1 Задание

В выбранной предметной области определить не менее 3-х связанных между собой сущностей, их атрибуты. На языке Python описать классы, соответствующие найденым сущностям. Создать объекты описанной связки классов. Реализовать запросы, рекомендованные преподавателем. Для полученной программы построить диаграмму классов UML.

Анализ динамики показателей финансовой отчетности различных предприятий

Описание предметной области

Написать программу для анализа динамики показателей предприятия. Программа должна хранить информацию о предприятии (название, реквизиты, телефон, контактное лицо). Показатели должны иметь важность, характеризуемую некоторыми числовыми константами, а также иметь значение и единицы измерения. Также, программа должна хранить динамику изменения показателей, в зависимости от периода.

Классы объектов

Показатели (Название, Важность, Единица измерения).

Предприятия (Название, Банковские реквизиты, Телефон, Контактное лицо).

Динамика показателей (Показатель, Предприятие, Дата, Значение).

Решение

Описывается общий класс предок general, от которого будут наследоваться базовые классы.

```
class general:
    def __init__(self,code=0, name=""):
        self.setCode(code)
        self.setName(name)
    def setCode(self, value):self.__code=value
    def getCode(self):return self.__code
    def setName(self,value):self.__name=value
    def getName(self):return self.__name
```

Описывается класс "enterprise"хранящий информацию о предприятии.

```
from general import general
from requise import requise
```

```
class enterprise(requise, general):
    def __init__(self, code=0, name="",requisits=None,
    phone="", contact=""):
        general.__init__(self, code, name)
        self.setRequis(requisits)
        self.setPhone(phone)
        self.setContact(contact)
    def setRequis(self, value): self.__requisits = value
    def setPhone(self, value): self.__phone = value
    def setContact(self, value): self.__contact = value
    def getRequis(self):return self.__requisits
    def getPhone(self):return self.__phone
    def getContact(self):return self.__contact
    def printEnterprise(self):
        print("Предприятие: " + self.getName(), "Телефон: " +
        self.getPhone(),"Контактное лицо: " +
        self.getContact(), sep="\n")
    def getReqEnterprise(self):
        if self.getRequis():
            inn=self.getRequis().getInn()
            orgn=self.getRequis().getOgrn()
            adress=self.getRequis().getAdress()
        else: ""
        print("ИНН: " + str(inn),"ОРГН: " + str(orgn),
        "Адрес: " + adress, sep="\n", end="\n\n")
```

Для хранении реквизитов используется отдельный класс "requise".

```
class requise:
    def __init__(self, inn=0, ogrn=0, adress=""):
        self.setInn(inn)
        self.setOgrn(ogrn)
        self.setAdress(adress)
    def setInn(self, value): self.__inn = value
    def setOgrn(self, value): self.__ogrn = value
    def setAdress(self, value): self.__adress = value
    def getInn(self): return self.__inn
    def getOgrn(self): return self.__ogrn
    def getAdress(self): return self.__adress
    def printValue(self):
        print(self.getInn(), self.getOgrn(), self.getAdress())
```

Далее описывается класс "index отвечающий за показатели предприятия.

```
class index(general):
    def __init__(self, code=0, name="", importance=0, unit=""):
        general.__init__(self,code,name)
        self.setImportance(importance)
        self.setUnit(unit)
    def setImportance(self, value):
        if value == 1:
            self.__importance = "Высокая"
        elif value == 2:
            self.__importance = "Средняя"
        elif value == 3:
            self.__importance = "Низкая"
        else:
            self.__importance = "Не определённая"
    def setUnit(self,value): self.__unit = value
    def getImportance(self):return self.__importance
    def getUnit(self):return self.__unit
    def printIndex(self):
        print("Показатель: " + self.getName(),
        "Важность показателя: " + self.getImportance(),
        "Единица измерения: " + self.getUnit(),
        sep="\n", end="\n\n")
```

