Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра програмування та захисту інформації

**Звіт**

з виконаної лабораторної роботи № 5

дисципліни “ Программування ”

на тему

«Об’єктно-орієнтоване програмування у Python»

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

Смiрнова Н.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2017

**Лабораторная работа №5**

**Тема:** объектно-ориентированное программирование в Python

**Цель:** приобрести навыки работы с классами в Python

**ЗАДАНИЕ**

**Задание:**

1. Разработать класс "домашняя библиотека". Реализовать возможность работы с произвольным числом книг, поиска по книгам по нескольким параметрам (по автору, по году издания, по жанру и т.д.), добавление книг в библиотеку, удаления книг из нее, доступа к книге по номеру. Написать программу, которая будет демонстрировать все разработанные элементы класса.

class Book:

name = ''

authors = ''

genre = ''

publishing\_house = ''

publishing\_year = 0

def \_\_init\_\_(self, name\_='' , authors\_=' ', genre\_=' ', house\_='', year\_=0):

self.name = name\_

self.authors = authors\_

self.genre = genre\_

self.publishing\_house = house\_

self.publishing\_year = year\_

def getter(self):

info = [self.name,self.authors, self.genre, self.publishing\_house, self.publishing\_year]

return info

def setter(self,name\_,authors\_,genre\_, house\_, year\_):

self.name = name\_

self.authors = authors\_

self.genre = genre\_

self.publishing\_house = house\_

self.publishing\_year = year\_

class HomeLibrary:

library\_name = ''

library = []

def \_\_init\_\_(self, name\_=''):

self.library\_name = name\_

def add(self):

name = input("Введите название книги: ")

authors = input("Введите автора книги: ")

genre = input("Введите жанр книги: ")

house = input("Введите издателя книги: ")

year = input("Введите год издания книги: ")

book = Book()

book.setter(name,authors, genre, house,year)

self.library.append(book)

def add(self, book):

self.library.append(book)

def delete(self,name\_ = '',authors\_= '',genre\_= '', house\_ = '', year\_ =0):

for book in self.library:

if book.genre == genre\_ \

or book.name == name\_ \

or book.publishing\_house == house\_ \

or book.publishing\_year == year\_:

self.library.remove(book)

def find\_book(self,name\_ = '',authors\_= '',genre\_= ' ', house\_ = ' ', year\_ = 0):

book\_ = []

iter = 0

for book in self.library:

iter += 1

if book.genre == genre\_ \

or book.name == name\_ \

or book.publishing\_house == house\_ \

or book.publishing\_year == year\_:

book\_.append(book)

for i in book\_:

print(" Название книги :", i.name)

print(" Автор книги :", i.authors)

print(" Жанр книги :", i.genre)

print(" Издатель книги :", i.publishing\_house)

print(" Год издания книги :", i.publishing\_year)

return iter

def display\_book(self, name\_ = '', authors\_= '', genre\_= '', house\_ = '', year\_ =0):

for book in self.library:

if book.genre == genre\_ \

or book.name == name\_ \

or book.publishing\_house == house\_ \

or book.publishing\_year == year\_:

print(" Название книги :", book.name)

print(" Автор книги :", book.authors)

print(" Жанр книги :", book.genre)

print(" Издатель книги :", book.publishing\_house)

print(" Год издания книги :", book.publishing\_year)

def show\_all(self):

for book in self.library:

print(" \n\n\nНазвание книги :", book.name)

print(" Автор книги :", book.authors)

print(" Жанр книги :", book.genre)

print(" Издатель книги :", book.publishing\_house)

print(" Год издания книги :", book.publishing\_year)

import home\_library as lib

object = lib.HomeLibrary() # create new object library

new\_book = lib.Book() # create new book

new\_book.setter("Война и мир", "Л.Н.Толстой ", "Роман", "Мир", 2000)

object.add(new\_book)

object.show\_all()

new\_book = lib.Book() # create new book

new\_book.setter("Робинзон Крузо", "Д.Дефо ", "Приключения", "Мир", 2000)

object.add(new\_book)

new\_book = lib.Book() # create new book

new\_book.setter("Три Мушкетера", "А.Дюма ", "Роман", "Мир", 2000)

object.add(new\_book)

new\_book = lib.Book() # create new book

new\_book.setter("Тарзан", "Э.Берроуз ", "Роман", "Мир", 2000)

object.add(new\_book)

object.show\_all()

result = object.find\_book("Три Мушкетера")

print(result)

object.display\_book("Тарзан")

