Оглавление

[Оглавление 3](#_Toc469871309)

[1. Введение 4](#_Toc469871310)

[2. Задания 5](#_Toc469871311)

[2.1. Объектно-ориентированное программирование и классы 5](#_Toc469871312)

[2.1.1. Задача 11.1 5](#_Toc469871313)

[Условие задачи 5](#_Toc469871314)

[Исходный код 7](#_Toc469871315)

[Иллюстрация работы 7](#_Toc469871316)

[2.1.2. Задача 11.2 13](#_Toc469871317)

[Условие задачи 13](#_Toc469871318)

[Исходный код 16](#_Toc469871319)

[Иллюстрация работы 17](#_Toc469871320)

[2.1.3. Задача 11.3 20](#_Toc469871321)

[Условие задачи 20](#_Toc469871322)

[Исходный код 20](#_Toc469871323)

[Иллюстрация работы 20](#_Toc469871324)

[2.2. Стандартная библиотека 21](#_Toc469871325)

[2.2.1. Задача 12.1 21](#_Toc469871326)

[Условие задачи 21](#_Toc469871327)

[Исходный код 22](#_Toc469871328)

[Иллюстрация работы 22](#_Toc469871329)

[2.2.2. Задача 12.2 23](#_Toc469871330)

[Условие задачи 23](#_Toc469871331)

[Исходный код 23](#_Toc469871332)

[Иллюстрация работы 24](#_Toc469871333)

[2.2.3. Задача 12.3 24](#_Toc469871334)

[Условие задачи 24](#_Toc469871335)

[Исходный код 25](#_Toc469871336)

[Иллюстрация работы 25](#_Toc469871337)

[3. Заключение 28](#_Toc469871338)

[Список источников 29](#_Toc469871339)

1. Введение

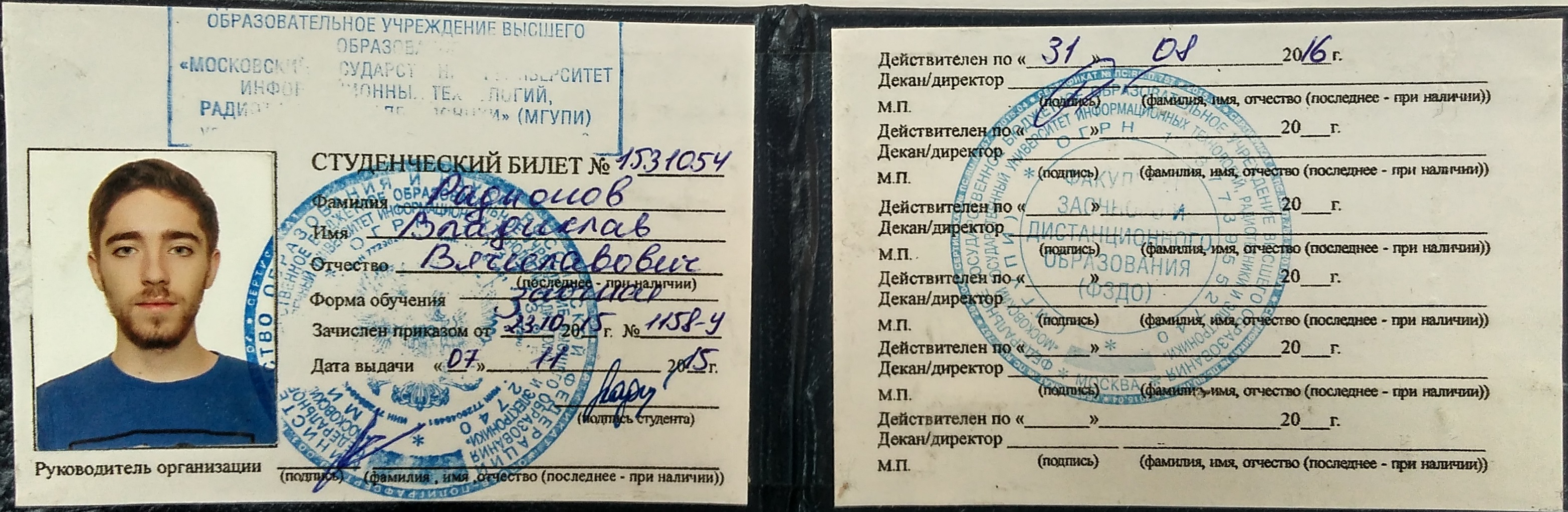
Документ содержит описание выполненных работ по Курсовому проекту.

Информация о выполненных заданиях содержится в Таблице 1.

Для вариативной части заданий фотография (скан) студенческого билета приведена на Рисунке 1.

Таблица 1 — Таблица выполненных заданий

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **11.1** | **11.2** | **11.3** | **12.1** | **12.2** | **12.3** | **13.1** | **13.2** | **13.3** | **13.4** | **14.1** | **14.2** | **14.3** |
| Выполнено | **+** | **+** | **+** | **+** | **+** | **+** |  |  |  |  |  |  |  |
| Зачтено |  |  |  |  |  |  |  |  |  |  |  |  |  |



**Рисунок 1 —** Фото студенческого билета

Исходный код задач расположен по ссылке: <https://drive.google.com/drive/u/0/folders/0B-tinlkly_eER2xkVzNrcGlhUkk>.

2. Задания

2.1. Объектно-ориентированное программирование и классы

2.1.1. Задача 11.1

Условие задачи

Спроектировать и реализовать классы Time (Время) и Fraction (Математическая дробь), предусмотреть необходимые атрибуты и методы.

1. Класс Time

Класс Time содержит следующие атрибуты:

* \_hour (int) – значение часа (0-23)
* \_minute (int) – значение минуты (0-59)
* \_second (int) – значение секунды (0-59)

Класс Time содержит следующие константы:

* MIN\_VALUE = 0 – минимальное значение для всех атрибутов класса
* SEC\_MINUTE\_MAX = 59 – максимальное значение для атрибутов «\_minute» и «\_second»
* HOUR\_MAX = 23 – максимальное значение для атрибута «\_hour»

Класс Time содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, hour, minute, second) – конструктор класса
* from\_string(time string) – создает экземпляр класса на основании переданной строки (статический конструктор)

1. специальные методы:

* \_\_eq\_\_(self, other) – метод сравнения времени (одинаковые ли значения)
* \_\_ne\_\_(self, other) – метод сравнения времени (разные ли значения)
* \_\_lt\_\_(self, other) – метод сравнения времени (меньше ли время self, чем время other)
* \_\_gt\_\_(self, other) – метод сравнения времени (больше ли время self, чем время other)
* \_\_le\_\_(self, other) – метод сравнения времени (время self меньше или равно времени other или нет)
* \_\_ge\_\_(self, other) – метод сравнения времени (время self больше или равно времени other или нет)
* \_\_add\_\_(self, other) – операция сложения для класса Time
* \_\_sub\_\_(self, other) – операция вычитания для класса Time
* \_\_str\_\_(self) – возвращает строковое представление класса

1. прочие методы:

* check\_hour(hour) – возвращает ответ, является ли переданное значение часа корректным (статический метод)
* check\_minute(minute) – возвращает ответ, является ли переданное значение минуты корректным (статический метод)
* check\_second(second) – возвращает ответ, является ли переданное значение секунды корректным (статический метод)
* set\_hour(self, hour) – метод, устанавливающий значение часа
* set\_minute(self, minute) – метод, устанавливающий значение минуты
* set\_second(self, second) – метод, устанавливающий значение секунды
* get\_hour(self) – метод, возвращающий значение часа
* get\_minute(self) – метод, возвращающий значение минуты
* get\_second(self) – метод, возвращающий значение секунды

1. Класс Fraction

Класс Fraction содержит класс типа перечисление (Enum) с именем «Type».

