Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчёт**

***по Лабораторной работе №26***

по дисциплине ***«Языки программирования»***

Исполнитель: Колосов Артём Александрович

Группа: 10701219

Преподаватель: Лапанович Ирина Олеговна

2020

**Цель работы:** Изучить основы методологии объектно-ориентированного программирования на языке Python; приобрести объектное мышление и научиться проектировать и описывать классы соответствующей предметной области; на базе описанных классов создавать и инициализировать объекты; практически закрепить полученные знания при решении соответствующих задач.

Основное задание:

Необходимо решить задачу с использованием методологии ООП. Для чего необходимо

подобрать самостоятельно соответствующую проблемную (предметную/доменную) область, которая базируется на объектах и событиях реального мира (примеры соответствующих предметных областей приведены ниже). Спроектировать классы (пользовательские типы данных) в языке Python для программного представления данных объектов и основной логики будущей программной системы. Программа должна решать, как

минимум, два полезных действия и иметь дополнительно следующие вещи:

* не менее 3 разнообразных классов предметной области;
* не менее 5 атрибутов (состояния) в каждом соответствующем классе-сущности;
* не менее 3 методов (поведения) в соответствующих функциональных классах;
* хранить глобальные характеристики системы или характеристики уровня отдельных классов.

Требования к выполнению:

1) Необходимо спроектировать UML-диаграмму взаимодействия классов и объектов создаваемой программной системы. На базе данной UML-диаграммы реализовать рабочее приложение с использованием архитектурного шаблона проектирования MVC.

2) Каждый класс разрабатываемого приложения должен иметь адекватное осмысленное

имя (обычно это имя существительное). Имена полей и методов должны нести также

логический смысл (имя метода, который что-то вычисляет, обычно называют глаголом,

а поле – именем существительным). Имя класса пишется с большой (заглавной) буквы,

а имена методов и переменных – с маленькой (строчной).

3) Соответствующие классы должны группироваться по модулям, которые затем подключаются там, где происходит создание объектов классов и их использование.

4) При проектировании классов необходимо придерживаться принципа единственной ответственности (Single Responsibility Principle), т.е. классы должны проектироваться и

реализовываться таким образом, чтобы они были слабо завязаны с другими классами

при своей работе – они должны быть самодостаточными.

5) При выполнении заданий необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый, легко поддерживаемый и читаемый код.

6) В соответствующих компонентах (классах, функциях) бизнес-логики необходимо

предусмотреть «защиту от дурака».

7) Рекомендуется избегать использования глобальных переменных.

8) Там, где это необходимо, необходимо обеспечить грамотную обработку исключительных ситуаций, которые могут произойти при выполнении разработанной программы.

9) Все действия, связанные с демонстрацией работы приложения, должны быть размещены в главном модуле программы в функции main. При проверки работоспособности

приложения необходимо проверить все тестовые случаи.

10) Программы должны обязательно быть снабжены комментариями на английском языке,

в которых необходимо указать краткое предназначение программы, номер лабораторной

