Министерство образования Республики Беларусь Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

### Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**Отчёт**

### по лабораторной работе № 1

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

## тема: «Python и ООП. Проектирование и реализация классов.

**Создание и манипулирование объектами»**

|  |  |
| --- | --- |
| Исполнитель: | студент группы 10701118 Воробей Иван Александрович |
| Преподаватель: | Лапакович И.О. |

2018-2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 1**

# PYTHON И ООП. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ КЛАССОВ. СОЗДАНИЕ И МАНИПУЛИРОВАНИЕ

**ОБЪЕКТАМИ**

#### Цель работы:

Изучить основы объектно-ориентированного программирования и приобрести *объектное мышление* для использования всей силы и мощи ООП на примере проектирования и реализации классов и объектов в Python.

#### Основное задание:

Необходимо выбрать объекты реального мира и спроектировать классы (пользовательские типы данных) в Python для программного представления данных объектов (объект выбирается самостоятельно, но согласовывается с преподавателем). У объектов должно быть не менее 5 атрибутов (характеристик) и не менее трёх методов.

Написать программу для создания объектов спроектированных классов и демонстрации взаимодействия между ними.

#### Требования к выполнению:

1. Программа должна обязательно быть снабжена 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 раз- работки.
2. Каждый класс должен иметь адекватное осмысленное имя (обычно это *имя существи- тельное*) и начинаться с заглавной буквы. Имена полей и методов должны начинаться с маленькой буквы и быть также осмысленными (имя метода, который что-то вычисляет, обычно называют *глаголом*, а поле – именем существительным).
3. Каждый класс необходимо разместить в отдельном модуле, который затем подключа- ется в другом модуле, где происходит создание объекта данного класса и его использо- вание.
4. При проектировании классов необходимо придерживаться принципа единственной от- ветственности (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 (стремитесь всегда разрабаты- вать масштабируемый и универсальный код).
5. Программа для демонстрации работоспособности разработанных классов должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.
6. В отчёте ОБЯЗАТЕЛЬНО привести UML-диаграмму классов, которая демонстрирует классы и объекты приложения, их атрибуты и методы, а также взаимосвязь между ними.

#### Описание предметной области:

Предметной (проблемной) областью является работа кофейни, подсчет количества заказанного кофе и оплата.

В результате анализа предметной области были выделены следующие классы:

* + студент (Customer) – общий класс, описывающий существенные характеристики покупателя и некоторую логику покупки кофе;
  + преподаватель (Barista) – общий класс, в котором описываются и существенные характеристики бариста, и некоторая логика продажи кофе, которая зависит от соответствующего состояния (параметров) преподавателя;
  + Всевышний (God) – утилитный класс для создания списка покупателей, а также представления данного списка в строковом варианте для вывода на консоль.

Ниже на рисунке 1 приведена UML-диаграмма классов заданной предметной (проблемной) области.