Класс Fraction.Type содержит следующие значения, отражающие тип дроби:

* simple = 1 – обыкновенная дробь
* mixed = 2 – смешанная дробь
* decimal = 3 – десятичная дробь

Класс Fraction.Type содержит следующие методы:

* \_\_str\_\_(self) – возвращает строковое представление класса
* describe(self) – возвращает описание типа дроби

Класс Fraction содержит следующие атрибуты:

* \_numerator (int) – числитель дроби (применимо для типов simple и mixed)
* \_denominator (int) – знаменатель (применимо для типов simple и mixed)
* \_negative (bool) – показатель отрицательности дроби
* \_integer (int) – целая часть (применимо для типа mixed)
* \_decimal (float) – значение десятичной дроби (применимо для типа decimal)
* \_type (Fraction.Type) – показатель типа дроби

Класс Fraction содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, numerator, denominator, negative=False, integer=0, decimal=None) – конструктор класса
* from\_decimal(decimal) – создает экземпляр класса Fraction с типом decimal на основании переданного значения (статический конструктор)
* from\_string(time\_string) – создает экземпляр класса на основании переданной строки (статический конструктор)

1. специальные методы:

* \_\_eq\_\_(self, other) – метод сравнения дроби (одинаковые ли значения)
* \_\_ne\_\_(self, other) – метод сравнения дроби (разные ли значения)
* \_\_lt\_\_(self, other) – метод сравнения дроби (меньше ли дробь self, чем дробь other)
* \_\_gt\_\_(self, other) – метод сравнения дроби (больше ли дробь self, чем дробь other)
* \_\_le\_\_(self, other) – метод сравнения дроби (дробь self меньше или равно дроби other или нет)
* \_\_ge\_\_(self, other) – метод сравнения дроби (дробь self больше или равно дроби other или нет)
* \_\_add\_\_(self, other) – операция сложения для класса Fraction
* \_\_sub\_\_(self, other) – операция вычитания для класса Fraction
* \_\_neg\_\_(self) – операция отрицания для класса Fraction
* \_\_mul\_\_(self, other) – операция умножения для класса Fraction
* \_\_truediv\_\_(self, other) – операция деления для класса Fraction
* \_\_floordiv\_\_(self, other) – операция целочисленного деления для класса Fraction
* \_\_mod\_\_(self, other) – операция получения остатка от деления для класса Fraction
* \_\_pow\_\_(self, other) – операция возведения в степень для класса Fraction
* \_\_str\_\_(self) – возвращает строковое представление класса

1. прочие методы:

* \_\_nod\_\_(a, b) – находит наибольший общий делитель для переданных значений (статический метод)
* \_\_nok\_\_(a, b) – находит наименьшее общее кратное для переданных значений (статический метод)
* reduce(self) – сокращает дробь, если это возможно
* abs(self) – операция получения абсолютной дроби для класса Fraction
* get\_type(self) – возвращает тип дроби
* to\_simple(self) – создает экземпляр класса Fraction с типом simple путем преобразования переданной дроби self
* to\_mixed(self) – создает экземпляр класса Fraction с типом mixed путем преобразования переданной дроби self
* to\_decimal(self) – создает экземпляр класса Fraction с типом decimal путем преобразования переданной дроби self
* to\_integer(self) – преобразует дробь в целое число, если это возможно
* get\_integer(self) – возвращает целую часть дроби
* get\_numerator(self) – возвращает числитель дроби
* get\_denominator(self) – возвращает знаменатель дроби

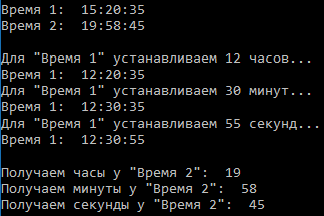
Исходный код

Исходный код приложения находится в папке «11.1» и содержится в следующих файлах:

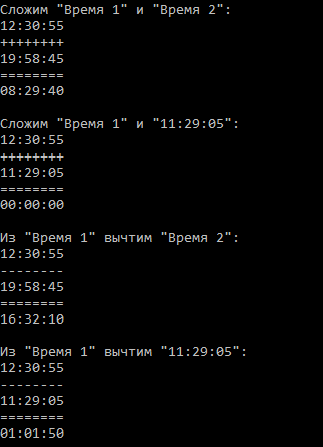
* «mytime.py»: описание и реализация класса Time
* «main\_time.py»: демонстрация работы класса Time
* «fraction.py»: описание и реализация класса Fraction
* «main\_fraction.py»: демонстрация работы класса Fraction

Иллюстрация работы

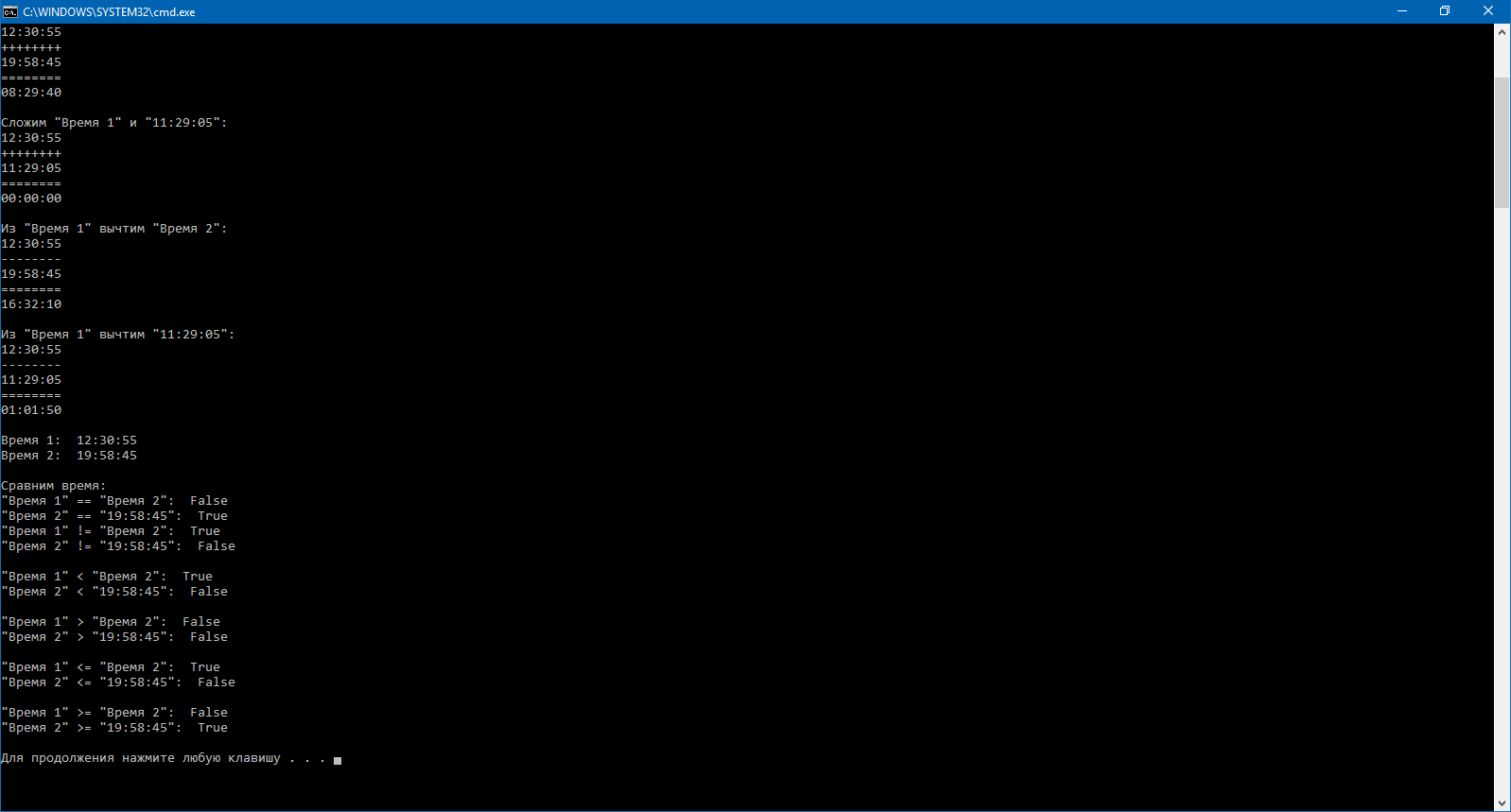
Иллюстрация работы с классом Time приведена на Рисунках 2.1 – 2.3.



**Рисунок 2.1 —** Иллюстрация работы методов класса Time

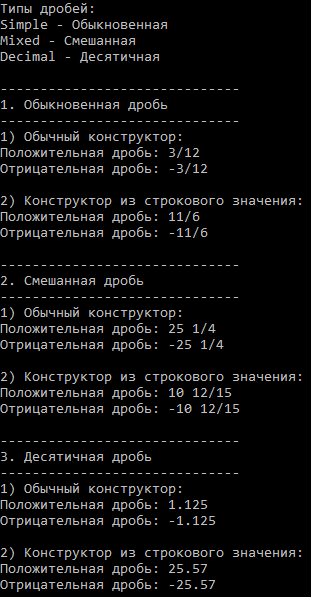


**Рисунок 2.2 —** Иллюстрация работы методов сложения и вычитания класса Time

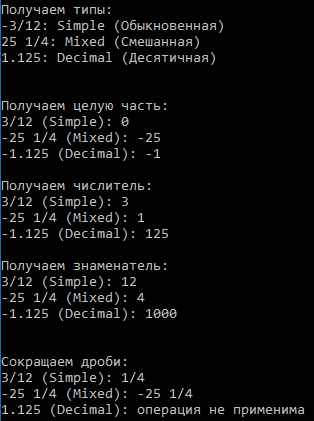


**Рисунок 2.3 —** Иллюстрация работы методов сравнения класса Time

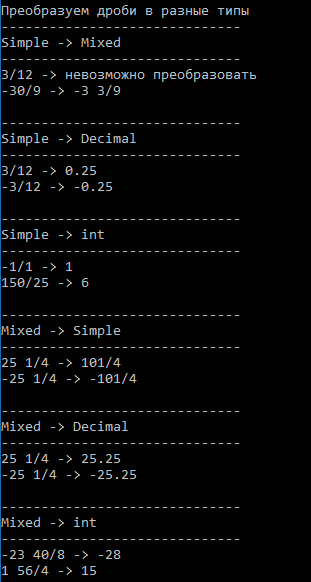
Иллюстрация работы с классом Fraction приведена на Рисунках 2.4 – 2.11.