Для объединения показателей с предприятием создаётся класс "dynamics".

```
from enterprise import enterprise
from index import index
import datetime as DT

class dynamics(index, enterprise):
    def __init__(self, code=0, date="", sense=0,
    indexs=None, enterprises=None):
        self.setCode(code)
        self.setIndex(indexs)
        self.setEnterprises(enterprises)
        self.setDate(date)
        self.setSense(sense)
    def setIndex(self, value):
        if isinstance(value, index):self.__indexs=value
        else:self.__indexs = None
    def setEnterprises(self, value):
        if isinstance(value, enterprise):
        self.__enterprises = value
        else:
            self.__enterprises = None
```

```
    def setDate(self, value):self.__data =
    DT.datetime.strptime(value, '%d.%m.%Y').date()
    def setSense(self, value):self.__sense = value
    def getIndex(self):return self.__indexs
    def getEnterprises(self):return self.__enterprises
    def getDate(self):return self.__data
    def getSense(self):return self.__sense
    def getEnterpriseCode(self):
        if self.getEnterprises():
            return self.__enterprises.getCode()
        else:
            ent = enterprise()
    def getIndexCode(self):
        if self.getIndex():
            return self.__indexs.getCode()
        else:
            inx = index()
    def printIndicator(self):
        if self.getEnterprises():
            ent = self.getEnterprises()
        else:
            ent = enterprise()
        if self.getIndex():
            inx = self.getIndex()
        else:
            inx = index()
        print(self.getDate(), str(ent.getName()),
        inx.getName() + " =", self.__sense, inx.getUnit())
```

Чтобы отследить динамику покателей предприятия описывается класс "analysis"

```
from dynamics import dynamics
class analysis(dynamics):
    def __init__(self, dyn1, dyn2):
        self.setDyn1(dyn1)
        self.setDyn2(dyn2)
    def setDyn1(self, value):
        self.__dyn1 = value
    def setDyn2(self, value):
        self.__dyn2 = value
    def getDyn1(self):
        return self.__dyn1
    def getDyn2(self):
        return self.__dyn2
    def getAnalysis(self):
```

```
        if self.getDyn1():
            dyn1 = self.getDyn1()
        else:
            dyn1 = dynamics()
        if self.getDyn2():
            dyn2 = self.getDyn2()
        else:
            dyn2 = dynamics()
        sen1 = dyn1.getSense()
        sen2 = dyn2.getSense()
        if dyn1.getDate() >= dyn2.getDate():
            return -(sen2 - sen1)
        elif dyn1.getDate() < dyn2.getDate():
            return -(sen1 - sen2)
    def printAnalysis(self):
        print("Изменение показателя =", self.getAnalysis())
```

Для проверки работы программы используется класс "main".

```
from dynamics import dynamics
from enterprise import enterprise
from index import index
from requise import requise
from analysis import analysis

ent1 = enterprise(1, "Вконтакте", phone="+78005553535",
contact="Павел Дуров")
ent1.setRequis(requise(7842349892, 1079847035179,
"Ленинградский просп., 39, стр. 79, Москва"))
ent1.printEnterprise()
ent1.getReqEnterprise()
inx1 = index(1, "выручка", 1, "руб.")
inx1.printIndex()
dynam1 = dynamics(1, "12.1.2023", 250000, inx1, ent1)
dynam1.printIndicator()
dynam2 = dynamics(1, "12.2.2023", 300000, inx1, ent1)
dynam2.printIndicator()
ans = analysis(dynam1, dynam2)
ans.printAnalysis()
```

Результат работы программы:

Диаграмма UML для первого задания:

```
C:\Users\niket\AppData\Local\Programs\Python\Python312\python.exe G:\pyCharm\Project\oop2\main.py
Предприятие: Вконтакте
Телефон: +78005553535
Контактное лицо: Павел Дуров
ИНН: 7842349892
ОРГН: 1079847035179
Адрес: Ленинградский просп., 39, стр. 79, Москва

Показатель: выручка
Важность показателя: Высокая
Единица измерения: руб.

2023-01-12 Вконтакте выручка = 250000 руб.
2023-02-12 Вконтакте выручка = 300000 руб.
Изменение показателя = 50000

Process finished with exit code 0
```