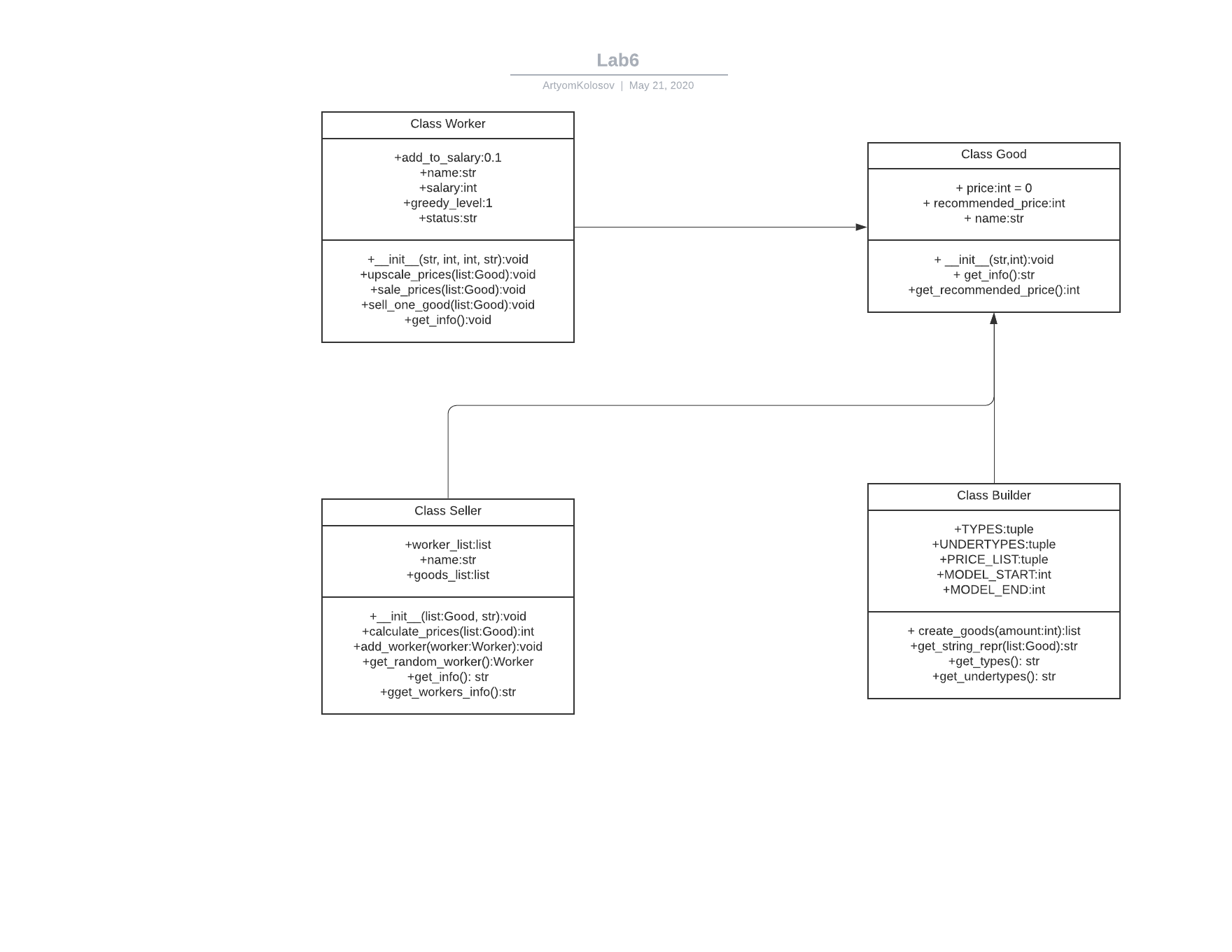
работы и её название, версию программы, ФИО разработчика, номер группы и дату разработки. Исходный текст программного кода и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.

11) Программа должна быть снабжены дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем. Интерфейс программы должен быть на английском языке.

12) При разработке программы придерживайтесь соглашений по написанию кода на Python

(Python Code Convention)

UML Диаграмма программы:



**Рис 26.1 UML-диаграмма программы**

Результаты выполнения основного задания:

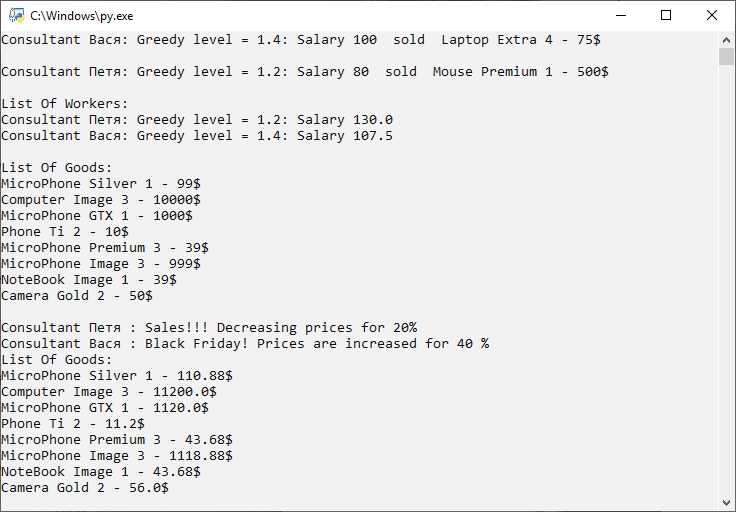


Рис 26.2 Результат выполнения основного задания

Ответы на контрольные вопросы:

1. Что такое ООП? Опишите базовые концепции, которые лежат в основе данной методологии программирования.

*Объектно-ориентированное программирование (сокращенно ООП) — это парадигма разработки программных систем, в которой приложения состоят из объектов.*

*Объекты — это сущности, у которых есть свойства и поведение. Обычно объекты являются экземплярами какого-нибудь класса.*

*1) абстракция; 2) инкапсуляция; 3) наследование; 4) полиморфизм; 5) посылка сообщений (вызов соответствующих методов, свойств и т.д.); 6) повторное использование кода.*

2. Для чего было создано ООП? На каких принципах базируется

ООП? Опишите их основные идеи?

*ООП было создано для облегчения написания компьютерных программ.*

*Каждый объект имеет состояние, обладает некоторым хорошо определённым поведением и уникальной идентичностью.*

*Состояние (state) (синонимы: параметры, аспекты, характеристики, свойства, атрибуты, …) – совокупный результат поведения объекта: одно из стабильных условий, в которых объект может существовать, охарактеризованных количественно;*

*Поведение (behavior) – действия и реакции объекта, выраженные в терминах передачи сообщений и изменения состояния; видимая извне и воспроизводимая а ктивность объекта.*

*Уникальность (identity) – эта природа объекта; то, что отличает один объект от других. В машинном представлении уникальность объекта – это адрес размещения объекта в памяти. Следовательно, уникальность объекта состоит в том, что всегда можно определить, указывают две ссылки на один и тот же объект, или на разные объекты.*

3. Приведите преимущества и недостатки объектно-ориентированного подхода.

|  |
| --- |
|  |
| ***Плюсы*** | ***Минусы*** |
| *Легко читается. Не нужно выискивать в коде функции и выяснять, за что они отвечают.* | *Потребляет больше памяти. Объекты потребляют больше оперативной*  *памяти, чем примитивные типы данных.* |
| *Быстро пишется. Можно быстро создать сущности, с которыми должна работать программа.* | ***Снижает производительность.*** *Многие вещи технически реализованы иначе,*  *поэтому они используют больше ресурсов.* |
| *Проще реализовать большой функционал. Так как на написание кода уходит меньше времени, можно гораздо быстрее создать приложение с множеством возможностей.* | *Сложно начать. Парадигма ООП сложнее*  *функционального программирования,*  *поэтому на старт уходит больше времени.* |
| *Меньше повторений. Не нужно писать однотипные функции для разных сущностей* |  |

4. Что такое объект и чем характеризуются объекты в ООП?

*Объект - это понятие, абстракция или любой предмет с чётко очерченными границами, имеющий смысл в контексте рассматриваемой прикладной проблемы. Введение объекта преследует две цели: понимание прикладной задачи (проблемы) и введение основы для реализации на компьютере.*

*В ООП объект может сохранять свое состояние (информацию) и обеспечивать набор операций для проверки и изменения этого состояния.*

5. Что такое класс и зачем он нужен? Приведите общее определение класса в ООП.

Какая разница между классом и объектом в ООП? Какие разновидности классов существуют в ООП?