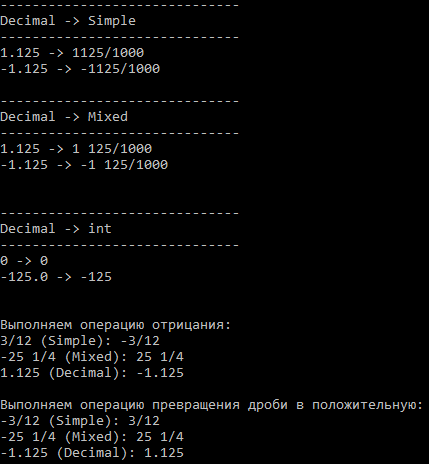
**Рисунок 2.4 —** Иллюстрация работы конструкторов класса Fraction



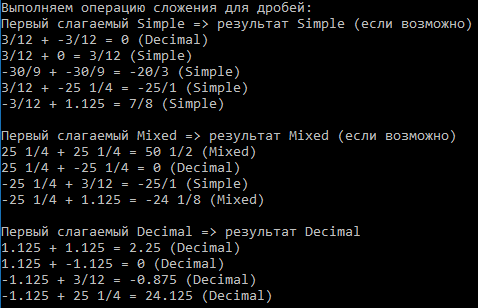
**Рисунок 2.5 —** Иллюстрация работы методов класса Fraction



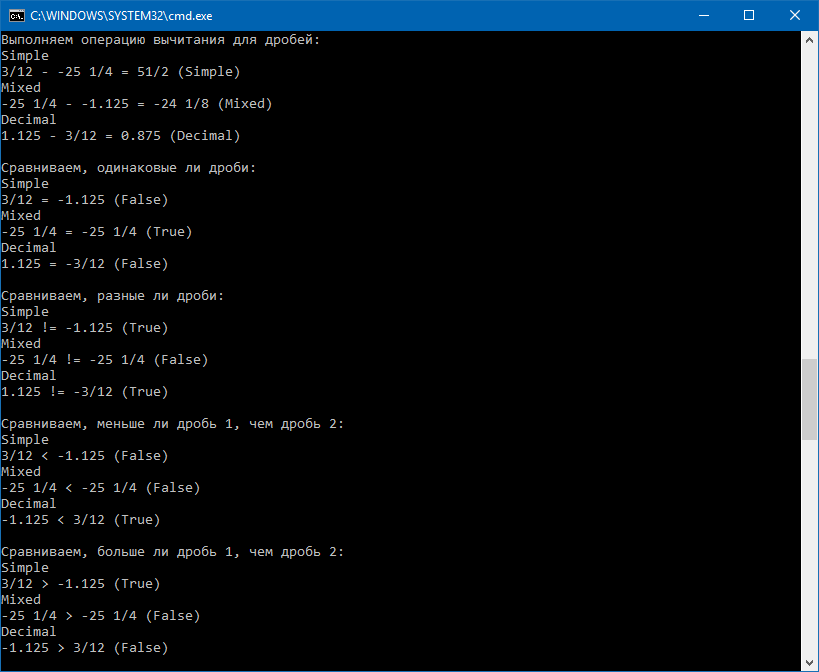
**Рисунок 2.6 —** Иллюстрация работы методов преобразования класса Fraction



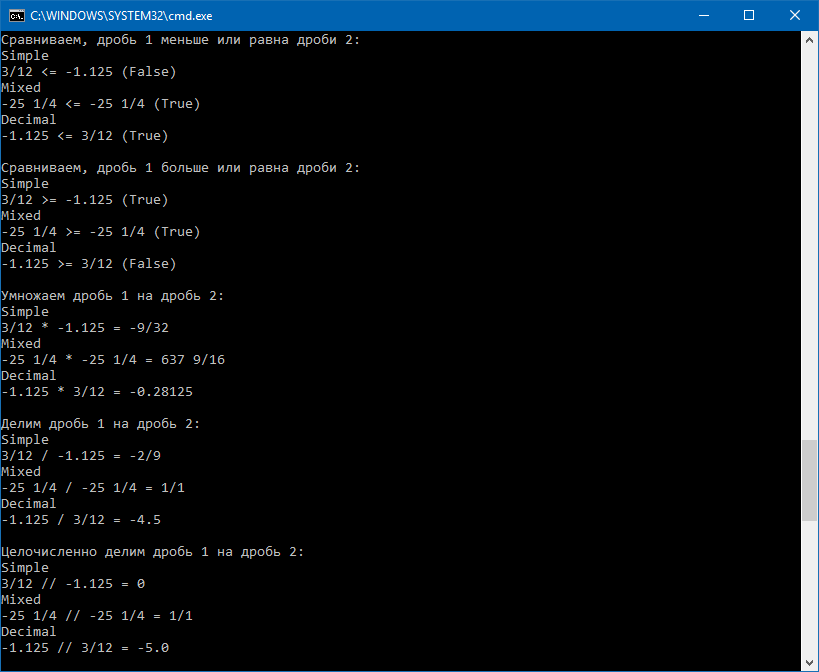
**Рисунок 2.7 —** Иллюстрация работы методов класса Fraction



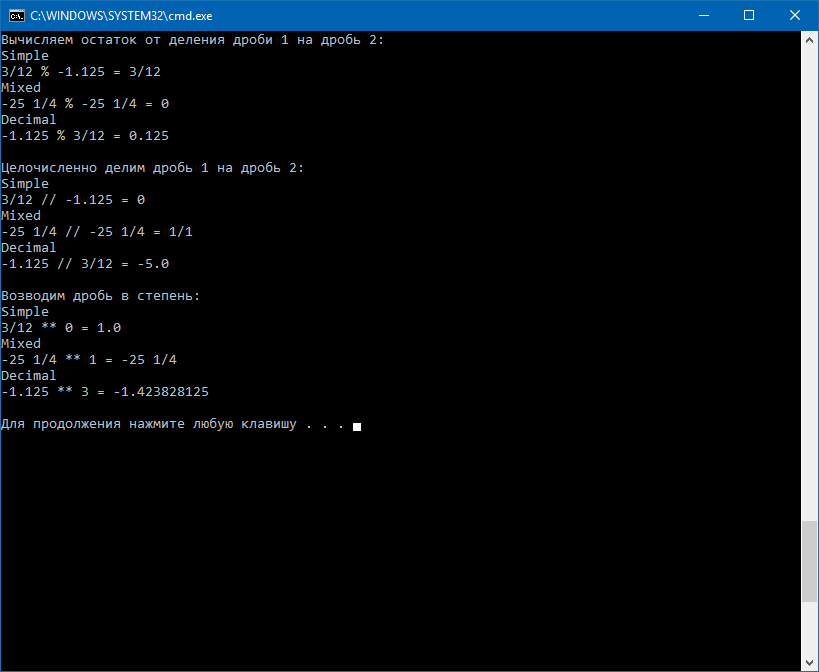
**Рисунок 2.8 —** Иллюстрация работы метода сложения класса Fraction



**Рисунок 2.9 —** Иллюстрация работы методов класса Fraction



**Рисунок 2.10 —** Иллюстрация работы методов класса Fraction



**Рисунок 2.11 —** Иллюстрация работы методов класса Fraction

2.1.2. Задача 11.2

Условие задачи

Выстроить объектную иерархию (от наиболее общего класса к наиболее частному). Спроектировать и реализовать классы, предусмотреть необходимые атрибуты и методы.

Я выбрал объектную иерархию, отражающую классификацию компьютерных устройств. Объекты и соответствующие классы представлены на Рисунке 2.12.

**Рисунок 2.12** — Объектная иерархия

1. Класс Computer

Класс Computer содержит следующие атрибуты:

* name (str) – наименование устройства (пользовательское)
* manufacturer (str) – производитель
* model (str) – наименование модели
* \_builtin\_memory (int) – объём встроенной памяти
* \_ram\_memory (int) – объём оперативной памяти

Класс Computer и наследуемые классы содержит следующие константы:

* UNDEFINED (str) – значение характеристик устройств по умолчанию

Класс Computer содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, name, manufacturer, model) – конструктор класса
* load(cls, filename) – создает экземпляр класса, загружая данные из файла в формате JSON

1. специальные методы:

* \_\_str\_\_(self) – возвращает строковое представление класса

1. прочие методы:

* builtin\_memory(self) – @property – метод возвращает значение атрибута \_builtin\_memory
* builtin\_memory(self, value) – @builtin\_memory.setter – метод устанавливает значение атрибута \_builtin\_memory
* ram\_memory(self) – @property – метод возвращает значение атрибута \_ram\_memory
* ram\_memory(self, value) – @ram\_memory.setter – метод устанавливает значение атрибута \_ram\_memory
* print\_specifications(self) – метод, печатающий характеристики устройства
* save(self, filename) – метод, сохраняющий данные об объекте в файл формата JSON

1. Класс Smartphone

Класс Smartphone наследуется от класса Computer. Здесь и далее представлены данные, которые были добавлены в класс или переопределены.