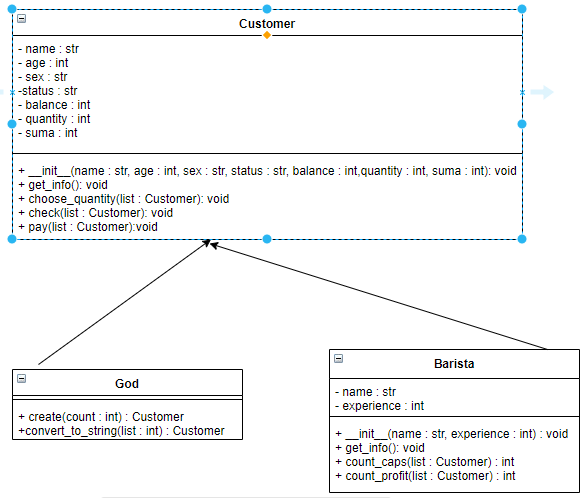


Рисунок 1 – UML-диаграмма предметной области

#### Результаты выполнения основного задания:

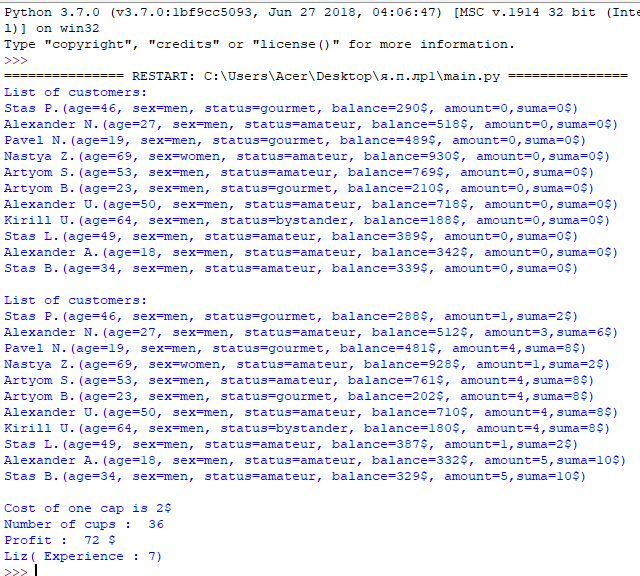


Рисунок 2 – Результат тестирования и работы простейшей ООП-программы на Python

#### Ответы на контрольные вопросы:

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

Методология программирования , основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классф образуют иерархию наследования.Главная концепция- все состоит из объектов.

1. Для чего было создано ООП? На каких принципах базируется ООП? Опишите их основные идеи?

Для улучшения управляемости процесса моделирования. Абстрагирование для выделения в моделируемом предмете важного для решения конкретной задачи по предмету. Инкапсуляция, наследование, полиморфизм.

1. Приведите преимущества и недостатки объектно-ориентированного подхода.

Преимущества: возможность легкой модификации, возможность отката при наличии версий, болие легкая расширяемость, сокращение количества межмодульных вызовов, повторное использывание кода.

Недостатки:резко увиличивается время на анализ и проектирование систем, сложность распределения работ на начальном этапе, себестоймость больше.

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

Объект в ооп – это объект реального мира. Сущность, способная сохранять свое состояние и обеспечивающая набор операций для проверки и изменения этого сосотояния.

1. 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?

Класс – это элемент ПО, описывающий абстрактный тип данных и его частичную или полную реализацию. Набор полей, атрибутов, методов. Объект это экземпляр класса, т.е. класс это как бы заготовка, а уже объект это уже «вещь», с которой можно непосредственно работать. Основной класс, утилитный, класс-сущность.

1. Как описывается класс в Python? Каким способом можно создать объект (экземпляр класса) в Python?

сlass ИмяКласса:

Код тела класса

имя\_переменной=ИмяКласса()

1. Как в терминах ООП называется метод, который вызывается сразу после создания объекта для инициализации его состояния. Как его описать в классе?

Конструктор.

def \_\_init\_\_(self, a, b):

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

Доступ к атрибутам выполняется по схеме obj.attrname .

имя\_переменной\_класса.имя\_метода()

self-первый пораметр в описании метода. Ссылается на конкретный экземпляр класса а не на класс в целом. Это не зарезервированное слово , это просто название переменной. Зачем это нужно? Для изменения\создания полей self.название\_поля=700 .

1. 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?

\_\_str\_\_ — возвращает строковое представление объекта

1. Что такое UML? Как с помощью UML можно описать классы и их взаимосвязь друг с другом (т.е. описать UML-диаграмму классов)?

UML-язык графического описания для объектного моделирования в области разработки программного обеспечения. UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы.

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

1. Методологию и фундаментальные основы, на которых базируется мир объектно- ориентированного программирования (ООП).
2. Основные столпы (парадигмы) ООП.
3. Преимуществами и недостатками ООП методологии.
4. Характеристики, которые есть у всех объекта в ООП-мире.
5. Процесс описания классов в Python и создание на базе них экземпляров (объектов), а также организация их взаимодействия друг с другом.
6. Правила описания классов, полей и методов.
7. Разницу между непосредственно самим объектом, который создаётся в динамической памяти (heap) и ссылочной переменной, через которую осуществляется манипулирование созданным объектом.
8. Разницу между ссылочными объектами изменяемых и неизменяемых типов.