***Клaсс****этo основной элемент, в рамках которого осуществляется конструирование программ. Класс содержит в себе данные и кoд, который управляет этими данными.*

*Класс - в объектно-ориентированном программировании, представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов.*

*Объекты одного и того же класса имеют один и тот же набор характеристик (значение которых может быть различным у разных объектов) и общее поведение (все объекты одинаково реагируют на одинаковые сообщения).*

*Абстрактные классы — это классы для которых не существует экземпляров, они лишь описывают общие характеристики классов-потомков.*

*Конкретные классы — это классы экземпляры которых могут существовать (или существуют) в системе в отличии от абстрактных классов.*

6. Как описывается класс в Python?

*При помощи ключевого слова class*

7. Каким способом можно создать объект (экземпляр класса) в Python?

*Для создания класса, надо произвести его инициализацию, также можно присвоить экземпляр класс переменной, которая будет указателем на класс в памяти.*

*var = Class()*

8. Как можно обратиться к полям и методам объекта? В чём состоит особенность первого аргумента в методах класса?

*Обращение к методам класса происходит через ссылку на класс. Первый аргумент в методах класса является ссылка на экземпляр класса*

9. Какой стандартный метод при описании класса необходимо переопределить, чтобы

гарантировать, что в тех ситуациях, где требуется строковое представление объекта

данного класса, система автоматических вызывала данный метод у ссылочной переменной?

*Нужно переопределить строковый метод \_\_repr\_\_, который, например, будет вызываться при передаче ссылка на экземпляр класс в метод print(). Также можно переопределить метод \_\_str\_\_, который вызывается при передаче ссылке на экземпляр класса в метод str()*

10. Что такое UML? Как с помощью UML можно описать классы и их взаимосвязь друг с

другом (т.е. описать UML-диаграмму классов)?

*UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. Язык UML применяется не только для проектирования, но и с целью документирования, а также эскизирования проекта.*

Что я освоил в процессе выполнения лабораторной работы (выводы):

ООП – современная методология программирования, которая используется практически везде при написании программ различных уровней сложности. ООП позволяет спроецировать реальный мир, его наполнение и свойства в программу в виде различных объектов, каждый их которых имеет свои свойства и возможности.

*ПРИЛОЖЕНИЕ A*

Листинг исходных кодов программ

# Файл Builder.py

# Класс Builder, утилитный класс для

# создания списка товаров, а также

# представления данного списка в

# строковом варианте для вывода на консоль.

#

# Version: 1.0

# Group: 10701219

# Author: Колосов Артём Александрович

# Date: 16.4.2020

import random

from Goods import Good

class Builder:

TYPES = ("Computer", "NoteBook", "Laptop",

"Phone", "Mouse", "Camera",

"MicroPhone", "HeadPhones", "Charge")

UNDERTYPES = ("Reborn", "Edge", "Plus",

"Extra", "Razr", "Sharp",

"Light", "Image", "GTX",

"Ti", "Gold", "Silver",

"Premium")

PRICE\_LIST = (100, 500, 50,

10000, 1000, 75,

25, 999, 99, 10, 39)

MODEL\_START = 1

MODEL\_END = 5

def create\_goods(self, amount):

goods\_list = []

for i in range(amount):

name = random.choice(self.TYPES) + " " + random.choice(self.UNDERTYPES) + " " + \

str(random.randrange(self.MODEL\_START, self.MODEL\_END))

price = random.choice(self.PRICE\_LIST)

goods\_list.append(Good(name, price))

return goods\_list

def get\_string\_repr(self, goods\_list):

string = "List Of Goods:\n"

for good in goods\_list:

if isinstance(good, Good):

string += good.get\_info()

return string

def get\_types(self):

info = ""

for typ in self.TYPES:

info += typ + "\n"

return info

def get\_undertypes(self):

info = ""

for undertype in self.UNDERTYPES:

info += undertype + "\n"

return info

# Файл Goods.py

# Класс Good, класс-сущность, описывающий существенные характеристики

# товара

#

# Version: 1.0

# Group: 10701219

# Author: Колосов Артём Александрович

# Date: 16.4.2020

class Good:

price = 0

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

self.name = name

self.recomended\_price = recomended\_price

self.price = recomended\_price

def get\_info(self):

