**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет МГТУ им. Н.Э. Баумана)**

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

**Рубежный контроль №1.**

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы № ИУ5-33Б

Пермяков Дмитрий

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

**Код программы.**

Также код программы можно найти на GitHub.

**Классы.**

**Путь: libs\DevelopEnvironment.**

*class* DevelopmentEnvironment:  
 *def* \_\_init\_\_(self, ID\_environment, name\_environment, count\_users, \*args):  
 *"""* ***:param*** *ID\_environment: id языка программирования. Int.* ***:param*** *name\_environment: Название языка программирования. Str.* ***:param*** *args: IDE для этих языков программирования. Tuple.  
 """* self.\_\_ID = ID\_environment  
 self.\_\_name = name\_environment  
 self.\_\_languageID = args  
 self.\_\_count\_users = count\_users  
  
 @property  
 *def* ID(self) -> int:  
 *return* self.\_\_ID  
  
 @ID.setter  
 *def* ID(self, ID\_environment):  
 self.\_\_ID = ID\_environment  
  
 @property  
 *def* name(self) -> str:  
 *return* self.\_\_name  
  
 @name.setter  
 *def* name(self, name\_env):  
 self.\_\_name = name\_env  
  
 @property  
 *def* languageID(self):  
 *return* self.\_\_languageID  
  
 @languageID.setter  
 *def* languageID(self, id):  
 self.\_\_languageID = id  
  
 @property  
 *def* count\_users(self) -> int:  
 *return* self.\_\_count\_users  
  
 @count\_users.setter  
 *def* count\_users(self, count):  
 *if* type(count) != int:  
 *raise* TypeError('Format of count\_users must be Int!')  
 self.\_\_count\_users = count

**Путь: libs\ProgrammingLanguage.**

*class* ProgrammingLanguage:  
 *def* \_\_init\_\_(self, ID\_language, name\_language):  
 self.\_\_ID\_language = ID\_language  
 self.\_\_name = name\_language  
  
 @property  
 *def* ID(self) -> int:  
 *return* self.\_\_ID\_language  
  
 @ID.setter  
 *def* ID(self, ID\_language):  
 self.\_\_ID\_language = ID\_language  
  
 @property  
 *def* name(self) -> str:  
 *return* self.\_\_name  
  
 @name.setter  
 *def* name(self, name\_language):  
 self.\_\_name = name\_language

**Путь: libs\IdeLanguage.**

*class* IdeLanguage:  
 *def* \_\_init\_\_(self, languageID, ideID):  
 self.language\_ID = languageID  
 self.IDE\_ID = ideID

**Tasks.py**

# Copyright © 2022 mightyK1ngRichard <dimapermyakov55@gmail.com>  
  
*from* operator *import* itemgetter  
  
  
*def* task\_01(one\_to\_many) -> list:  
 *""" ЗАДАНИЕ №1.  
 Вывести имя IDE, которое начинается с буквы 'V' и количество пользователей и отсортированный список языков,  
 которые поддерживается данным IDE.* ***:param*** *one\_to\_many: Список один ко многим.* ***:return****: Отфильтрованный список по первой букве, с отсортированным внутренним списком.  
 """* # Пояснение. Сначала отфильтровал список one\_to\_many по принципу: первая буква равна 'v', потом  
 # создал новый список res, сортируя список класса. Новая переменная, так как tuple нельзя изменять.  
 *return* [(name, count, sorted(lst)) *for* name, count, lst *in* list(filter(*lambda* el: el[0][0].lower() == 'v', one\_to\_many))]  
  
  
*def* task\_02(one\_to\_many) -> list:  
 *""" Задание 2.  
 Вывести список IDE, которое поддерживает меньше всего языков. Вывод совершить в порядке убывания.* ***:param*** *one\_to\_many:* ***:return****: Список кортежей, состоящий из имени IDE и кол-ва языков, которые поддерживает.  
 """* # Пояснение. Сначала создаю список, состоящий из кортежей. Первый элемент - имя IDE, второй - кол-во языков,  
 # которое он поддерживает.  
 # Потом сортирую список по второму элементу.  
 *return* sorted([(el[0], len(el[2])) *for* el *in* one\_to\_many],  
 key=itemgetter(1),  
 reverse=*True*)  
  