2. Разработать класс для представления сведений об успеваемости студента. Объект класса должен содержать поля для сохранения имени студента и баллов, полученных им за выполнение лабораторных работ и индивидуального творческого задания.

Обеспечить следующие методы класса:

конструктор, принимающий строку имя \_ студента и словарь, содержащий настройки курса в следующем формате:

1) максимально возможное количество баллов за сдачу индивидуального творческого задания;

2) максимально возможное количество баллов за сдачу одной лабораторной работы;

3) количество лабораторных работ в курсе;

4) доля баллов от максимума, которую необходимо набрать для получения экзамена автоматом.

* метод, помощью которого вносятся данные о количество попыток сдать лабораторную работу и оценка по последнюю попытку.
* метод, помощью которого вносятся данные о количество попыток сдать индивидуальное творческое Задание и оценка по последнюю попытку.
* метод, который возвращает кортеж (tuple), что содержит действительное число (сумму баллов студента по прохождения курса), и логическоезначение True или False в зависимости от того, достаточно этих баллов для получения оценки по экзамен автоматом.

class Dictionary:

lab\_count = 0

lab\_dictionary = {}

ind\_dictionary = {}

def \_\_init\_\_(self):

pass

class StudentInfo:

student\_name = ''

dictionary = Dictionary()

def \_\_init\_\_(self, name, lab\_count):

self.student\_name = name

self.dictionary.lab\_count = lab\_count

def set\_lab\_result(self, lab\_number, lab\_mark):

if lab\_number <=self.dictionary.lab\_count:

lab = {lab\_number: lab\_mark}

self.dictionary.lab\_dictionary.update(lab)

else:

print("All lab mark fill")

def set\_ind\_result(self, ind\_number, ind\_mark):

ind = {ind\_number: ind\_mark}

self.dictionary.ind\_dictionary.update(ind)

def get\_lab\_result(self, lab\_number):

res = self.dictionary.lab\_dictionary.get(lab\_number)

return res

def get\_ind\_result(self, ind\_number):

res = self.dictionary.ind\_dictionary.get(ind\_number)

return res

def result\_info(self):

for i in self.dictionary.lab\_dictionary:

if i <= self.dictionary.lab\_count:

res = self.get\_lab\_result(i)

print("Lab work # ", i, " mark = ", res)

for i in self.dictionary.ind\_dictionary:

res = self.get\_ind\_result(i)

print("Individual work # ", i, " mark = ", res)

def is\_allowed\_to\_exam(self):

lab = 0

iter= 0

ind = 0

for i in self.dictionary.lab\_dictionary:

if i <= self.dictionary.lab\_count:

lab += self.get\_lab\_result(i)

iter += 1

lab = lab / iter

iter = 0

for i in self.dictionary.ind\_dictionary:

ind += self.get\_ind\_result(i)

iter += 1

ind =ind / iter

is\_allowed = False

if lab >= 3 and ind >= 3:

is\_allowed = True

print("Total mark for lab work = ", lab)

print("Total mark for ind work = ", ind)

print("Is Allowed to exam : ", is\_allowed)

3. Разработать класс, следует функциональность стандартного типа str и содержит 2 новых метода:

1) метод, который принимает 1 аргумент s и возвращает True или False в зависимости от того, содержит ли строка повторы последовательностей символов длиной от 3 символов.

