**Міністерство освіти та науки України  
Національний технічний університет України  
«Київський політехнічний інститут»  
Факультет прикладної математики  
Кафедра системного програмування і спеціалізованих  
комп’ютерних систем**

**Лабораторна робота №1**з дисципліни

**«Організація баз даних»**

**Варіант №10**

Виконав: Латюк Сергій Олександрович

Студент групи КВ-42

Перевірив(ла)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**м. Київ**

**2016**

**Структура бази даних у вигляді колекцій**

**[Performer\_1, Performer\_2 … Performer\_n] —** список обєктів класу **Performer.**

**Performer:**

**name —** строка

**country** — строка

**composition list —** список обєктів класу **Composition**

**Composition:**

**title** — строка

**duration** — число

**performer** — строка

**Текст програми**

**model.py**

**import** pickle

**class Performer**:

**def** \_\_init\_\_(self, name, country):

self.name = name

self.country = country

self.composition\_list = []

**def get\_name**(self):

**return** self.name

**def set\_name**(self, name):

self.name = name

**for** composition **in** self.composition\_list:

composition.performer = name

**def get\_country**(self):

**return** self.country

**def set\_country**(self, country):

self.country = country

**def get\_composition\_list**(self):

**return** self.composition\_list

**def add\_composition**(self, title, duration, performer):

self.composition\_list.append(Composition(title, duration, performer))

**def find\_composition**(self, title):

**for** composition **in** self.composition\_list:

**if** composition.title == title:

**return** composition

**def remove\_composition**(self, title):

**for** composition **in** self.composition\_list:

**if** composition.title == title:

**del** composition

**break**

**else**:

**return** False

**return** True

**class Composition**:

**def** \_\_init\_\_(self, title, duration, performer):

self.title = title

self.duration = duration

self.performer = performer

**def set\_title**(self, title):

self.title = title

**def get\_title**(self):

**return** self.title

**def set\_duration**(self, duration):

self.duration = duration

**def get\_duration**(self):

**return** self.duration

**def set\_performer**(self, performer):

self.performer = performer

**def get\_performer**(self):

**return** self.performer

**class Database**:

# def \_\_init\_\_(self, first\_essence\_datafile, second\_essence\_datafile, pkl\_filename):

**def** \_\_init\_\_(self, pkl\_filename):

self.database\_file = pkl\_filename

f = open(pkl\_filename, 'rb')

self.data\_list = pickle.load(f)

# for performer in open(first\_essence\_datafile, 'r'):

# name, country = performer.split(';')

# country = country.rstrip('\n')

# self.data\_list.append(Performer(name, country))

# i = 0

# for composition in open(second\_essence\_datafile, 'r'):

# title, duration, performer = composition.split(';')

# performer = performer.rstrip('\n')

# if performer != self.data\_list[i].name:

# i += 1

# self.data\_list[i].composition\_list.append(Composition(title, duration, performer))

**def add\_performer**(self, name, country):

self.data\_list.append(Performer(name, country))

**def find\_performer**(self, name):

**for** performer **in** self.data\_list:

**if** performer.name == name:

**return** performer

**def del\_performer**(self, name):

search\_res = self.find\_performer(name)

**if** search\_res:

self.data\_list.remove(search\_res)

**def performers\_with\_average\_time\_4**(self):

res = []

**for** performer **in** self.data\_list:

sum\_of\_times = 0.0

**for** composition **in** performer.composition\_list:

dur = float(composition.duration)

sum\_of\_times += float(int(dur) \* 60 + # convert float format to seconds

100 \* (dur - int(dur))) / 60

comp\_list\_len = len(performer.composition\_list)

**if** comp\_list\_len != 0 **and** sum\_of\_times / len(performer.composition\_list) > 4:

res.append(performer)

**return** res

**def rewrite**(self):

f = open(self.database\_file, 'wb')

pickle.dump(self.data\_list, f)

**view.py**

*"""Functions for printing database data"""*

**def print\_composition\_list**(lst):

**for** elem **in** lst:

**print** '\tAuthor: %s; title: %s; duration: %s' % (elem.get\_performer(),

elem.get\_title(), elem.get\_duration())