Рис. 1

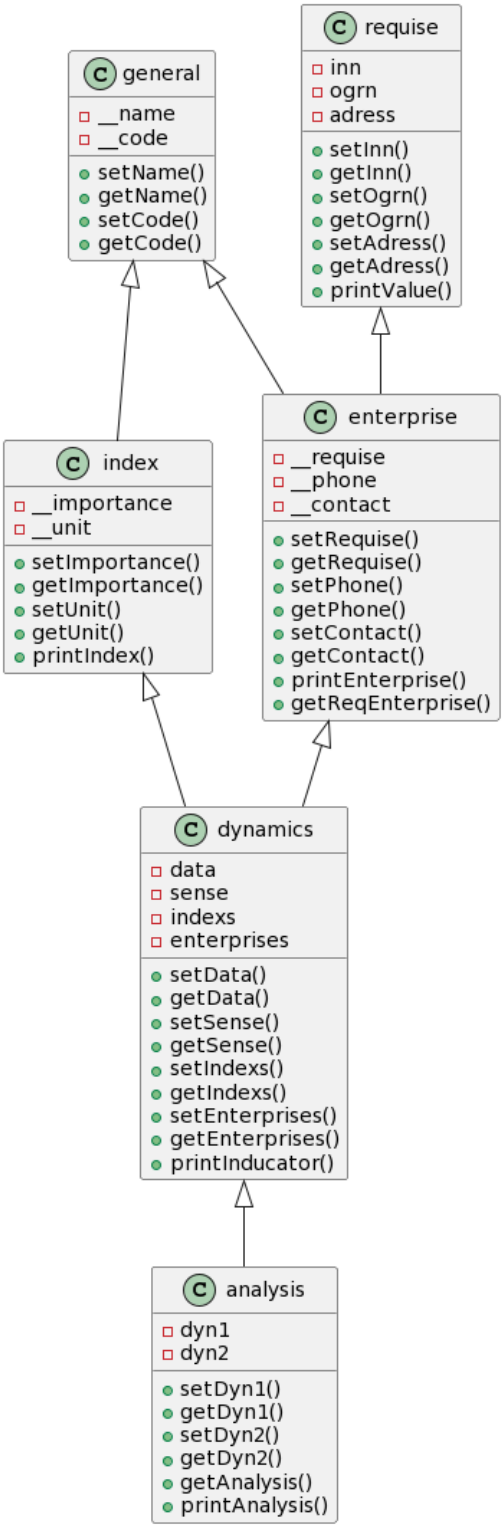
general
__name
__code
setName()
getName()
setCode()
getCode()
requise
inn
ogrn
adress
setInn()
getInn()
setOgrn()
getOgrn()
setAdress()
getAdress()
printValue()
index
__importance
__unit
setImportance()
getImportance()
setUnit()
getUnit()
printIndex()
enterprise
__requise
__phone
__contact
setRequise()
getRequise()
setPhone()
getPhone()
setContact()
getContact()
printEnterprise()
getReqEnterprise()
dynamics
data
sense
indexs
enterprises
setData()
getData()
setSense()
getSense()
setIndexs()
getIndexs()
setEnterprises()
getEnterprises()
printInducator()
analysis
dyn1
dyn2
setDyn1()
getDyn1()
setDyn2()
getDyn2()
getAnalysis()
printAnalysis()

Рис. 2

2 Задание

Написать функции для считывания и записи данных о выбранных сущностях из файла в формате XML, используя при этом классы, полученные при выполнении задания 1. Для полученной программы построить диаграмму классов UML. Реализовать запросы, рекомендованные преподавателем. Для демонстрации работы функции организовать вывод данных на экран в консоли.

В тектовом редакторе создаётся XML Файл "old.xml":

```
<archive>
 <enterprise code="1" name="Лукойл" phone='123456789' contact="Вадим Воробьев"/>
 <enterprise code="2" name="Сбербанк" phone='900' contact="Герман Греф"/>
 <enterprise code="3" name="Tesla" phone='999777999' contact="Илон Маск"/>

 <index code="1" name="Выручка" importance="1" unit="руб."/>
 <index code="2" name="Расходы" importance="1" unit="руб."/>
 <index code="3" name="Индекс" importance="1" unit="пт."/>


 <dynamics code="1" date="12.2.2023" sense="300" index="1" enterprise="1"/>
 <dynamics code="2" date="12.2.2023" sense="100" index="2" enterprise="1"/>

 <dynamics code="3" date="12.2.2023" sense="624" index="3" enterprise="1"/>

</archive>
```

Для хранения в программе считанных данных из XML-файла потребуется несколько объектов классов «enterprise», «index», «dynamics», которые удобно хранить в списках.

