Файл main.py

```
from operator import itemgetter
from typing import List, Dict, Tuple

# Models
class IDE:
    """Средство разработки"""
    def __init__(self, id: int, name: str, platform: str,
license: str):
        self.id = id
        self.name = name
        self.platform = platform
        self.license = license


class PL:
    """Язык программирования"""
    def __init__(
        self,
        id: int,
        memory: int,
        name: str,
        type: str,
        progParadigm: str,
        IDE_id: int,
    ):
        self.id = id
        self.memory = memory  # Занимаемая память (МБ)
        self.name = name
        self.type = type
        self.progParadigm = progParadigm
        self.IDE_id = IDE_id


class IDE_PL:
    """
    'Языки программирования средств разработки' для реализации
    связи многие-ко-многим
    """
    def __init__(self, PL_id: int, IDE_id: int):
        self.IDE_id = IDE_id
        self.PL_id = PL_id


# Data
IDEs = [
    IDE(1, "Visual Studio Code", "Windows, macOS, Linux",
"Бесплатная, открытый код"),
    IDE(2, "IntelliJ IDEA", "Windows, macOS, Linux",
"Коммерческая"),
    IDE(3, "Eclipse", "Windows, macOS, Linux", "Бесплатная,
открытый код"),
```

```
    IDE(4, "PyCharm", "Windows, macOS, Linux", "Коммерческая"),
    IDE(5, "Xcode", "macOS", "Бесплатная"),
]

PLs = [
    PL(1, 1, "Python", "Скриптовый", "Объектно-ориентированный",
1),
    PL(2, 3, "Java", "Компилируемый", "Объектно-
ориентированный", 3),
    PL(3, 2, "JavaScript", "Скриптовый", "Объектно-
ориентированный", 2),
    PL(4, 5, "C++", "Компилируемый", "Объектно-ориентированный",
4),
    PL(5, 4, "Go", "Компилируемый", "Процедурный", 5),
]

IDEs_PLs = [
    IDE_PL(1, 1),
    IDE_PL(2, 2),
    IDE_PL(3, 1),
    IDE_PL(4, 3),
    IDE_PL(3, 5),
    IDE_PL(2, 1),
    IDE_PL(1, 4),
    IDE_PL(5, 4),
    IDE_PL(1, 3),
    IDE_PL(2, 5),
]


# Functions
def get_one_to_many(PLs: List[PL], IDEs: List[IDE]) ->
List[Tuple[str, int, str]]:
    """Соединение данных один-ко-многим"""
    return [(p.name, p.memory, ide.name) for p in PLs for ide in
IDEs if p.IDE_id == ide.id]


def get_many_to_many(PLs: List[PL], IDEs: List[IDE], IDEs_PLs:
List[IDE_PL]) -> List[Tuple[str, int, str]]:
    """Соединение данных многие-ко-многим"""
    many_to_many_temp = [
        (ide.name, ip.IDE_id, ip.PL_id) for ide in IDEs for ip
in IDEs_PLs if ide.id == ip.IDE_id
    ]
    return [
        (pl.name, pl.memory, ide_name)
        for ide_name, _, pl_id in many_to_many_temp
        for pl in PLs
        if pl.id == pl_id
    ]
```

```python
def task_a1(one_to_many: List[Tuple[str, int, str]]) ->
List[Tuple[str, int, str]]:
    """Задание A1: Сортировка языков по имени IDE"""
    return sorted(one_to_many, key=itemgetter(2))


def task_a2(one_to_many: List[Tuple[str, int, str]], IDEs:
List[IDE]) -> List[Tuple[str, int]]:
    """Задание A2: Подсчёт памяти, занимаемой языками для каждой
IDE"""
    res = []
    for ide in IDEs:
        ide_pls = list(filter(lambda i: i[2] == ide.name,
one_to_many))
        if ide_pls:
            ide_memories = [memory for _, memory, _ in ide_pls]
            res.append((ide.name, sum(ide_memories)))
    return sorted(res, key=itemgetter(1), reverse=True)


def task_a3(many_to_many: List[Tuple[str, int, str]], IDEs:
List[IDE]) -> Dict[str, List[str]]:
    """Задание A3: Группировка языков по IDE"""
    res = {}
    for ide in IDEs:
        ide_pls = list(filter(lambda i: i[2] == ide.name,
many_to_many))
        res[ide.name] = [pl_name for pl_name, _, _ in ide_pls]
    return res


# Main
if __name__ == "__main__":
    one_to_many = get_one_to_many(PLs, IDEs)
    many_to_many = get_many_to_many(PLs, IDEs, IDEs_PLs)

    print("Задание A1")
    print(task_a1(one_to_many))

    print("\nЗадание A2")
    print(task_a2(one_to_many, IDEs))

    print("\nЗадание A3")
    print(task_a3(many_to_many, IDEs))
```