*ПРИЛОЖЕНИЕ A*

#### Листинг исходных кодов классов и программы в целом Файл customer*.py*

*import random*

*class Customer:*

*def \_\_init\_\_(self, name, age,sex,status,balance,quantity=0,suma=0):*

*self.name=name*

*self.age=age*

*self.sex=sex*

*self.status=status*

*self.balance=balance*

*self.quantity=quantity*

*self.suma=suma*

*def get\_info(self):*

*return (self.name+"(age="+str(self.age)+*

*", sex="+self.sex+ ", status="+*

*self.status+", balance="+str(self.balance)+"$"+*

*", amount="+str(self.quantity)+*

*",suma="+str(self.suma)+"$"+")")*

*def choose\_quantity(self, customers\_list):*

*for cus in customers\_list:*

*cus.quantity=random.randint(1,5)*

*def check(self,customers\_list):*

*for cus in customers\_list:*

*cus.suma=2\*cus.quantity*

*def pay(self, customers\_list):*

*for cus in customers\_list:*

*cus.balance=cus.balance-cus.suma*

***Файл barista.py***

*from customer import Customer*

*class Barista:*

*def \_\_init\_\_(self,name, experience=1):*

*self.name=name*

*self.experience=experience*

*def get\_info(self):*

*return(self.name+"( Experience : "+str(self.experience)+")")*

*def count\_cups(self, customers\_list):*

*c=0*

*for cus in customers\_list:*

*c=c+cus.quantity*

*return c*

*def count\_profit(self, customers\_list):*

*b=0*

*for cus in customers\_list:*

*b=b+cus.suma*

*return b*

##### Файл god.py

*import random*

*from customer import Customer*

*class God:*

*START\_ALPHABET\_WITH\_UPPER\_LETTER=65*

*END\_ALPHABET\_WITH\_UPPER\_LETTER=91*

*def create(self, count):*

*namesm=["Alexander","Pavel","Artyom","Kirill","Stas"]*

*namesw=["Olya","Liza","Nastya","Katia"]*

*sexes=["men","women"]*

*statuses=["amateur","bystander","gourmet"]*

*list\_of\_customers=[]*

*for i in range(count+1):*

*sex=random.choice(sexes)*

*if sex=="men":*

*name=random.choice(namesm)*

*name+=" " + chr(random.randint(*

*God.START\_ALPHABET\_WITH\_UPPER\_LETTER,*

*God.END\_ALPHABET\_WITH\_UPPER\_LETTER))+"."*

*else:*

*name=random.choice(namesw)*

*name+=" " + chr(random.randint(*

*God.START\_ALPHABET\_WITH\_UPPER\_LETTER,*

*God.END\_ALPHABET\_WITH\_UPPER\_LETTER))+"."*

*balance=random.randint(20,1000)*

*age=random.randint(16,70)*

*status=random.choice(statuses)*

*customer=Customer(name,age,sex,status,balance)*

*list\_of\_customers.append(customer)*

*return list\_of\_customers*

*def convert\_to\_string(self,list\_of\_customers):*

*string="List of customers:\n"*

*for customer in list\_of\_customers:*

*if isinstance(customer, Customer):*

*string+=str(customer.get\_info())+"\n"*

*return string*

##### Файл main.py

##### from god import God

##### from barista import Barista

##### from customer import Customer

##### def main():

##### god=God()

##### cus=Customer(1,2,3,4,5,6,7)

##### customers\_list=god.create(10)

##### barista=Barista("Liz", 7)

##### print(god.convert\_to\_string(customers\_list))

##### cus.choose\_quantity(customers\_list)

##### cus.check(customers\_list)

##### cus.pay(customers\_list)

##### print(god.convert\_to\_string(customers\_list))

##### print("Cost of one cap is 2$")

##### print("Number of cups : ",barista.count\_cups(customers\_list))

##### print("Profit : ",barista.count\_profit(customers\_list),"$")

##### print(barista.get\_info())

##### 

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

##### main()