Создаётся общий потомок всех классов-списков "generalList":

```
from general import general

class generalList:
    def __init__(self):self.__list=[]
    def clear(self):self.__list=[]
    def findByCode(self,code):
        for l in self.__list:
            if l.getCode()==code:return l
    def getCodes(self):return [s.getCode() for s in self.__list]
    def getNewCode(self):return max(self.getCodes())+1
    def getItems(self):return [s for s in self.__list]
    def appendItem(self, value):
```

```
        if isinstance(value,general):self.__list.append(value)
    def appendList(self,value):self.__list.append(value)
    def removeList(self,code):
       for s in self.__list:
           if s.getCode()==code:self.__list.remove(s)
    def removeItem(self,value):
        if isinstance(value,general):self.__list.remove(value)
        if isinstance(value,int):
        self.__list.remove(self.findByCode(value))
```

Далее, создаются классы-списки для каждого основного класса:
Класс enterpriseList:

```
from generalList import generalList
from enterprise import enterprise

class enterpriseList(generalList, enterprise):
    def createItem(self, code, name, requisits=None, phone=0, contact=""):
        if code in self.getCodes():
        print("Предприятие с кодом %s уже существует")
        else:generalList.appendItem(self, enterprise(code, name,
        requisits, phone, contact))
    def newItem(self, name, requisits=None, phone="", contact=""):
        generalList.appendItem(self, enterprise(self.getNewCode(),
        name, requisits, phone, contact))
    def appendItem(self, value):
        if isinstance(value, enterprise):
        generalList.appendItem(self, value)
```

Класс indexList:

```
from generalList import generalList
from index import index

class indexList(generalList, index):
    def createItem(self, code, name, importance, unit):
        if code in self.getCodes():
            print("Показатель с кодом %s уже существует")
        else:
            generalList.appendItem(self, index(code, name,
            importance, unit))
    def newItem(self, name, importance, unit):
        generalList.appendItem(self, index(self.getNewCode(),
        name, importance, unit))
    def appendItem(self, value):
```

```
        if isinstance(value, index):
        generalList.appendItem(self, value)
```

Класс dynamicsList:

```
from generalList import generalList
from dynamics import dynamics

class dynamicsList(generalList, dynamics):
    def createItem(self, code=0, date="", sense=0,
    indexs=None, enterprises=None):
        if code in self.getCodes():
        print("Динамика показателей с кодом %s уже существует")
        else:generalList.appendItem(self, dynamics(code, date,
        sense, indexs, enterprises))
    def newItem(self, date="", sense=0, indexs=None, enterprises=None):
        generalList.appendItem(self, dynamics(self.getNewCode(), date,
        sense, indexs, enterprises))
    def appendItem(self, value):
        if isinstance(value, dynamics): generalList.appendItem(self, value)
```

