ValueError):
        company.add(hr)


def test_get_nonexistent_agent():
    company = Company(
        org_chart=[[ceo, [dev, va]]],
        shared_instructions=shared_instructions,
    )
    with pytest.raises(ValueError):
        company.get("Nonexistent")


def test_remove_nonexistent_agent():
    company = Company(
        org_chart=[[ceo, [dev, va]]],
        shared_instructions=shared_instructions,
    )
    with pytest.raises(ValueError):
        company.remove(hr)
```