2) метод, который возвращает True или False в зависимости от того, является ли строка палиндромом. Регистрами символов пренебрегать.Пустая строка считать палиндромом.

class String(str):

string = ''

def \_\_init\_\_(self, string):

self.string = string

def is\_have\_repeat(self, s):

print(s, self.string)

iterator = 0

result = 0

for i in range(len(self.string)):

result = self.string.find(s, result + 1, len(self.string))

print(result)

if result >= len(self.string) \

or result == -1:

return False

iterator += 1

if iterator >= 3:

return True

def is\_polindrom(self):

if len(self.string) == 0:

return True

else:

beg = 0

end = len(self.string)-1

for i in range(int((end +1)/ 2)):

if self.string[beg] != self.string[end]:

return False

else:

beg += 1

end -= 1

return True

import string as s

string = "шалаш"

obj = s.String(string)

repeat = obj.is\_have\_repeat("ал")

palindrom = obj.is\_polindrom()

print("String have repeat substring :", repeat)

print("String is a palindrom", palindrom)

4. Разработать класс "колода карт", который будет включать закрыт массив элементов класса "карта". В карте будет храниться масть и номер.При создании экземпляра класса "колода карт", карты в колоде располагаются случайным образом. Обеспечить возможность вывода карты по телефону расположение в колоде, вывод всех карт, перемешивание колоды, выдачи одной карты из колоды, выдачи 6 карт из колоды. Написать программу, которая будет демонстрировать все разработанные элементы класса.

from random import shuffle, choice

class Card:

name = ''

number = -1

def \_\_init\_\_(self, n, num):

self.name = n

self.number = num

def get\_info(self,):

lists = [self.name, self.number]

return lists

def set\_info(n, num):

self.name = n

self.number = num

def display(self):

print(" Масть :", self.name)

print(" Номер карты :", self.number)

class Coloda:

card\_count = 36

mast = ["Бубен", "Сердце", "Треф", "Пики"]

kart = ["Шесть", "Семь", "Восемь", "Девять", "Десять","Валет", "Дама","Король","Туз"]

coloda = []

def \_\_init\_\_(self):

for i in self.mast:

for j in self.kart:

new\_card = Card(j,i)

self.coloda.append(new\_card)

shuffle(self.coloda)

def mix\_coloda(self):

shuffle(self.coloda)

def show\_card(self):

card = choice(self.coloda)

print(card.name, "-", card.number, ":", )

def show\_six(self):

list\_ = []

for card in range(6):

list\_.append(choice(self.coloda))

for card in list\_:

print(card.name, "-", card.number, ":", )

def show\_all(self):

for card in self.coloda:

print( card.name, "-" , card.number, ":",)

import card

obj = card.Coloda()

obj.show\_card()

obj.show\_six()

#obj.show\_all()

#obj.mix\_coloda()

#obj.show\_all()

5. Разработать класс "англо-русский словарь", обеспечить возможность хранения нескольких вариантов перевода для каждого слова.Обеспечить возможность вывода всех вариантов перевода введенного английского слова.

class Word:

word = ''

translator = []

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

self.word = name

def set\_translate(self, lis):

self.translator = lis

def set(self, w):

self.word = w.word

self.translator.append(w.translator)

def get(self):

return self.translator

class Dictionary:

dictionary = []

def \_\_init\_\_(self, word):

self.dictionary.append(word)

def add(self,w):

self.dictionary.append(w)

def find\_word(self, w):

for word in self.dictionary:

if w == word :

return word.translator

def delete(self, w):

for word in self.dictionary:

if w == word:

self.dictionary.remove(w)

def show(self, w):

for word in self.dictionary:

if w == word.word:

print(" Слово :", word.word)

print(" Перевод :", word.translator)

def show\_dictionary(self ):

for word in self.dictionary:

print(" Слово :", word.word)

print(" Перевод :", word.translator)

import dictionary as d

new\_word = d.Word("Hello" )

l = ["Привет", "Здравствуйте"]

new\_word.set\_translate(l)

new\_dictionary = d.Dictionary(new\_word)

new\_dictionary.show("Hello")

new\_word = d.Word("Call" )

l = ["Звонить", "Звать"]

new\_word.set\_translate(l)

new\_dictionary.add(new\_word)

new\_dictionary.show\_dictionary()

