ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РЕСПУБЛИКИ ХАКАСИЯ

«ХАКАССКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 09.02.07

«Информационные системы и программирование»

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3

ТЕМА: СОЗДАНИЕ КЛАССОВ

Руководитель:

\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Головина

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, дата)

Выполнили:

Студент группы ИС(ПРО) – 31

\_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Кравченко

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Абакан 2024 г.

**ОТЧЕТ**

**1. Задание 1**

Решение на Python представлено в листинге 1.1.

Листинг 1.1 – Решение задания 1 на Python

class PC: # Создание класса компьютера (системного блока)

def \_\_init\_\_(self, motherboard, cpu, psu): # Конструктор класса

self.\_\_motherboard = motherboard # Инициализация поля материнской платы

self.\_\_cpu = cpu # Инициализация поля процессора

self.\_\_psu = psu # Инициализация поля блока питания

# Для материнской платы

def set\_motherboard(self, motherboard): # set

self.\_\_motherboard = motherboard # Присвоение нового значения поля

def get\_motherboard(self): # get

return self.\_\_motherboard # Возврат значения поля материнской платы

# Для процессора

def set\_cpu(self, cpu): # set

self.\_\_cpu = cpu # Присвоение нового значения поля

def get\_cpu(self): # get

return self.\_\_cpu # Возврат значения поля процессора

# Для блока питания

def set\_psu(self, psu): # set

self.\_\_psu = psu # Присвоение нового значения поля

def get\_psu(self): # get

return self.\_\_psu # Возврат значения поля блока питания

# Создание объектов

# Системный блок 1

pc1 = PC("ASUS Z790", "AMD Threadripper 9999X", "KCAS 1500W Bomb Edition")

# Системный блок 2

pc2 = PC("GIGABYTE B550", "Intel i5-10400F", "Thermaltake 600W")

# Вывод значений свойств объектов

for pc in [pc1, pc2]: # Перебор компьютеров

Продолжение листинга 1.1 – Решение задания 1 на Python

# Вывод свойств

print(f"Материнская плата: {pc.get\_motherboard()}\n" +\

f"Процессор: {pc.get\_cpu()}\n" +\

f"Блок питания: {pc.get\_psu()}")print(f"Сумма вклада = {vklad}")

Результат выполнения представлен на рисунке 1.1.

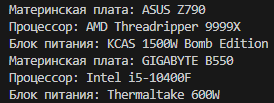


Рисунок 1.1 – Результат задание 1

**2. Задание 2**

Решение на Python представлено в листинге 2.1.

Листинг 2.1 – Решение задания 2 на python

import re # Импорт RegEx для проверки ограничений

class PC: # Создание класса компьютера (системного блока)

def \_\_init\_\_(self, motherboard, cpu, psu): # Конструктор класса

self.set\_motherboard(motherboard) # Инициализация поля материнской платы

self.set\_cpu(cpu) # Инициализация поля процессора

self.set\_psu(psu) # Инициализация поля блока питания

# Для материнской платы

def set\_motherboard(self, motherboard): # set

if len(motherboard) < 20 or len(motherboard) > 150: # Проверка ограничения

# Вывод ошибки

print("[!] Название материнской платы должно быть не менее 20 символов и не более 150 символов")

return # Досрочное завершение функции

self.\_\_motherboard = motherboard # Присвоение нового значения поля

def get\_motherboard(self): # get

return self.\_\_motherboard # Возврат значения поля материнской платы

# Для процессора

def set\_cpu(self, cpu): # set

Продолжение листинга 2.1 – Решение задания 2 на Python

if not re.fullmatch(r"[A-Za-z0-9 .;-]+", cpu): # Проверка ограничения

# Вывод ошибки

print("[!] Процессор должен содержать только буквы, цифры, тире, точки и/или точки с запятой")

return # Досрочное завершение функции

self.\_\_cpu = cpu # Присвоение нового значения поля

def get\_cpu(self): # get

return self.\_\_cpu # Возврат значения поля процессора

# Для блока питания

def set\_psu(self, psu): # set

if not re.fullmatch(r".\*\d{3,4}W.\*", psu): # Проверка ограничения

# Вывод ошибки

print("[!] Блок питания должен содержать значение напряжение в формате \"<Число>W\"")

return # Досрочное завершение функции

self.\_\_psu = psu # Присвоение нового значения поля

def get\_psu(self): # get

return self.\_\_psu # Возврат значения поля блока питания

# Создание объектов

# Системный блок 1

pc1 = PC("ASUSSSSSSSSSSSSSSSSSSS Z790", "AMD Threadripper 9999X", "KCAS 1500W Bomb Edition")

# Системный блок 2

pc2 = PC("GIGABYTEEEEEEEEEEEE B550", "Intel i5-10400F", "Thermaltake 600W")

# Вывод значений свойств объектов

for pc in [pc1, pc2]: # Перебор компьютеров

# Вывод свойств

print(f"Материнская плата: {pc.get\_motherboard()}\n" +\

f"Процессор: {pc.get\_cpu()}\n" +\

f"Блок питания: {pc.get\_psu()}")

# Проверка работы ограничений

pc1.set\_motherboard("0") # Материнская плата

pc1.set\_cpu("##@##@##@#") # Процессор

pc1.set\_psu("No w no digits :(") # Блок питания

# Замена данными, подходящими под ограничения

Продолжение листинга 2.1 – Решение задания 2 на Python

pc2.set\_motherboard("1" \* 30) # Материнская плата

pc2.set\_cpu("Qualcomm 850-1X") # Процессор

pc2.set\_psu("Omega 200W") # Блок питания

print(pc2.get\_motherboard(), pc2.get\_cpu(), pc2.get\_psu()) # Вывод свойств

Результат выполнения представлен на рисунке 2.1.

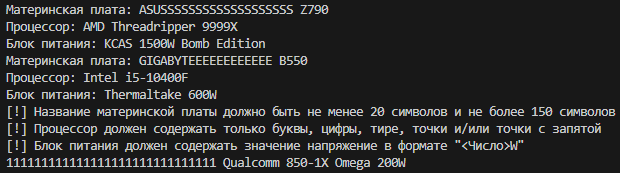


Рисунок 2.1 – Результат задание 2

**3. Заключение**

В ходе данного практического занятия были созданы классы с использованием способов реализации принципа ООП – инкапсуляции. Реализация принципа была сделана при помощи создания методов присвоения и получения значений свойств. Во втором задании также были сделаны проверки ограничений, связанных с данными свойствами.