Класс Smartphone содержит класс OperatingSystem, который, в свою очередь, содержит класс типа перечисление (Enum) с именем «Type».

В классе OperatingSystem.Type содержатся следующие значения, отражающие тип ОС:

* Android = 1
* iOS = 2
* WindowsPhone = 3
* BlackBerry = 4
* SailfishOS = 5
* Tizen = 6
* UbuntuTouch = 7

Класс OperatingSystem.Type содержит следующий метод:

* \_\_str\_\_(self) – возвращает строковое представление класса

Класс OperatingSystem содержит следующие атрибуты:

* os\_type (OperatingSystem.Type) – тип ОС
* os\_version (str) – версия ОС

Класс OperatingSystem содержит следующие методы:

* \_\_init\_\_(self, os\_type, os\_version) – конструктор класса
* \_\_str\_\_(self) – возвращает строковое представление класса

В класс Smartphone добавлены следующие атрибуты:

* mobile\_os (OperatingSystem) – данные об ОС
* \_sim\_type (str) – тип SIM-карты
* \_sim\_count (int) – количество слотов для SIM-карт

Класс Smartphone содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, name, manufacturer, model, mobile\_os) – конструктор класса
* load(cls, filename) – переопределённый конструктор (здесь и далее – см. описание класса Computer)

1. специальные методы:

* \_\_str\_\_(self) – переопределённый метод

1. прочие методы:

* sim\_type(self) – @property – метод возвращает значение атрибута \_sim\_type
* sim\_type(self, value) – @sim\_type.setter – метод устанавливает значение атрибута \_sim\_type
* sim\_count(self) – @property – метод возвращает значение атрибута \_sim\_count
* sim\_count(self, value) – @sim\_count.setter – метод устанавливает значение атрибута \_sim\_count
* print\_specifications(self) – переопределённый метод
* save(self, filename) – переопределённый метод

1. Класс PersonalComputer

Класс PersonalComputer наследуется от класса Computer. Здесь и далее представлены данные, которые были добавлены в класс или переопределены.

Класс PersonalComputer содержит класс OperatingSystem, который, в свою очередь, содержит класс типа перечисление (Enum) с именем «Type».

В классе OperatingSystem.Type содержатся следующие значения, отражающие тип ОС:

* MicrosoftWindows = 0
* OSX = 1
* Linux = 2
* Solaris = 3
* FreeBSD = 4
* Android = 5
* FirefoxOS = 6

Класс OperatingSystem.Type содержит следующий метод:

* \_\_str\_\_(self) – возвращает строковое представление класса

Класс OperatingSystem содержит следующие атрибуты:

* os\_type (OperatingSystem.Type) – тип ОС
* os\_version (str) – версия ОС

Класс OperatingSystem содержит следующие методы:

* \_\_init\_\_(self, os\_type, os\_version) – конструктор класса
* \_\_str\_\_(self) – возвращает строковое представление класса

В класс PersonalComputer добавлены следующие атрибуты:

* pc\_os (OperatingSystem) – данные об ОС

Класс PersonalComputer содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, name, manufacturer, model, pc\_os) – конструктор класса
* load(cls, filename) – переопределённый конструктор (здесь и далее – см. описание класса Computer)

1. специальные методы:

* \_\_str\_\_(self) – переопределённый метод

1. прочие методы:

* save(self, filename) – переопределённый метод

1. Класс DesktopComputer

Класс DesktopComputer наследуется от класса PersonalComputer. Здесь и далее представлены данные, которые были добавлены в класс или переопределены.

В класс DesktopComputer добавлены следующие атрибуты:

* \_form\_factor (str) – форм-фактор настольного ПК

Класс DesktopComputer содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, name, manufacturer, model, pc\_os) – конструктор класса
* load(cls, filename) – переопределённый конструктор (здесь и далее – см. описание класса PersonalComputer)

1. специальные методы:

* \_\_str\_\_(self) – переопределённый метод

1. прочие методы:

* form\_factor(self) – @property – метод возвращает значение атрибута \_ form\_factor
* form\_factor(self, value) – @form\_factor.setter – метод устанавливает значение атрибута \_form\_factor
* print\_specifications(self) – переопределённый метод
* save(self, filename) – переопределённый метод

1. Класс Laptop

Класс Laptop наследуется от класса DesktopComputer. Здесь и далее представлены данные, которые были добавлены в класс или переопределены.

В класс Laptop добавлены следующие атрибуты:

* \_laptop\_type (str) – тип ноутбука

Класс Laptop содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, name, manufacturer, model, pc\_os) – конструктор класса
* load(cls, filename) – переопределённый конструктор (здесь и далее – см. описание класса DesktopComputer)

1. специальные методы:

* \_\_str\_\_(self) – переопределённый метод

1. прочие методы:

* laptop\_type(self) – @property – метод возвращает значение атрибута \_ form\_factor
* laptop\_type(self, value) – @laptop\_type.setter – метод устанавливает значение атрибута \_laptop\_type
* print\_specifications(self) – переопределённый метод
* save(self, filename) – переопределённый метод

1. Класс Tablet

Класс Tablet наследуется от класса DesktopComputer. Здесь и далее представлены данные, которые были добавлены в класс или переопределены.

Класс Laptop содержит следующие методы:

1. конструкторы: используются конструкторы родительского класса
2. специальные методы:

* \_\_str\_\_(self) – переопределённый метод

1. прочие методы:

* print\_specifications(self) – переопределённый метод

Исходный код

Исходный код приложения находится в папке «11.2» и содержится в следующих файлах:

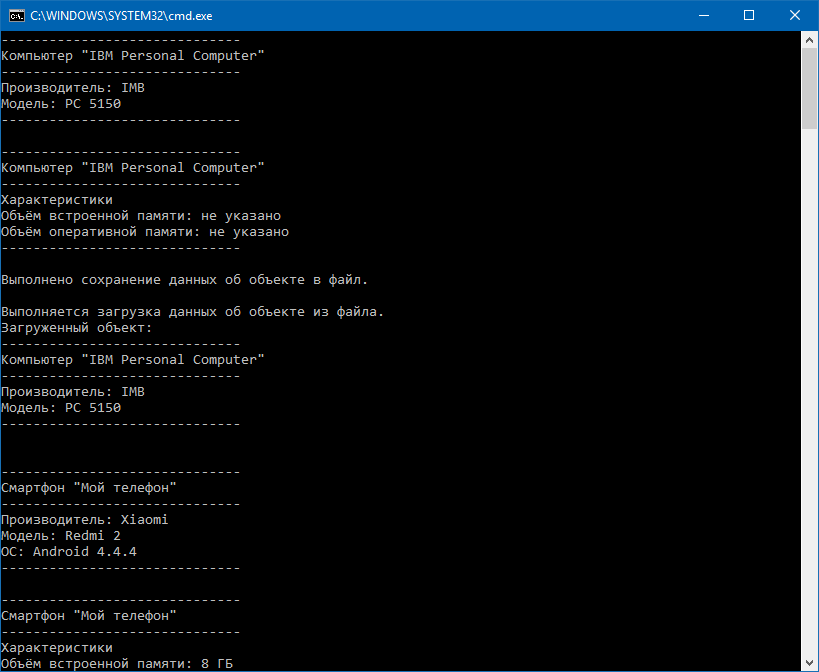
* «computer.py»: описание и реализация классов объектной иерархии
* «main.py»: демонстрация работы классов

Также добавлены файлы, содержащие структуру хранения объектов классов:

* «computer.json»: объект класса Computer
* «my\_phone.json»: объект класса Smartphone
* «pc.json»: объект класса PersonalComputer
* «desktop\_pc.json»: объект класса DesktopComputer
* «my\_job\_notebook.json»: объект класса Laptop
* «tablet\_pc.json»: объект класса Tablet

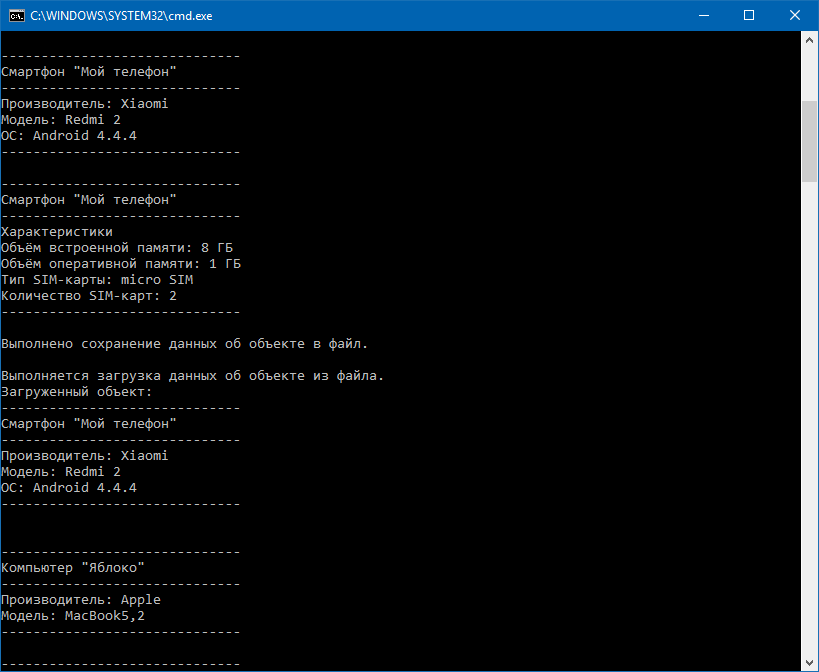
Иллюстрация работы

Иллюстрация работы с классом Computer приведена на Рисунке 2.13.