  
*def* task\_03(many\_to\_many) -> list:  
 *"""  
 Выведите список всех связанных IDE и языков, отсортированный по IDE, сортировка по языкам  
 произвольная.* ***:param*** *many\_to\_many: список созданный связями многие ко многим.* ***:return****: Список IDE и языков, отсортированные по IDE.  
 """* # Пояснение. Изначально сортирую many\_to\_many по IDE.name, после чего возвращаю список кортежей:  
 # имя и список языков.  
 *return* [(name, languages) *for* name, count, languages *in* sorted(many\_to\_many, key=itemgetter(0))]

**Main.py**

# Copyright © 2022 mightyK1ngRichard <dimapermyakov55@gmail.com>  
  
*from* libs.DevelopmentEnvironment *import* DevelopmentEnvironment  
*from* libs.IdeLanguage *import* IdeLanguage  
*from* libs.ProgrammingLanguage *import* ProgrammingLanguage  
*from* tasks *import* task\_01  
*from* tasks *import* task\_02  
*from* tasks *import* task\_03  
  
LANGUAGES = [  
 ProgrammingLanguage(0, 'C++'),  
 ProgrammingLanguage(1, 'C#'),  
 ProgrammingLanguage(2, 'Python'),  
 ProgrammingLanguage(3, 'Swift'),  
 ProgrammingLanguage(4, 'JavaScript'),  
 ProgrammingLanguage(5, 'CSS'),  
 ProgrammingLanguage(6, 'HTML'),  
 ProgrammingLanguage(7, 'Rust'),  
 ProgrammingLanguage(8, 'Go')  
]  
IDE = [  
 DevelopmentEnvironment(0, 'PyCharm', 267209, 2, 6, 5),  
 DevelopmentEnvironment(1, 'Atom', 171106, 5),  
 DevelopmentEnvironment(2, 'LITE', 305374, 8),  
 DevelopmentEnvironment(3, 'XCode', 253462, 3, 6, 5),  
 DevelopmentEnvironment(4, 'Visual Studio', 343304, 1, 0, 2, 6, 7),  
 DevelopmentEnvironment(5, 'IntelliJ IDEA', 426855, 4),  
 DevelopmentEnvironment(6, 'Rider', 308911, 1),  
 DevelopmentEnvironment(7, 'VScode', 908311, 0, 1, 2, 4, 5, 6, 7)  
]  
IDE\_LANGUAGES = [  
 IdeLanguage(0, 4),  
 IdeLanguage(1, 6),  
 IdeLanguage(2, 0),  
 IdeLanguage(3, 3),  
 IdeLanguage(4, 5),  
 IdeLanguage(5, 1)  
]  
  
  
*def* main\_code():  
 # Пояснение: Так как у в одной среде разработки можно использовать несколько языков,  
 # я создаю список в списке. Т.е список языков для определённого IDE.  
 one\_to\_many = [  
 (ide.name, ide.count\_users,  
 [language.name *for* el1 *in* ide.languageID *for* language *in* LANGUAGES *if* language.ID == el1])  
 *for* ide *in* IDE  
 ]  
 many\_to\_many\_temp = [  
 (language.name, language.ID, ide\_language.IDE\_ID)  
 *for* language *in* LANGUAGES  
 *for* ide\_language *in* IDE\_LANGUAGES  
 *if* language.ID == ide\_language.language\_ID  
 ]  
 many\_to\_many = [  
 (ide\_temp.name, ide\_temp.count\_users,  
 [lang.name *for* el *in* ide\_temp.languageID *for* lang *in* LANGUAGES *if* lang.ID == el])  
 *for* name\_language, languageID, ideID *in* many\_to\_many\_temp  
 *for* ide\_temp *in* IDE  
 *if* ide\_temp.ID == ideID  
 ]  
  
 # Можно посмотреть содержимое списков.  
 # print(\*one\_to\_many, sep='\n', end=f'\n {"=" \* 30}\n')  
 # print(\*many\_to\_many\_temp, sep='\n', end=f'\n {"=" \* 30}\n')  
 # print(\*many\_to\_many, sep='\n', end=f'\n {"=" \* 30}\n')  
  
 print(f'{"-" \* 10} Задание №1. {"-" \* 10}')  
 print(\*task\_01(one\_to\_many), sep='\n', end='\n\n')  
 print(f'{"-" \* 10} Задание №2. {"-" \* 10}')  
 print(\*task\_02(one\_to\_many), sep='\n', end='\n\n')  
 print(f'{"-" \* 10} Задание №3. {"-" \* 10}')  
 print(\*task\_03(many\_to\_many), sep='\n', end='\n\n')  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 main\_code()

**Результат программы.**