**print** ''

**def print\_data**(database):

**if** database:

**for** field **in** database.data\_list:

**print** 'Performer: %s; country: %s' % (field.get\_name(), field.get\_country())

print\_composition\_list(field.get\_composition\_list())

**def print\_performer**(performer):

**if** performer:

**print** 'Performer: %s; country: %s\nCompositions:\n' % (performer.get\_name(),

performer.get\_country())

print\_composition\_list(performer.get\_composition\_list())

**def print\_special**(lst):

**print** 'Authors with average duration of song more than 4 minutes: \n'

**for** elem **in** lst:

print\_performer(elem)

"""Functions for displaying user interface"""

**def start\_menu**():

**print** '''Hello. You entered to the database with different music performers\n

where you can find out list ot their composition and some other infor-

mation about them.

To find performer by name press - 1

Add new performer to database press - 2

Edit information about performer press - 3

See list of performers with average

duration of the composition press - 4

See the list of all performers press - 5

Remove the performer press - 6

Exit - press - 7\n\n'''

**return** int(raw\_input('Enter the number: '))

**def search\_or\_del**():

**return** raw\_input('Enter name of performer: ')

**def add\_performer**():

**return** (raw\_input('Enter name of new performer: '),

raw\_input('Enter country of new performer: '))

**def add\_composition**():

**return** (raw\_input('Enter title of new composition: '),

raw\_input('Enter duration of new composition: '))

**def edit**():

**print** '''Edit information about performer press - 1

Add new composition to the list press - 2

Edit information about some composition press - 3\n\n'''

**return** int(raw\_input('Enter the number: '))

**def edit\_performer**():

**print** '''Edit name press - 1

Edit country press - 2\n\n'''

**return** int(raw\_input('Enter the number: '))

**def edit\_composition**():

**print** '''Edit title press - 1

Edit duration press - 2\n\n'''

**return** int(raw\_input('Enter the number: '))

**def search\_or\_del\_compos**():

**return** raw\_input('Enter the title: ')

**def new\_value**():

**return** raw\_input('Enter new value: ')

"""Functions for printing error messages"""

**def failed\_search**(object\_name):

**print** "%s you want to search doesn't exist\n" % object\_name

**controller.py**

**import** model

**import** view

db = model.Database('database.pkl')

**while** True:

query = view.start\_menu()

**if** query == 1:

search\_res = db.find\_performer(view.search\_or\_del())

**if** search\_res:

view.print\_performer(search\_res)

**else**:

view.failed\_search('Performer')

**elif** query == 2:

name, country = view.add\_performer()

db.add\_performer(name, country)

db.rewrite()

**elif** query == 3:

edit\_query = view.edit()

**if** edit\_query == 1:

edit\_performer\_query = view.edit\_performer()

search\_res = db.find\_performer(view.search\_or\_del())

**if** search\_res:

**if** edit\_performer\_query == 1:

search\_res.set\_name(view.new\_value())

**elif** edit\_performer\_query == 2:

search\_res.set\_country(view.new\_value())

**else**:

view.failed\_search('Performer')

**elif** edit\_query == 2:

title, duration = view.add\_composition()

perf\_name = view.search\_or\_del()

search\_res = db.find\_performer(perf\_name)

**if** search\_res:

search\_res.add\_composition(title, duration, perf\_name)

**else**:

view.failed\_search('Performer')

**elif** edit\_query == 3:

edit\_composition\_query = view.edit\_composition()

search\_res = db.find\_performer(view.search\_or\_del())

**if** search\_res:

composition = search\_res.find\_composition(view.search\_or\_del\_compos())

**if** composition:

**if** edit\_composition\_query == 1:

composition.set\_title(view.new\_value())

**elif** edit\_composition\_query == 2:

composition.set\_duration(view.new\_value())

**else**:

view.failed\_search('Composition')

**else**:

view.failed\_search('Performer')

db.rewrite()

**elif** query == 4:

view.print\_special(db.performers\_with\_average\_time\_4())

**elif** query == 5:

view.print\_data(db)

**elif** query == 6:

db.del\_performer(view.search\_or\_del())

db.rewrite()

**elif** query == 7:

**break**

**Скріншоти**