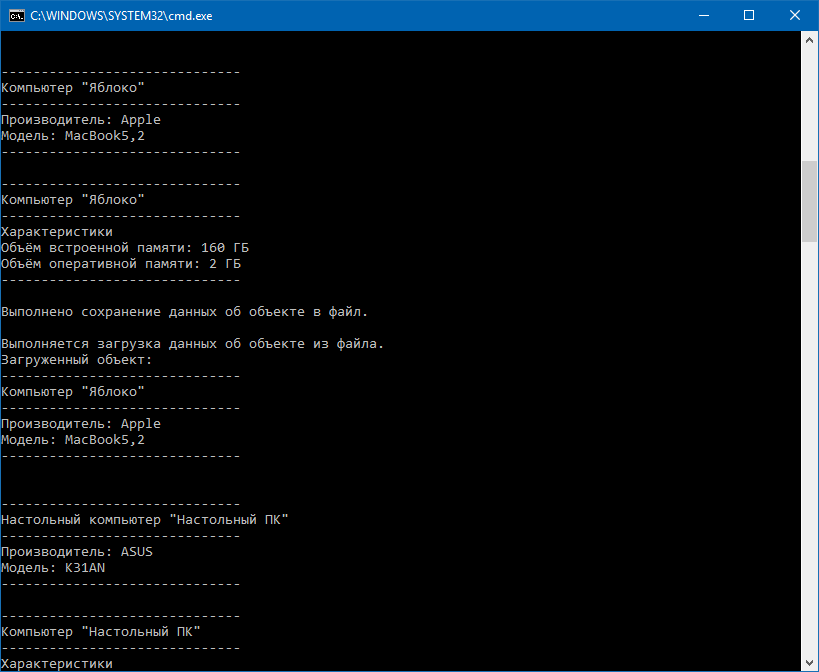
**Рисунок 2.13 —** Иллюстрация работы с классом Computer

Иллюстрация работы с классом Smartphone приведена на Рисунке 2.14.



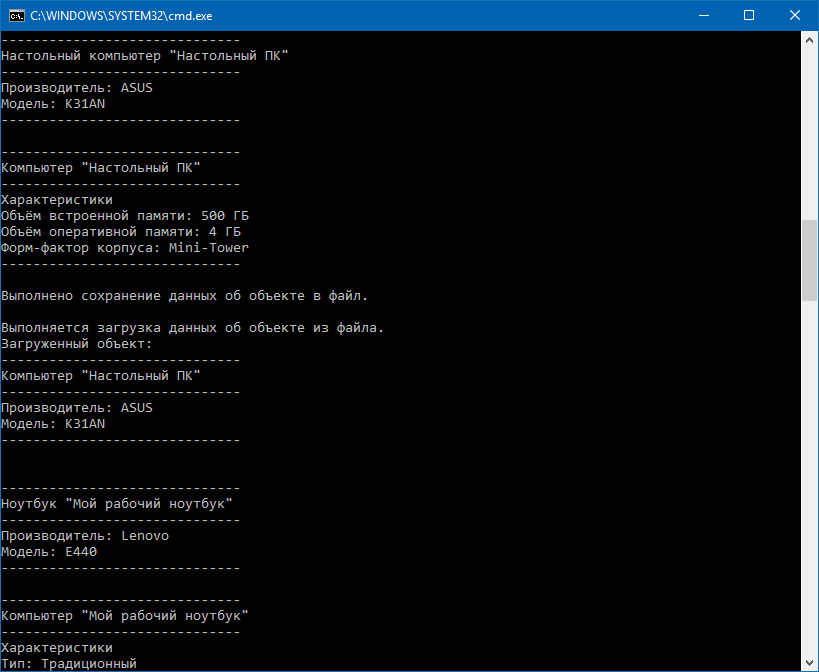
**Рисунок 2.14 —** Иллюстрация работы с классом Smartphone

Иллюстрация работы с классом PersonalComputer приведена на Рисунке 2.15.



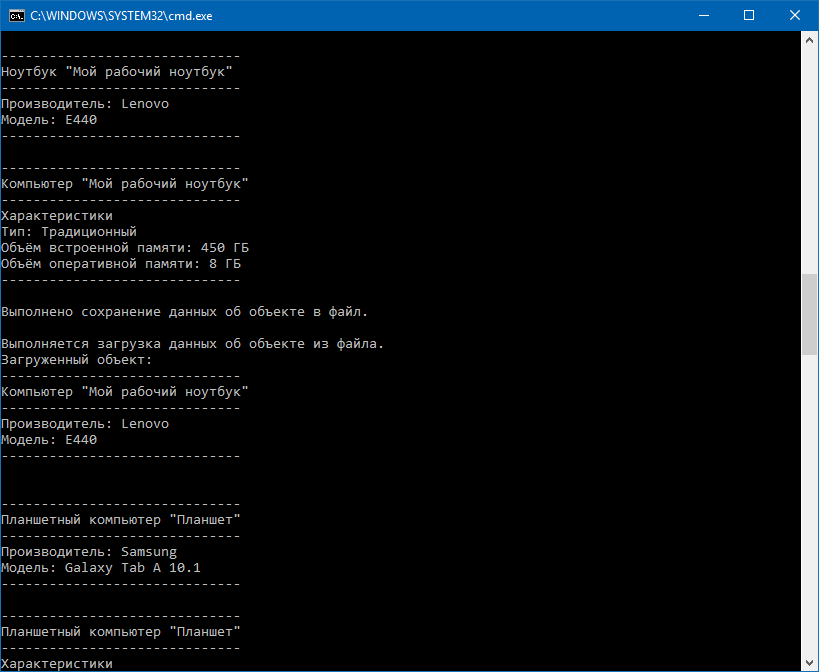
**Рисунок 2.15 —** Иллюстрация работы с классом PersonalComputer

Иллюстрация работы с классом DesktopComputer приведена на Рисунке 2.16.



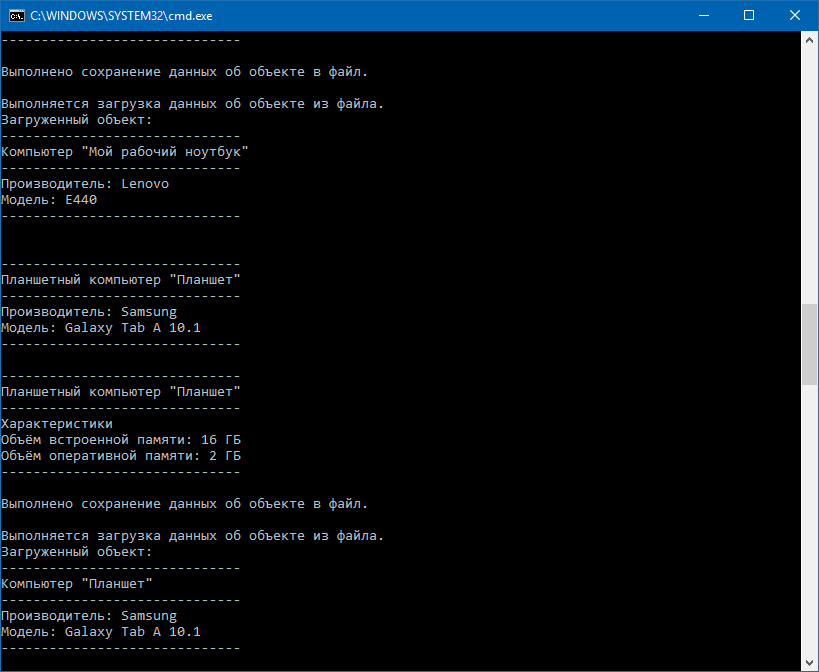
**Рисунок 2.16 —** Иллюстрация работы с классом DesktopComputer

Иллюстрация работы с классом Laptop приведена на Рисунке 2.17.