Все эти списки объектов будут храниться в объекте класса «archive» в модуле «archive.py». Данный класс также содержит функции унечтожения объектов.

```
from enterpriseList import enterpriseList
from indexList import indexList
from dynamicsList import dynamicsList

class archive(enterpriseList, indexList, dynamicsList):
    def __init__(self):
        self.__enterpriseList=enterpriseList()
        self.__indexList=indexList()
        self.__dynamicsList=dynamicsList()
    def clear(self):
        self.__enterpriseList.clear()
        self.__indexList.clear()
        self.__dynamicsList.clear()
    def createEnterprise(self, code, name, requisits=None, phone="", contact=""):
        self.__enterpriseList.createItem(code, name, requisits, phone, contact)
    def newEnterprise(self, name, requisits=None, phone="", contact=""):
        self.__enterpriseList.newItem(name, requisits, phone, contact)
    def removeEnterprise(self, value):
        self.__enterpriseList.removeItem(value)
    def getEnterprise(self, code):return self.__enterpriseList.findByCode(code)
    def getEnterpriseList(self): return self.__enterpriseList.getItems()
```

```
    def getEnterpriseCodes(self): return self.__enterpriseList.getCodes()
    def createIndex(self, code=0, name="", importance=0, unit=""):
        self.__indexList.createItem(code, name, importance, unit)
    def newIndex(self, name, importance, unit):
        self.__enterpriseList.newItem(name, importance, unit)
    def removeIndex(self, value):
        self.__indexList.removeItem(value)
    def getIndex(self, code):return self.__indexList.findByCode(code)
    def getIndexList(self): return self.__indexList.getItems()
    def getIndexCodes(self): return self.__indexList.getCodes()
    def createDynamics(self, code=0, date="", sense=0, index=None, enterprise=None):
        self.__dynamicsList.createItem(code,date,sense, index, enterprise)
    def newDynamics(self, date,sense, index=None, enterprise=None):
        self.__dynamicsList.newItem(date, sense, index, enterprise)
    def removeDynamics(self, value):
        self.__dynamicsList.removeItem(value)
    def getDynamics(self, code):return self.__dynamicsList.findByCode(code)
    def getDynamicsList(self): return self.__dynamicsList.getItems()
    def getDynamicsCodes(self): return self.__dynamicsList.getCodes()
    def appendIndex(self, value):self.__indexList.appendItem(value)
    def appendEnterprise(self, value):self.__enterpriseList.appendItem(value)
```

Теперь реализуется чтение данных из XML-файла. Описывается класс «data», который будет являться общим предком для всех классов, предназначенных для чтения и записи данных в различных форматах. Различаться они будут только методами чтения и записи, которые зависят от форматов данных. Модуль «data.py»:

```
class data:
    def __init__(self, arch=None, inp="", out=""):
        self.setArch(arch)
        self.setInp(inp)
        self.setOut(out)
    def setArch(self, value): self.__arch = value
    def setInp(self, value): self.__inp = value
    def setOut(self, value): self.__out = value
    def getArch(self): return self.__arch
    def getInp(self): return self.__inp
    def getOut(self): return self.__out
    def readFile(self, filename):
        self.setInp(filename)
        self.read()
    def writeFile(self, filename):
        self.setOut(filename)
        self.write()
    def read(self): pass
```

```
    def write(self): pass
```

Для реализуется класс «dataxml» для работы с данными в формате XML. Этот класс наследуется от data, в нем переопределены методы «read()» и «write()».

Модуль «dataxml.py»:

```
import os,xml.dom.minidom
from data import data

class dataxml(data):
    def read(self):
        dom=xml.dom.minidom.parse(self.getInp())
        dom.normalize()
        for node in dom.childNodes[0].childNodes:
            if (node.nodeType==node.ELEMENT_NODE)and
            (node.nodeName=="enterprise"):
                code,name,phone, contact=0,"","",""
                for t in node.attributes.items():
                    if t[0]=="code":code=int(t[1])
                    if t[0]=="name":name=t[1]
                    if t[0]=="phone":phone=t[1]
                    if t[0]=="contact":contact=t[1]
                self.getArch().createEnterprise(code,name,
                None,phone,contact)
            if (node.nodeType==node.ELEMENT_NODE)and
            (node.nodeName=="index"):
                code,name,importance, unit=0,"","", ""
                for t in node.attributes.items():
                    if t[0]=="code":code=int(t[1])
                    if t[0]=="name":name=t[1]
                    if t[0]=="importance":importance=int(t[1])
                    if t[0] == "unit": unit = t[1]
                self.getArch().createIndex(code,name,
                importance, unit)
            if (node.nodeType==node.ELEMENT_NODE)and
            (node.nodeName=="dynamics"):
                code,date,sense, index, enterprise=0,""
                ,"", None, None
                for t in node.attributes.items():
                    if t[0]=="code":code=int(t[1])
                    if t[0]=="date":date=t[1]
                    if t[0]=="sense":sense=int(t[1])
                    if t[0]=="enterprise":enterprise
                    =self.getArch().getEnterprise(int(t[1]))
```

```
                    if t[0] == "index": index
                    = self.getArch().getIndex(int(t[1]))
                self.getArch().createDynamics(code,date,sense,
                index, enterprise)

    def write(self):
        dom=xml.dom.minidom.Document()
        root=dom.createElement("archive")
        dom.appendChild(root)
        for a in self.getArch().getEnterpriseList():
            aut=dom.createElement("enterprise")
            aut.setAttribute("code",str(a.getCode()))
            aut.setAttribute("name",a.getName())
            aut.setAttribute("phone",a.getPhone())
            aut.setAttribute("contact",a.getContact())
            root.appendChild(aut)
        for p in self.getArch().getIndexList():
            pub=dom.createElement("index")
            pub.setAttribute("code",str(p.getCode()))
            pub.setAttribute("name",p.getName())
            pub.setAttribute("importance",str(p.getImportance()))
            root.appendChild(pub)
        for b in self.getArch().getDynamicsList():
            bk=dom.createElement("dynamics")
            bk.setAttribute("code",str(b.getCode()))
            bk.setAttribute("date",str(b.getDate()))
            bk.setAttribute("sense",str(b.getSense()))
            bk.setAttribute("index", str(b.getIndexCode()))
            bk.setAttribute("enterprise", str(b.getEnterpriseCode()))
            root.appendChild(bk)
        f = open(self.getOut(),"w")
        f.write(dom.toprettyxml())
```

