**Билет 5**

**1. Правила множественного наследования классов**

Множественное наследование — это возможность класса унаследовать свойства и методы сразу от нескольких родительских классов.

**Правила и особенности:**

1. **Порядок поиска методов (MRO – Method Resolution Order)**
   * Python использует алгоритм C3-линеаризации (правило старшинства классов), который определяет порядок вызова методов в иерархии наследования.
   * Используется метод super(), который помогает правильно вызывать методы родительских классов.
2. **Конфликты имен (diamond problem)**
   * Если несколько родительских классов имеют одинаковые методы, Python разрешает конфликт с помощью MRO.
   * Чтобы избежать проблем, можно использовать super(), миксины или интерфейсы.

Пример множественного наследования:

# Родительский класс A

class A**:**

def show**(**self**):**

print**(**"Метод класса A"**)**

# Родительский класс B

class B**:**

def show**(**self**):**

print**(**"Метод класса B"**)**

# Класс C наследуется от A и B

class C**(**A**,** B**):**

pass

# Создаём объект класса C

obj **=** C**()**

obj**.**show**()** # Выведет**:** "Метод класса A"**,** так как A стоит первым в наследовании

Чтобы понять порядок вызова методов, можно посмотреть MRO:

print**(**C**.**mro**())** # Покажет порядок вызова методов

2. Используя объектно-ориентированный подход (допускается STL) написать программу, выполняющую следующие действия:  
а) ввод с консоли информации о студенте (ФИО, средний балл)   
б) запись в файл.   
в) поиск в файле списка студентов, средний балл которых находится в указанных пользователем пределах.  
Код:   
import json

class Student**:**

"""

Класс для хранения информации о студенте

"""

def \_\_init\_\_**(**self**,** full\_name**:** str**,** gpa**:** float**):**

"""

Конструктор класса Student

**:**param full\_name**:** ФИО студента

**:**param gpa**:** Средний балл студента

"""

self**.**full\_name **=** full\_name # Полное имя студента

self**.**gpa **=** gpa # Средний балл студента

def to\_dict**(**self**):**

"""

Метод для представления объекта в виде словаря **(**для сохранения в JSON**)**

"""

**return** **{**"full\_name"**:** self**.**full\_name**,** "gpa"**:** self**.**gpa**}**

@staticmethod

def from\_dict**(**data**:** dict**):**

"""

Статический метод для создания объекта Student из словаря

"""

**return** Student**(**data**[**"full\_name"**],** data**[**"gpa"**])**

class StudentDatabase**:**

"""

Класс для работы с базой данных студентов **(**файл JSON**)**

"""

def \_\_init\_\_**(**self**,** filename**=**"students.json"**):**

"""

Конструктор класса StudentDatabase

**:**param filename**:** Имя файла**,** в котором хранятся данные

"""

self**.**filename **=** filename

def save\_student**(**self**,** student**:** Student**):**

"""

Метод для сохранения информации о студенте в файл

"""

students **=** self**.**load\_students**()** # Загружаем существующих студентов

students**.**append**(**student**.**to\_dict**())** # Добавляем нового

with open**(**self**.**filename**,** "w"**,** encoding**=**"utf-8"**)** as f**:**

json**.**dump**(**students**,** f**,** ensure\_ascii**=**False**,** indent**=**4**)** # Сохраняем в файл

def load\_students**(**self**):**

"""

Метод для загрузки списка студентов из файла

"""

**try:**

with open**(**self**.**filename**,** "r"**,** encoding**=**"utf-8"**)** as f**:**

**return** json**.**load**(**f**)**

except **(**FileNotFoundError**,** json**.**JSONDecodeError**):**

**return** **[]** # Если файла нет или он пуст**,** возвращаем пустой список

def find\_students\_by\_gpa**(**self**,** min\_gpa**:** float**,** max\_gpa**:** float**):**

"""

Метод для поиска студентов по среднему баллу в заданном диапазоне

**:**param min\_gpa**:** Минимальный средний балл

**:**param max\_gpa**:** Максимальный средний балл

"""

students **=** self**.**load\_students**()**

result **=** **[**s **for** s in students **if** min\_gpa **<=** s**[**"gpa"**]** **<=** max\_gpa**]**

**return** result

# Основная программа

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

db **=** StudentDatabase**()**

# Ввод данных о студенте

full\_name **=** input**(**"Введите ФИО студента: "**)**

gpa **=** float**(**input**(**"Введите средний балл: "**))**

student **=** Student**(**full\_name**,** gpa**)**

db**.**save\_student**(**student**)**

print**(**"Студент успешно сохранён!\n"**)**

# Поиск студентов по среднему баллу

min\_gpa **=** float**(**input**(**"Введите минимальный средний балл: "**))**

max\_gpa **=** float**(**input**(**"Введите максимальный средний балл: "**))**

found\_students **=** db**.**find\_students\_by\_gpa**(**min\_gpa**,** max\_gpa**)**

# Вывод найденных студентов

**if** found\_students**:**

print**(**"Найденные студенты:"**)**

**for** s in found\_students**:**

print**(**f"{s['full\_name']} - {s['gpa']}"**)**

**else:**

print**(**"Студенты не найдены в указанном диапазоне."**)**

 **Класс Student**

* Хранит информацию о студенте (ФИО и средний балл).
* Методы to\_dict() и from\_dict() помогают сохранять и загружать объекты в JSON.

 **Класс StudentDatabase**

* Работает с файлом JSON (чтение, запись, поиск студентов).

 **Основная программа**

* Вводит данные о студенте и сохраняет их.
* Позволяет найти студентов по среднему баллу в указанном диапазоне.