**Рисунок 2.17 —** Иллюстрация работы с классом Laptop

Иллюстрация работы с классом Tablet приведена на Рисунке 2.18.



**Рисунок 2.18 —** Иллюстрация работы с классом Tablet

2.1.3. Задача 11.3

Условие задачи

Спроектировать и реализовать класс-контейнер, содержащий набор объектов из Задачи 11.2, предусмотреть необходимые атрибуты и методы.

Для решения данной задачи были модифицированы классы из Задачи 11.2. Было добавлено два метода: get\_dict(self) и from\_dict(cls, json\_data), они позволяют получить словарь из объекта и объект из словаря соответственно. Функционал данных методов был получен из части кода функций safe() и load(), поэтому кардинальных изменений кода не произошло.

Класс CompSet содержит следующие атрибуты:

* \_\_set (set) – набор объектов

Класс Computer содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self) – конструктор класса
* load(cls, filename) – создает экземпляр класса, загружая данные из файла в формате JSON

1. методы, позволяющие изменять контейнер:

* add(self, comp) – добавление объекта
* remove(self, comp) – удаление объекта

1. специальные методы:

* \_\_str\_\_(self) – возвращает строковое представление класса
* \_\_getitem\_\_(self, k) – возвращает элемент(ы) по индексу или по срезу
* \_\_setitem\_\_(self, k, v) – присваивает элемент(ы) по индексу или по срезу
* \_\_delitem\_\_(self, k) – удаляет элемент(ы) по индексу или по срезу
* \_\_len\_\_(self) – возвращает количество элементов в наборе
* \_\_add\_\_(self, other) – выполняет конкатенацию наборов

1. прочие методы:

* save(self, filename) – метод, сохраняющий данные об объекте в файл формата JSON

Исходный код

Исходный код приложения находится в папке «11.3» и содержится в следующих файлах:

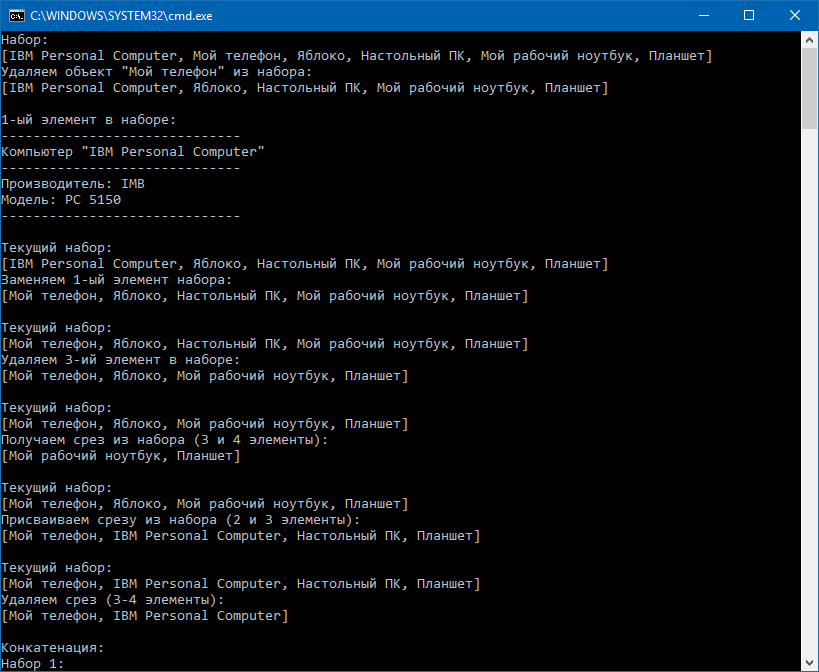
* «computer.py»: описание и реализация классов объектной иерархии
* «comp\_set.py»: описание и реализация класса CompSet
* «main.py»: демонстрация работы класса CompSet

Также добавлен файл, содержащий структуру хранения объекта класса CompSet:

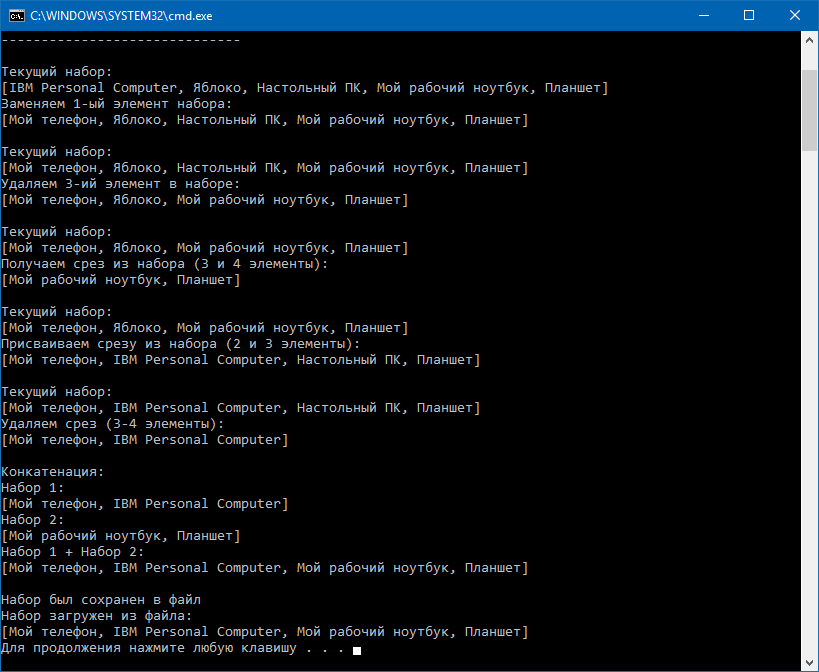
* «computer\_set.json»: объект

Иллюстрация работы

Иллюстрация работы с классом Computer приведена на Рисунках 2.18 – 2.19.



**Рисунок 2.18 —** Иллюстрация работы методов класса CompSet



**Рисунок 2.19 —** Иллюстрация работы методов класса CompSet

2.2. Стандартная библиотека

2.2.1. Задача 12.1

Условие задачи

Спроектировать приложение, позволяющее получить информацию о книге из входного html-файла и записать полученные данные в выходной файл.

Для работы приложения был спроектирован класс BookInfo.

Класс BookInfo содержит следующие атрибуты:

* source\_file (str) – имя файла-источника данных о книге
* output\_file (str) – имя выходного файла с информацией о книге
* source\_data (str) – содержимое файла-источника
* author (str) – автор книги
* title (str) – название книги
* publisher (str) – издательство книги
* year (int) – год издания
* description (str) – описание книги
* price (int) – цена книги

Класс BookInfo содержит следующие константы:

* NOT\_FOUND (str) – значение по умолчанию для атрибутов, отражающих информацию о книге
* AUTHOR\_REG (str) – регулярное выражение для поиска автора книги
* TITLE\_REG (str) – регулярное выражение для поиска названия книги
* PUBLISHER\_REG (str) – регулярное выражение для поиска издательства книги
* YEAR\_REG (str) – регулярное выражение для поиска года издания книги
* DESCRIPTION\_REG (str) – регулярное выражение для поиска описания книги
* PRICE\_REG (str) – регулярное выражение для поиска цены книги

Класс BookInfo содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self, source\_file, output\_file) – конструктор класса

1. специальные методы:

* \_\_str\_\_(self) – возвращает строковое представление класса

1. прочие методы:

* get\_source\_data(self) – получение данных из файла-источника
* get\_book\_info(self) – получение информации о книге
* get\_author(self) – получение информации об авторе
* get\_title(self) – получение информации о названии
* get\_publisher(self) – получение информации об издательстве
* get\_year(self) – получение информации о годе издания
* get\_description(self) – получение информации об описании
* get\_price(self) – получение информации о цене
* write\_book\_info(self) – запись информации о книге в файл

Исходный код

Исходный код приложения находится в папке «12.1» и содержится в следующих файлах:

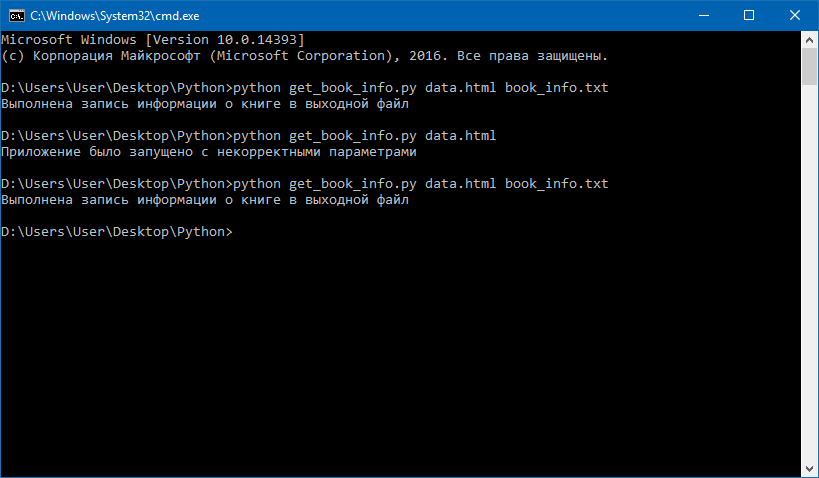
* «book\_info.py»: описание и реализация класса BookInfo
* «get\_book\_info.py»: обработка входных параметров и работа с классом BookInfo