6. Создать абстрактный класс "Транспортное средство". На его основе реализовать классы "Самолет", "Автомобиль" и "Корабль". Классы должны иметь возможность задавать и получать координаты и параметры средств передвижения (стоимость, скорость, год выпуска и т.д.) задать с помощью полей. Для самолета должна быть определена высота, для самолета и корабля - количество пассажиров, для корабля - порт приписки.Динамические характеристики задать с помощью методов.

class Transport:

produced\_year = 0

name = ''

cost = 0

max\_speed = 0

weight = 0

coor\_x = 0

coor\_y = 0

pass\_count = 0

def \_\_init\_\_(self, cost\_= 0,prod\_ = 0, max\_s = 0, weight\_ = 0):

self.cost = cost\_

self.max\_speed = max\_s

self.produced\_year = prod\_

self.weight = weight\_

def set\_coordinate(self,x,y):

self.coor\_x = x

self.coor\_y = y

def get\_coordinate (self):

return self.coor\_x, self.coor\_y

def set\_paramet(self,prod\_, max\_s, cost\_, pas, weight\_):

self.cost = cost\_

self.max\_speed = max\_s

self.produced\_year = prod\_

self.weight = weight\_

def get\_parametr(self):

lis = [self.produced\_year, self.max\_speed, self.cost, self.pass\_count, self.weight]

return lis

class Auto(Transport):

gas\_tank =0

width = 0

height = 0

door\_count =0

def \_\_init\_\_(self):

Transport.\_\_init\_\_(self)

def display\_info(self):

print("год производства :", self.produced\_year)

print("стоимость :", self.cost)

print("Обьем бензобака :", self.gas\_tank)

print("макс скорость :", self.max\_speed)

print("текущии коорд. :", self.coor\_x, self.coor\_y)

print("Ширина :", self.width)

print("Высота :", self.height)

def setter(self, gas, w, h):

self.gas\_tank = gas

self.width = w

self.height = h

def getter (self):

listi = [self.gas\_tank, self.width, self.height]

l = self.get\_parametr()

listi.append(l)

return l

class Jet(Transport):

holder = ''

max\_height =0

def \_\_init\_\_(self):

Transport.\_\_init\_\_(self)

def display\_info(self):

print("владелец :", self.holder)

print("год производства :", self.produced\_year)

print("стоимость :", self.cost)

print("Макс. высота :", self.max\_height)

print("макс скорость :", self.max\_speed)

print("текущии коорд. :", self.coor\_x, self.coor\_y)

def getter(self):

lis = [self.holder, self.capacity]

res = self.get\_parametr()

lis.append(res)

return lis

def setter(self, hold, max\_height\_):

self.holder = hold

self.max\_height = max\_height\_

class Sheep(Transport):

holder = ''

capacity = 0

def \_\_init\_\_(self):

Transport.\_\_init\_\_(self)

def setter(self, hol, cap ):

self.capacity = cap

self.holder = hol

def display\_info(self):

print("владелец :", self.holder)

print("год производства :", self.produced\_year)

print("стоимость :", self.cost)

print("грузоподьемность :", self.capacity)

print("макс скорость :", self.max\_speed)

print("текущии коорд. :", self.coor\_x, self.coor\_y)

def getter(self):

listt = [self.holder, self.capacity]

res = self.get\_parametr()

listt.append(res)

return listt

import transport

auto = transport.Auto()

jet = transport.Jet()

sheep = transport.Sheep()

auto.set\_coordinate(25,20)

auto.setter(75, 3, 2)

auto.set\_paramet(2015,200,12000,4,2000)

res = auto.getter()

auto.display\_info()

print(res)

jet.set\_coordinate(25,20)

jet.setter("Firm.corp", 12000)

jet.set\_paramet(2015,200,12000,4,2000)

res = auto.getter()

jet.display\_info()

print(res)

sheep.set\_coordinate(25,20)

sheep.setter("Hold.comp", 50000)

sheep.set\_paramet(2015,200,12000,4,2000)

res = auto.getter()

sheep.display\_info()

print(res)