Файл main_test.py

```python
import unittest
from main import *


class TestRefactoredProgram(unittest.TestCase):
```

```
    def setUp(self):
        # Sample data for testing
        self.IDEs = [
            IDE(1, "IDE1", "Platform1", "Free"),
            IDE(2, "IDE2", "Platform2", "Commercial"),
        ]

        self.PLs = [
            PL(1, 100, "Python", "Script", "OOP", 1),
            PL(2, 200, "Java", "Compiled", "OOP", 2),
        ]

        self.IDEs_PLs = [
            IDE_PL(1, 1),
            IDE_PL(2, 2),
        ]

    def test_get_one_to_many(self):
        expected = [
            ("Python", 100, "IDE1"),
            ("Java", 200, "IDE2"),
        ]
        result = get_one_to_many(self.PLs, self.IDEs)
        self.assertEqual(result, expected)

    def test_get_many_to_many(self):
        expected = [
            ("Python", 100, "IDE1"),
            ("Java", 200, "IDE2"),
        ]
        result = get_many_to_many(self.PLs, self.IDEs,
self.IDEs_PLs)
        self.assertEqual(result, expected)

    def test_task_a1(self):
        one_to_many = get_one_to_many(self.PLs, self.IDEs)
        expected = [
            ("Python", 100, "IDE1"),
            ("Java", 200, "IDE2"),
        ]  # Already sorted by IDE name
        result = task_a1(one_to_many)
        self.assertEqual(result, expected)

    def test_task_a2(self):
        one_to_many = get_one_to_many(self.PLs, self.IDEs)
        expected = [
            ("IDE2", 200),
            ("IDE1", 100),
        ]  # Sorted by memory descending
        result = task_a2(one_to_many, self.IDEs)
        self.assertEqual(result, expected)

    def test_task_a3(self):
```

```
        many_to_many = get_many_to_many(self.PLs, self.IDEs,
self.IDEs_PLs)
        expected = {
            "IDE1": ["Python"],
            "IDE2": ["Java"],
        }
        result = task_a3(many_to_many, self.IDEs)
        self.assertEqual(result, expected)


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

Результаты main.py:

```
Задание A1
[('Java', 3, 'Eclipse'), ('JavaScript', 2, 'IntelliJ IDEA'), ('C++', 5, 'PyCharm'), ('Python', 1, 'Visual Studio Code'), ('Go', 4, 'Xco

Задание A2
[('PyCharm', 5), ('Xcode', 4), ('Eclipse', 3), ('IntelliJ IDEA', 2), ('Visual Studio Code', 1)]

Задание A3
{'Visual Studio Code': ['Python', 'JavaScript', 'Java'], 'IntelliJ IDEA': ['Java'], 'Eclipse': ['C++', 'Python'], 'PyCharm': ['Python',

Process finished with exit code 0
```

Результаты main_test.py

```
Testing started at 20:51 ...
Launching unittests with arguments python -m unittest



Ran 5 tests in 0.006s

OK

Process finished with exit code 0
```