Также добавлены входной и выходной файлы приложения:

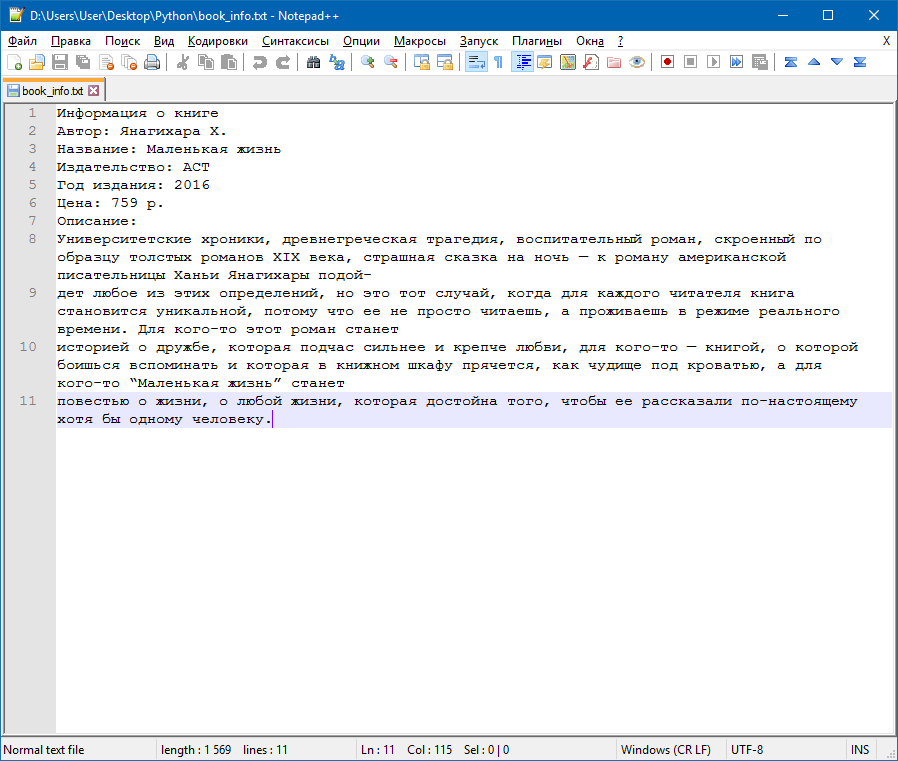
* «data.html»: html-страница с информацией о книге
* «book\_info.txt»: текстовый файл с найденной информацией о книге

Иллюстрация работы

Иллюстрация работы приложения приведена на Рисунках 2.20 – 2.21.



**Рисунок 2.20 —** Иллюстрация запуска приложения



**Рисунок 2.21 —** Иллюстрация содержимого выходного файла

2.2.2. Задача 12.2

Условие задачи

Реализовать игру «Угадай число», где в качестве игрока, угадывающего число, выступает Компьютер.

Для игры был спроектирован класс GuessNumGame.

Класс GuessNumGame содержит следующие атрибуты:

* \_nums (set) – в данном наборе учитываются возможные варианты загаданного числа
* \_tries\_num (int) – количество использованных попыток для угадывания числа

Класс GuessNumGame содержит следующие константы:

* LOG\_FILENAME (str) – имя файла для записи логов игры

Класс GuessNumGame содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self) – конструктор класса

1. прочие методы:

* \_now(self) – возвращает строку с текущими датой и временем
* \_tf(self, text) – форматирует сообщение для вывода на экран
* \_log(self, text, console=False) – выполняет запись в лог-файл, а также в случае необходимости выводит сообщение на экран
* start(self) – начало игры
* input\_range(self) – ввод интервала, в которое входит загаданное число
* get\_answer(self, question) – получение ответа на вопрос (да/нет)
* quess(self) – попытка угадать число
* finish(self) – окончание игры: вывод угаданного числа, а также количества предпринятых попыток

Исходный код

Исходный код приложения находится в папке «12.2» и содержится в следующих файлах:

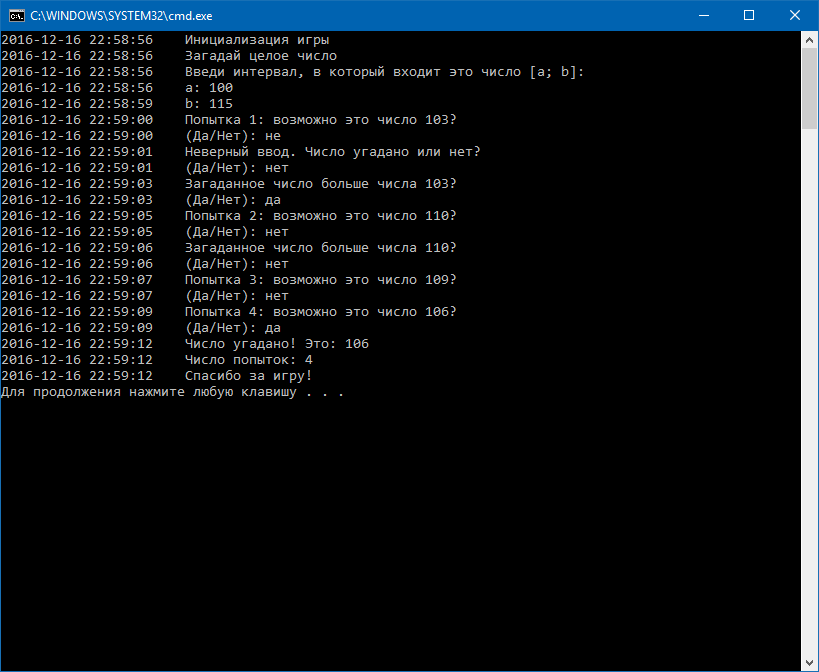
* «guessnum\_game.py»: описание и реализация класса GuessNumGame
* «main.py»: инициализация и начало игры

Также добавлен файл с записями лога игр:

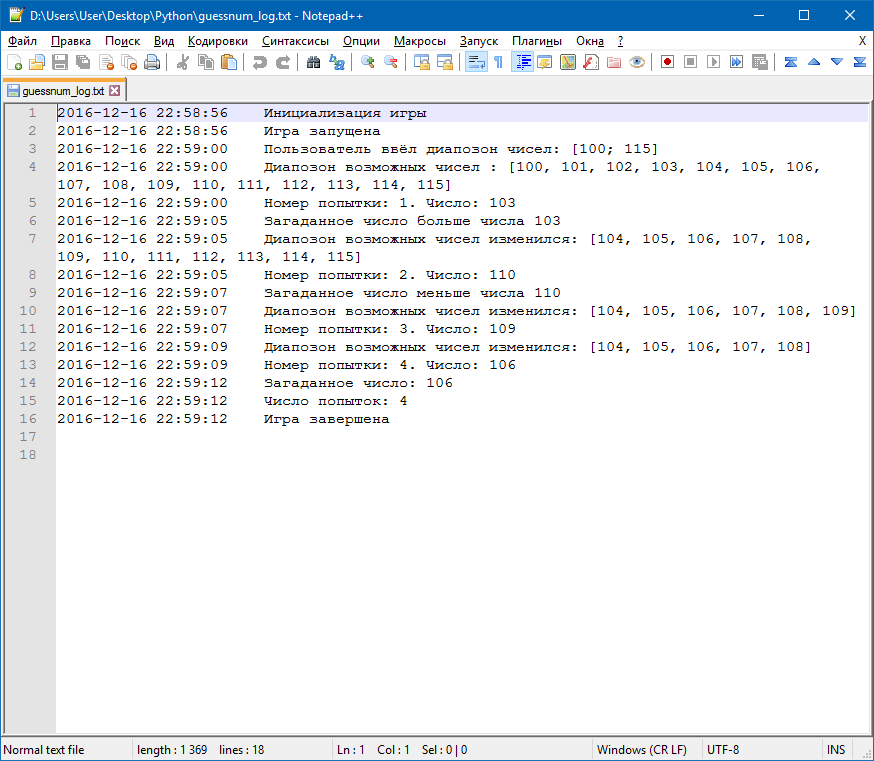
* «guessnum\_log.txt»: html-страница с информацией о книге

Иллюстрация работы

Иллюстрация игры приведена на Рисунках 2.22 – 2.23.



**Рисунок 2.22 —** Иллюстрация процесса игры



**Рисунок 2.23 —** Иллюстрация файла с логами игры

2.2.3. Задача 12.3

Условие задачи

Создать приложение, позволяющее выполнять копирование папок/файлов.