return "{0} - {1}$\n".format(self.name, self.price)

def get\_recomended\_price(self):

return self.recomended\_price

# Файл main\_sc.py

# Главный Файл, который собирает

# все классы и позволяет с ними работать

#

# Version: 1.0

# Group: 10701219

# Author: Колосов Артём Александрович

# Date: 16.4.2020

from Builder import Builder

from Sell import Seller

from Worker import Worker

def main():

build = Builder()

goods\_list = build.create\_goods(10)

eldorado = Seller(goods\_list, "eldorado")

worker\_vasya = Worker("Вася", salary=100, greedy\_level=1.4)

worker\_petya = Worker("Петя", salary=80, greedy\_level=1.2)

eldorado.add\_worker(worker\_petya)

eldorado.add\_worker(worker\_vasya)

worker\_vasya.sell\_one\_good(eldorado.goods\_list)

worker\_petya.sell\_one\_good(eldorado.goods\_list)

print(eldorado.get\_workers\_info())

print(build.get\_string\_repr(eldorado.goods\_list))

worker\_petya.sale\_prices(eldorado.goods\_list, 0.8)

worker\_vasya.upscale\_prices(eldorado.goods\_list)

print(build.get\_string\_repr(eldorado.goods\_list))

input()

if \_\_name\_\_ == "\_\_main\_\_":

main()

# Файл Sell.py

# Класс Seller, который работает с классами Worker, Good

# Класс Seller, общий класс, в котором описываются и существенные

# характеристики работника, товара

# Version: 1.0

# Group: 10701219

# Author: Колосов Артём Александрович

# Date: 16.4.2020

from Goods import Good

from Worker import Worker

import random

class Seller:

worker\_list = []

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

self.goods\_list = goods\_list

self.name = name

def calculate\_prices(self, goods\_list):

all\_price = 0

for good in goods\_list:

if isinstance(good, Good):

all\_price += good.price

return all\_price

def add\_worker(self, worker):

if isinstance(worker, Worker):

self.worker\_list.append(worker)

def get\_random\_worker(self):

return random.choice(self.worker\_list)

def get\_info(self):

return self.name

def get\_workers\_info(self):

result = "List Of Workers:\n"

for worker in self.worker\_list:

if isinstance(worker, Worker):

result += worker.get\_info() + "\n"

return result

# Файл Worker.py

# Класс Worker, функциональный класс, в котором описывается

# основная бизнес логика приложения

#

# Version: 1.0

# Group: 10701219

# Author: Колосов Артём Александрович

# Date: 16.4.2020

from Goods import Good

import random

class Worker:

add\_to\_salary = 0.1

def \_\_init\_\_(self, name, salary, greedy\_level=1, status="Consultant"):

self.name = name

self.status = status

self.salary = salary

self.greedy\_level = greedy\_level

def upscale\_prices(self, goods\_list):

print(self.status, self.name, ": Black Friday! Prices are increased for {0} %".format(round((self.greedy\_level - 1) \* 100)))

for good in goods\_list:

if isinstance(good, Good):

good.price = round(good.price \* self.greedy\_level, 2)

def sale\_prices(self, goods\_list, sale\_coef=0.5):

if 0 < sale\_coef < 1:

print(self.status, self.name, ": Sales!!! Decreasing prices for {0}%".format(round((1-sale\_coef) \* 100)))

for good in goods\_list:

if isinstance(good, Good):

good.price = round(good.price \* sale\_coef, 2)

else:

print("What?")

def sell\_one\_good(self, goods\_list):

delete\_index = random.randint(0, len(goods\_list)-1)

if isinstance(goods\_list[delete\_index], Good):

good = goods\_list.pop(delete\_index)

print(self.get\_info(), " sold ", good.get\_info())

self.salary += (self.add\_to\_salary \* good.price)

def get\_info(self):

return "{0} {1}: Greedy level = {2}: Salary {3}".format(self.status,

self.name,

self.greedy\_level,

self.salary)