Теперь для проверки описывается программа в файле «main2.py», которая будет создавать объекты классов «archive» и «dataxml»

```
from archive import archive
from dataxml import dataxml

arch1=archive()
dat1=dataxml(arch1,"old.xml","new.xml")
dat1.read()
dat1.write()
```

В результате выполнения программы появится файл "new.xml"содержащий следующее:

```
<?xml version="1.0" ?>
<archive>
 <enterprise code="1" name="Лукойл" phone="123456789" contact="Вадим Воробьев"/>
 <enterprise code="2" name="Сбербанк" phone="900" contact="Герман Греф"/>
 <enterprise code="3" name="Tesla" phone="999777999" contact="Илон Маск"/>
 <index code="1" name="Выручка" importance="Высокая"/>
 <index code="2" name="Расходы" importance="Высокая"/>
 <index code="3" name="Индекс" importance="Высокая"/>
 <dynamics code="1" date="2023-02-12" sense="300" index="1" enterprise="1"/>
 <dynamics code="2" date="2023-02-12" sense="100" index="2" enterprise="1"/>
 <dynamics code="3" date="2023-02-12" sense="624" index="3" enterprise="1"/>
</archive>
```

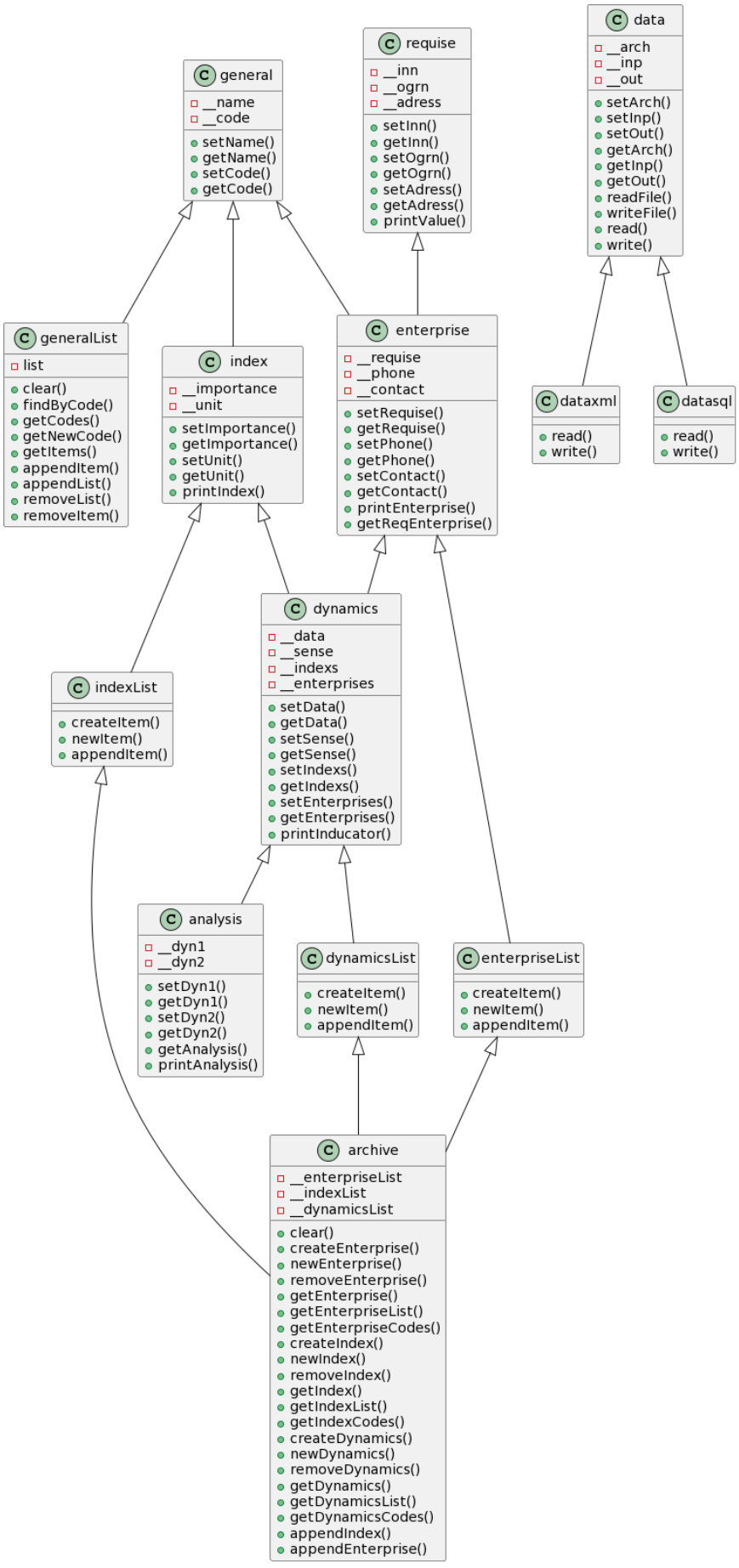
general
__name
__code
setName()
getName()
setCode()
getCode()
requise
__inn
__ogrn
__adress
setInn()
getInn()
setOgrn()
getOgrn()
setAdress()
getAdress()
printValue()
data
__arch
__inp
__out
setArch()
setInp()
setOut()
getArch()
getInp()
getOut()
readFile()
writeFile()
read()
write()
generalList
list
clear()
findByCode()
getCodes()
getNewCode()
getItems()
appendItem()
appendList()
removeList()