Для приложения был спроектирован класс Backup.

Класс Backup содержит следующие атрибуты:

* \_log\_filename (str) – имя файла для записи логов приложения
* \_confirm (bool) – значение атрибута определяет, нужно ли запрашивать подтверждение
* \_projects (dict) – содержит сведения о проектах, загруженные из файла настроек

Класс Backup содержит следующие константы:

* SETTINGS\_FILENAME (str) – имя файла настроек

Класс Backup содержит следующие методы:

1. конструкторы:

* \_\_init\_\_(self) – конструктор класса

1. прочие методы:

* \_now(self) – возвращает строку с текущими датой и временем
* \_log(self, text) – выполняет запись в лог-файл
* \_copy\_confirmed(self) – запрашивает подтверждение операции (если указано в настройках)
* \_check\_project\_info(self, project) – проверяет наличие всех необходимых данных о проекте
* \_explore\_files(self, path, datetime\_cond) – выполняет поиск файлов в папках-источниках, которые удовлетворяют условию «datetime» в настройках
* copy\_project (self, project) – выполняет копирование файлов указанного проекта

Исходный код

Исходный код приложения находится в папке «12.3» и содержится в следующих файлах:

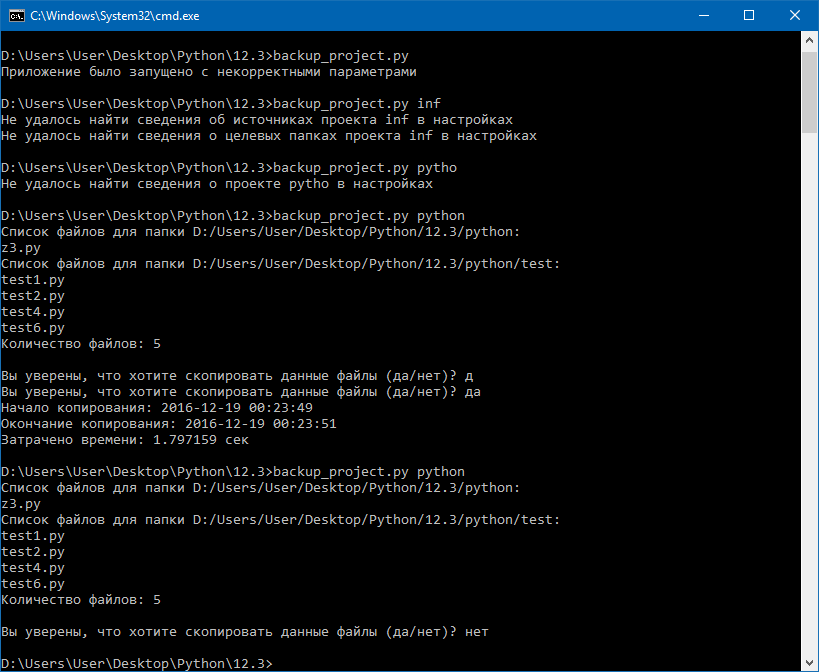
* «backup.py»: описание и реализация класса Backup
* «backup\_project.py»: инициализация и запуск приложения

Также добавлен файл с записями лога приложения:

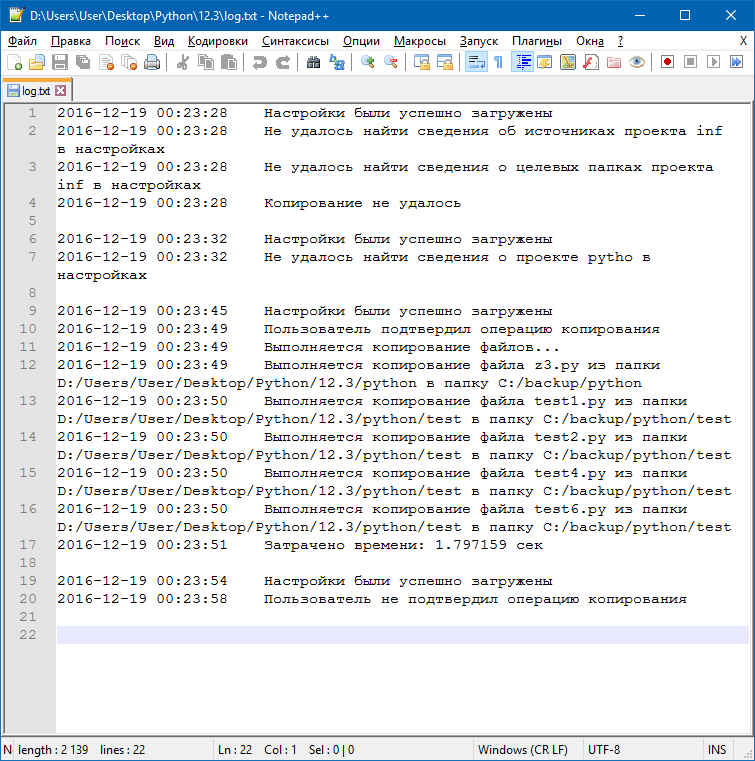
* «log.txt»: html-страница с информацией о книге

Иллюстрация работы

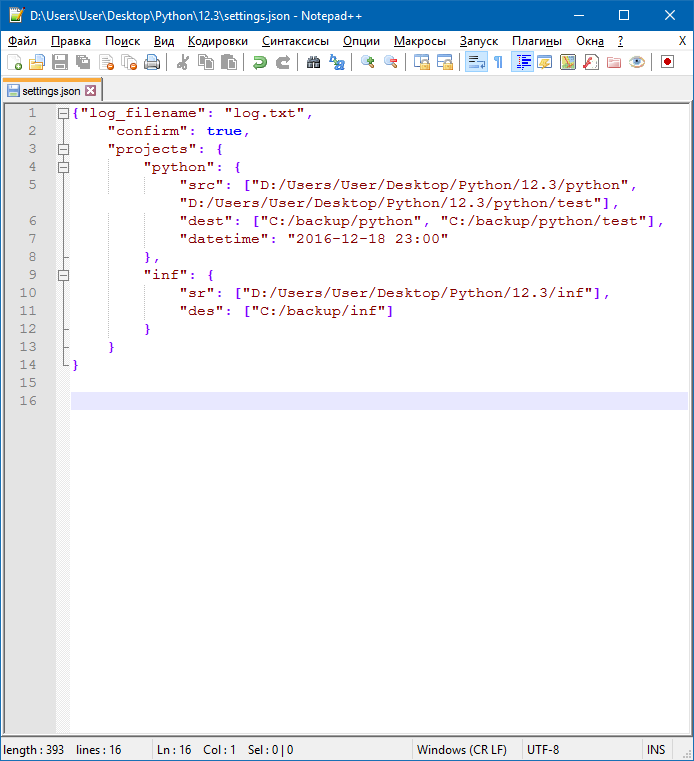
Иллюстрация работы приложения приведена на Рисунках 2.24 – 2.26.



**Рисунок 2.22 —** Иллюстрация работы приложения



**Рисунок 2.23 —** Иллюстрация файла с логами приложения



**Рисунок 2.24 —** Иллюстрация файла с настройками

1. Заключение

Во время выполнения данного курсового проекта я ознакомился с некоторыми принципами разработки программных приложений на языке Python. Я изучил различные тонкости его использования и выявил для себя следующие преимущества:

* простой и понятный синтаксис языка;
* наличие эффективных и упрощающих разработку методов и функций;
* разнообразие и доступность библиотек, расширяющих возможности разработки;
* наличие удобной и легкодоступной документации;
* развитое сообщество разработчиков, позволяющее быстро находить решение возникающих сложностей и проблем при разработке.

Список источников

Python 3.5.2 documentation. — [Электронный ресурс], (дата обращения – в течение всего времени выполнения курсового проекта) / Режим доступа: https://docs.python.org/3/index.html, свободный. — Загл. с экрана.

Computer. — [Электронный ресурс], (дата обращения 15.11.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/Computer, свободный. — Загл. с экрана.

Personal Сomputer. — [Электронный ресурс], (дата обращения 15.11.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/Personal\_computer, свободный. — Загл. с экрана.

IBM Personal Computer. — [Электронный ресурс], (дата обращения 15.11.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/IBM\_Personal\_Computer, свободный. — Загл. с экрана.

MacBook. — [Электронный ресурс], (дата обращения 15.11.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/MacBook, свободный. — Загл. с экрана.

Desktop Сomputer. — [Электронный ресурс], (дата обращения 15.11.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/Desktop\_computer, свободный. — Загл. с экрана.

Laptop. — [Электронный ресурс], (дата обращения 15.11.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/Laptop, свободный. — Загл. с экрана.

Tablet Сomputer. — [Электронный ресурс], (дата обращения 26.01.2016 г.) / Режим доступа: https://en.wikipedia.org/wiki/Tablet\_computer, свободный. — Загл. с экрана.