Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра РТ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №1

Вариант Е1

Выполнил:
студент группы РТ5-31Б
Бляблин Егор

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2024 г.

Предметная область: студент, группа.

Текст программы:

main.py

```python
import exercises


def main():
    print('Задание E1 (поиск групп, начинающихся на "РТ5"):')
    exercises.exercise_e1()

    print("\nЗадание E2 (Сортировка групп по баллам за РК1):")
    exercises.exercise_e2()

    print('\nЗадание E3 (Поиск студентов с именем на "А"):')
    exercises.exercise_e3()


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

classes.py

```python
class Group:
    """Класс учебной группы"""

    _next_id = 0  # уникальный номер для следующей группы

    def __init__(self, title: str):
        """
        Конструктор группы

        Args:
            title (str): Название группы
        """
        self._id = Group._next_id
        self.title = title

        Group._next_id += 1  # увеличиваем счетчик

    @property
    def id(self):  # объявляем id через декоратор свойства, чтобы заблокировать смену
        return self._id

    @id.setter
    def id(self):
        raise AttributeError("Поле id не может быть изменено")

    def __eq__(self, other: "Group"):  # метод сравнения уменьшает повторяющейся кода
        return self.id == other.id

    def __hash__(
        self,
    ):  # метод хэширования для использования объекта группы в качестве ключа словаря
        return hash(self.id)

    def __repr__(self):  # текстовое представление объекта (для разработки)
        return f"Group ({self.id=}, {self.title=})"
```

```python
    def __str__(self):  # текстовое представление объекта
        return self.title


class Student:
    """Класс студента"""

    _next_id = 0  # уникальный номер для следующей группы

    def __init__(
        self,
        first_name: str,
        last_name: str,
        group: Group | None = None,
        rk_score: int = None,
    ):
        """
        Конструктор группы

        Args:
            first_name (str): Имя студента
            last_name (str): Фамилия студента
            group (Group | None, optional): Группа студента (опционально)
            rk_score (int, optional): Результат РК
        """
        self._id = Student._next_id
        self.first_name = first_name
        self.last_name = last_name
        self.group = group
        self.rk_score = rk_score

        Student._next_id += 1  # увеличиваем счетчик

    @property
    def id(self):  # объявляем id через декоратор свойства, чтобы заблокировать смену
        return self._id

    @id.setter
    def id(self):
        raise AttributeError("Поле id не может быть изменено")

    def __repr__(self):  # текстовое представление объекта (для разработки)
        return f"Student ({self.id=}, {self.first_name=}, {self.last_name=},
{self.group=}, {self.rk_score=})"

    def __str__(self):  # текстовое представление объекта
        return f"{self.last_name} {self.first_name}"


class GroupsStudents:
    """Класс связи группа-студент"""

    def __init__(self, group: Group, student: Student):
        """
        Конструктор связи

        Args:
            group (Group): Группа
            student (Student): Студент
        """
        self.group = group
        self.student = student
```

```
    def __eq__(
        self, other: "GroupsStudents"
    ):  # метод сравнения уменьшает повторяющейся кода
        return self.group.id is other.group.id and self.student.id is
other.student.id

    def __hash__(self):  # метод хэширования для использования объекта связи в
множестве
        return hash(f"{self.group.id}_{self.student.id}")
```

exercises.py

```
from classes import Group, GroupsStudents, Student


def exercise_e1():
    """
    ### Задание №1

    "Группа" и "Студент" связаны соотношением 1:M

    Выводит список всех групп, название которых начинается на "РТ5",
    и список числящихся в ней студентов
    """

    # объявляем группы
    ui5_36b = Group("ИУ5-36Б")
    rt5_31b = Group("РТ5-31Б")
    rt5_11b = Group("РТ5-11Б")

    # объединяем их в массив
    groups = [ui5_36b, rt5_31b, rt5_11b]

    students = [  # создаем объекты студентов (все совпадения случайны)
        Student("Евгений", "Турсков", ui5_36b),
        Student("Иван", "Щербаков", ui5_36b),
        Student("Тимур", "Яншин", ui5_36b),
        Student("Егор", "Бляблин", rt5_31b),
        Student("Андрей", "Васильев", rt5_31b),
        Student("Амир", "Габдулхаков", rt5_31b),
        Student("Артем", "Гладышев", rt5_31b),
        Student("Александр", "Красников", rt5_11b),
    ]

    for rt5_group in filter(
        lambda group: group.title.startswith("РТ5"), groups
    ):  # фильтруем группы, начинающиеся на "РТ5"
        print(f"Группа {rt5_group}:")
        print(
            "\n".join(
                (
                    f"> {student}"
                    for student in filter(  # отбираем только тех студентов, чья
группа соответствует текущей
                        lambda student: student.group is rt5_group, students
                    )
                )
            )
            or "[Пусто]"
        )
```