removeItem()
index
__importance
__unit
setImportance()
getImportance()
setUnit()
getUnit()
printIndex()
enterprise
__requise
__phone
__contact
setRequise()
getRequise()
setPhone()
getPhone()
setContact()
getContact()
printEnterprise()
getReqEnterprise()
dataxml
read()
write()
datasql
read()
write()
dynamics
__data
__sense
__indexs
__enterprises
setData()
getData()
setSense()
getSense()
setIndexs()
getIndexs()
setEnterprises()
getEnterprises()
printInducator()
indexList
createItem()
newItem()
appendItem()
analysis
__dyn1
__dyn2
setDyn1()
getDyn1()
setDyn2()
getDyn2()
getAnalysis()
printAnalysis()
dynamicsList
createItem()
newItem()
appendItem()
enterpriseList
createItem()
newItem()
appendItem()
archive
__enterpriseList
__indexList
__dynamicsList
clear()
createEnterprise()
newEnterprise()
removeEnterprise()
getEnterprise()
getEnterpriseList()
getEnterpriseCodes()
createIndex()
newIndex()
removeIndex()
getIndex()
getIndexList()
getIndexCodes()
createDynamics()
newDynamics()
removeDynamics()
getDynamics()
getDynamicsList()
getDynamicsCodes()
appendIndex()
appendEnterprise()

Рис. 3

3 Задание

Написать функции для считывания и записи данных о выбранных сущностях из базы данных SQLite, используя при этом классы, полученные при выполнении заданиях 1 и 2. Для полученной программы построить диаграмму классов UML.

Реализовать запросы, рекомендованные преподавателем. Для демонстрации работы функции организовать вывод данных на экран в консоли.

