|  |  |  |
| --- | --- | --- |
| Защищено:  Нардид А.Н.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |  | Демонстрация:  Нардид А.Н.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Отчет по рубежному контролю № 1 по курсу**

**Парадигмы и конструкции языков программирования**

**ГУИМЦ**

#### Тема работы: "Рубежный контроль представляет собой разработку программы на языке Python."

3

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-53Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Рабинович А. Д. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Москва, МГТУ - 2024 г.

# Тема и задание для выполнения рубежного контроля.

Тема работы: Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
   * ID записи о сотруднике;
   * Фамилия сотрудника;
   * Зарплата (количественный признак);
   * ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
   * ID записи об отделе;
   * Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
   * ID записи о сотруднике;
   * ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

**Вариант В.**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

**Варианты предметной области**

№ варианта: 23

Класс 1: Синтаксическая конструкция

Класс 2: Язык программирования

# Листинг программы

from operator import itemgetter

# Классы

class Syntax:

"""Синтаксическая конструкция"""

def \_\_init\_\_(self, id, name, lang\_id):

self.id = id

self.name = name

self.lang\_id = lang\_id

class Language:

"""Язык программирования"""

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class SyntaxLang:

"""

'Синтаксические конструкции языков'

для реализации связи многие-ко-многим

"""

def \_\_init\_\_(self, lang\_id, syntax\_id):

self.lang\_id = lang\_id

self.syntax\_id = syntax\_id

# Данные

languages = [

Language(1, 'Python'),

Language(2, 'C++'),

Language(3, 'JavaScript'),

]

syntaxes = [

Syntax(1, 'Цикл for', 1),

Syntax(2, 'Условие if', 2),

Syntax(3, 'Функция', 1),

Syntax(4, 'Объект', 3),

Syntax(5, 'Массив', 2),

]

syntaxes\_langs = [

SyntaxLang(1, 1),

SyntaxLang(1, 3),

SyntaxLang(2, 2),

SyntaxLang(2, 5),

SyntaxLang(3, 4),

]

# Основная функция

def main():

# Соединение данных один-ко-многим

one\_to\_many = [

(s.name, l.name)

for l in languages

for s in syntaxes

if s.lang\_id == l.id

]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [

(l.name, sl.lang\_id, sl.syntax\_id)

for l in languages

for sl in syntaxes\_langs

if l.id == sl.lang\_id

]

many\_to\_many = [

(s.name, l\_name)

for l\_name, \_, syntax\_id in many\_to\_many\_temp

for s in syntaxes

if s.id == syntax\_id

]

print('Задание В1: Синтаксические конструкции для Python')

res\_1 = [name for name, lang in one\_to\_many if lang == 'Python']

print(res\_1)

print('\nЗадание В2: Языки с минимальной конструкцией по алфавиту')

res\_2\_unsorted = []

for l in languages:

l\_syntaxes = list(filter(lambda x: x[1] == l.name, one\_to\_many))

if l\_syntaxes:

min\_syntax = min([syntax for syntax, \_ in l\_syntaxes])

res\_2\_unsorted.append((l.name, min\_syntax))

res\_2 = sorted(res\_2\_unsorted, key=itemgetter(1))

print(res\_2)

print('\nЗадание В3: Связь конструкций и языков')

res\_3 = sorted(many\_to\_many, key=itemgetter(0))

print(res\_3)

if \_\_name\_\_ == '\_\_main\_\_':

main()

# Результаты работы программы

Задание В1: Синтаксические конструкции для Python

['Цикл for', 'Функция']

Задание В2: Языки с минимальной конструкцией по алфавиту

[('C++', 'Массив'), ('JavaScript', 'Объект'), ('Python', 'Функция')]

Задание В3: Связь конструкций и языков

[('Массив', 'C++'), ('Объект', 'JavaScript'), ('Условие if', 'C++'), ('Функция', 'Python'), ('Цикл for', 'Python')]