```python
def exercise_e2():
    """
    ### Задание №2

    "Группа" и "Студент" связаны соотношением 1:M

    Выводит список групп со средним результатом за РК1, отсортированный по этому
значению
    """

    # объявляем группы
    ui5_36b = Group("ИУ5-36Б")
    rt5_31b = Group("РТ5-31Б")
    rt5_11b = Group("РТ5-11Б")

    students = [  # объявляем объекты студентов (все совпадения случайны)
        Student("Евгений", "Турсков", ui5_36b, rk_score=20),
        Student("Иван", "Щербаков", ui5_36b),
        Student("Тимур", "Яншин", ui5_36b, rk_score=27),
        Student("Егор", "Бляблин", rt5_31b, rk_score=25),
        Student("Андрей", "Васильев", rt5_31b),
        Student("Амир", "Габдулхаков", rt5_31b),
        Student("Артем", "Гладышев", rt5_31b),
        Student("Александр", "Красников", rt5_11b),
    ]

    rk_results = {}  # средний балл за РК1 для каждой группы

    for group in [
        rt5_11b,
        ui5_36b,
        rt5_31b,
    ]:
        rk_scores = [  # собираем все баллы группы
            student.rk_score
            for student in filter(
                lambda student: student.group is group, students
            )  # если группа студента равна текущей
            if student.rk_score is not None  # если указана оценка
        ]

        rk_results[group] = (
            (sum(rk_scores) / len(rk_scores)) if rk_scores else None
        )  # записываем результат

    for group, result in sorted(
        rk_results.items(), key=lambda item: item[1] or 0, reverse=True
    ):  # сортируем по среднему баллу
        print(
            f"Средний балл группы {group} за РК1: {round(result, 2) if result else
'н/д'}"
        )


def exercise_e3():
    """
    ### Задание №3

    "Группа" и "Студент" связаны соотношением M:M
```

```
    Выводит список всех студентов, у которых имя начинается с буквы «А»,
    и названия групп, в которых они состоят
    """

    # объявляем студентов
    egor = Student("Егор", "Бляблин")
    andrey = Student("Андрей", "Васильев")
    amir = Student("Амир", "Габдулхаков")
    artem = Student("Артем", "Гладышев")

    # объявляем группы
    rt5 = Group("РТ5-31Б")
    ofp1 = Group("ОФП-1")
    smg1 = Group("СМГ-1")

    # объявляем все связи
    relations: set[GroupsStudents] = set(
        (
            GroupsStudents(rt5, egor),
            GroupsStudents(rt5, andrey),
            GroupsStudents(rt5, amir),
            GroupsStudents(rt5, artem),
            GroupsStudents(smg1, egor),
            GroupsStudents(ofp1, andrey),
            GroupsStudents(smg1, amir),
            GroupsStudents(ofp1, artem),
        )
    )

    # проходимся по всем уникальным студентам, имя которых начинается на "А"
    for found_student in set(
        filter(
            lambda student: student.first_name.startswith("A"),
            map(lambda relation: relation.student, relations),
        )
    ):
        print(
            f"{found_student} (состоит в ",
            ", ".join(
                str(group)  # выводим группу
                for group in map(
                    lambda relation: relation.group,
                    filter(  # фильтруем отношения, студент которых равен текущему
                        lambda relation: relation.student is found_student, relations
                    ),
                )
            ),
            ")",
            sep="",
        )
```

Результат выполнения:

```
Задание E1 (поиск групп, начинающихся на "РТ5"):
Группа РТ5-31Б:
> Бляблин Егор
> Васильев Андрей
> Габдулхаков Амир
> Гладышев Артем
Группа РТ5-11Б:
> Красников Александр

Задание E2 (Сортировка групп по баллам за РК1):
Средний балл группы РТ5-31Б за РК1: 25.0
Средний балл группы ИУ5-36Б за РК1: 23.5
Средний балл группы РТ5-11Б за РК1: н/д

Задание E3 (Поиск студентов с именем на "А"):
Гладышев Артем (состоит в ОФП-1, РТ5-31Б)
Габдулхаков Амир (состоит в СМГ-1, РТ5-31Б)
Васильев Андрей (состоит в ОФП-1, РТ5-31Б)
```