Для реализации чтения данных из базы данных SQLite описывается класс "datasql содержащий функции чтения и записи данных в фромате SQLite.

Модуль «datasql.py»:

```
import os
import sqlite3 as db
from data import data

emptydb = """
PRAGMA foreign_keys = ON;

create table enterprise
(code integer primary key,
name text,
phone text,
contact text);

create table indexs
(code integer primary key,
name text,
importance text,
unit text);

create table dynamics
(code integer primary key,
date text,
sense integer,
indexs integer references indexs(code) on update cascade on delete set null,
enterprise integer references enterprise(code) on update cascade on delete set null);
"""

class datasql(data):
    def read(self):
        conn = db.connect(self.getInp())
        curs = conn.cursor()
        curs.execute("select code, name, phone, contact from enterprise")
        data=curs.fetchall()
```

```
        for r in data:self.getArch().createEnterprise(r[0], r[1], r[2], r[3])
        curs.execute("select code, name, importance, unit from indexs")
        data=curs.fetchall()
        for r in data:self.getArch().createIndex(r[0], r[1], r[2], r[3])
        curs.execute("select code, sense, indexs, enterprise from dynamics")
        data = curs.fetchall()
        for r in data: self.getArch().createDynamics(r[0], r[1],
        self.getArch().getIndex(int(r[2])), self.getArch().getEnterprise(int(r[3])))
        conn.close()
    def write(self):
        conn = db.connect(self.getOut())
        curs = conn.cursor()
        curs.executescript(emptydb)
        for a in self.getArch().getEnterpriseList():
            curs.execute("insert into enterprise(code, name, phone, contact)
            values('%s','%s','%s','%s')"%(
            str(a.getCode()),a.getName(), a.getPhone(), a.getContact()))
        for p in self.getArch().getIndexList():
            curs.execute("insert into indexs(code, name, importance, unit)
            values('%s','%s','%s','%s')"%(
            str(p.getCode()), p.getName(), str(p.getImportance()), p.getUnit()))
        for c in self.getArch().getDynamicsList():
            if c.getEnterprises(): ent = c.getEnterpriseCode()
            else: ent = "NULL"
            if c.getIndex(): inx = c.getIndexCode()
            else: inx = "NULL"
            curs.execute("insert into dynamics(code, sense, indexs, enterprise)
            values('%s','%s','%s','%s')" % (
                str(c.getCode()), str(c.getSense()), inx, ent))
        conn.commit()
        conn.close()
```

Для проверки работы написаных функций нужна база данных в формате SQLite. Без установки дополнительного программного обеспечения ее можно получить следующим образом. Пишется программа в файле «main3.py», которая будет создавать объекты классов «achive», «datasql» и «dataxml».

```
from archive import archive
from datasql import datasql
from dataxml import dataxml
import os

arch1=archive()
arch2=archive()
dxml1 = dataxml(arch1, "old.xml", "new.xml")
dxml2 = dataxml(arch1, "old.xml", "new.xml")
```

```
dsql1=datasql(arch1, 'new.sqlite', 'new.sqlite')
dxml1.read()
if os.path.isfile(dsql1.getOut()):os.remove(dsql1.getOut())
dsql1.write()
dsql1.read()
dxml2.write()
```

В результате работы программы «main3.py» получается новый файл «new.xml», совпадающий с «old.xml».

```
<?xml version="1.0" ?>
<archive>
        <enterprise code="1" name="Лукойл" phone="123456789" contact="Вадим Воробьев"/>
        <enterprise code="2" name="Сбербанк" phone="900" contact="Герман Греф"/>
        <enterprise code="3" name="Tesla" phone="999777999" contact="Илон Маск"/>
        <index code="1" name="Выручка" importance="Высокая"/>
        <index code="2" name="Расходы" importance="Высокая"/>
        <index code="3" name="Индекс" importance="Высокая"/>
        <dynamics code="1" date="2023-02-12" sense="300" index="1" enterprise="1"/>
        <dynamics code="2" date="2023-02-12" sense="100" index="2" enterprise="1"/>
        <dynamics code="3" date="2023-02-12" sense="624" index="3" enterprise="1"/>
</archive>
```

Рис. 3