Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА №3

«Объектно-ориентированное программирование на Python»

ДИСЦИПЛИНА: «Перспективные языки программирования»

Выполнил: студент гр. ИУК4-32Б _____ (_____Зудин Д.В._____)
(Подпись) (Ф.И.О.)

Проверил: _____ (___Пчелинцева Н.И.___)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты): 25.10.22

Результаты сдачи (защиты):
- Балльная оценка: 105
- Оценка: Отлично

Калуга, 2022 г.

Цель: формирование практических навыков объектно-ориентированного программирования, разработки и отладки программ, овладение методами и средствами разработки и оформления технической документации.

Задачи:

1. Изучить особенности создания классов;
2. Научиться создавать экземпляры классов;
3. Изучить типовые алгоритмы решения задач с использованием принципов объектно-ориентированного программирования.

Вариант №17

Формулировка задания №1

Создайте класс Student, определите метод amountOfHomework, подсчитывающий сумму необходимых к выполнению домашних заданий по трём предметам, получившееся число метод printAmountOfStudent выводит на экран; метод group, выводит на экран сообщение о том, что студент обучается в группе ИУК4-32Б. Создайте два класса-наследника Kate и Peter. Для класса Peter переопределите метод amountOfHomework, убавив один входной параметр. Введите количество заданий по предметам, необходимых для выполнения каждым студентом. Выведите для каждого студента сумму необходимых к выполнению домашних заданий и его группу.

Листинг программы для задания №1

```
class Student:
    x = 0
    y = 0
    z = 0
    group = "ИУК4-32Б"

    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def amount_of_homework(self):
        res = self.x + self.y + self.z
        return res
```

```
    def print_amount_of_student(self):
        print("Необходимо выполнить:", self.amount_of_homework(), "домашних
заданий.")

    def print_group(self):
        print("Группа:", self.group)


class Kate(Student):
    pass


class Peter(Student):
    def __init__(self, x, y):
        super().__init__(x, y, z=None)

    def amount_of_homework(self):
        res = self.x + self.y
        return res


math_k = int(input("Количество заданий по математике для Kate: "))
language_k = int(input("Количество заданий по английскому для Kate: "))
literature_k = int(input("Количество заданий по литературе для Kate: "))
kate = Kate(math_k, language_k, literature_k)

math_p = int(input("\nКоличество заданий по математике для Peter: "))
language_p = int(input("Количество заданий по английскому для Peter: "))
peter = Peter(math_p, language_p)

print("Kate")
kate.print_group()
kate.print_amount_of_student()

print("\nPeter")
peter.print_group()
peter.print_amount_of_student()
```

Результат выполнения программы для задания №1

```
Количество заданий по математике для Kate: 3
Количество заданий по английскому для Kate: 2
Количество заданий по литературе для Kate: 5
```

```
Количество заданий по математике для Peter: 7
Количество заданий по английскому для Peter: 4
Kate
Группа: ИУК4-32Б
Необходимо выполнить: 10 домашних заданий.

Peter
Группа: ИУК4-32Б
Необходимо выполнить: 11 домашних заданий.
```

Формулировка задания №2

Написать программу с использованием абстрактного класса для перевода чисел в двоичную систему счисления. Пользователь должен ввести десятичное число, а получить результат на экране в двоичном виде.

Листинг программы для задания №2

```
from abc import ABC, abstractmethod


class Convert(ABC):
    @abstractmethod
    def convert(self, value):
        pass


class Bin(Convert):
    def convert(self, value):
        res = ''
        while value > 0:
            res += str(value % 2)
            value //= 2
        return res


a = int(input("Введите число: "))
print("Число", a, "в двочной системе счисления:", Bin().convert(a))
```

Результат выполнения программы для задания №2

```
Введите число: 10
Число 10 в двоичной системе счисления: 0101
```

Формулировка задания №3

Создать класс Book и описать в нем имя автора книги и ее идентификация в библиотеке. Идентификация может быть кодом книги (8-значное число) или названием (строка). Описать метод order, определяющий, должна ли книга стоять до или после данной (по номеру или по алфавиту).

Листинг программы для задания №3

```
class Book:
    author = ''
    id = None

    def __init__(self, author, _id):
        self.author = author
        self.id = _id

    def order(self, book):
        if not isinstance(self.id, type(book.id)):
            print("Невозможно сравнить различные типы id")
        else:
            if self.id > book.id:
                print("1-ая книга должна стоять после 2-ой")
            else:
                print("1-ая книга должна стоять перед 2-ой")


b1 = Book("Пушкин", "Евгений Онегин")
b2 = Book("Толстой", "Война и мир")
b1.order(b2)
```

Результат выполнения программы для задания №3

```
1-ая книга должна стоять после 2-ой
```

Формулировка задания №4

Для описания всех людей, находящихся в магазине, необходимо выделить подмножество работников и подмножество посетителей. У каждого человека есть полное имя, у сотрудников магазина имеет значение должность, а у посетителей возраст. Необходимо сгенерировать список людей в магазине и вызовом виртуального метода из абстрактного класса напечатать для сотрудников фамилию и должность, а для посетителей имя и возраст.

Листинг программы для задания №4

```
from abc import ABC


class Person(ABC):
    name = ''
    surname = ''
    third_name = ''

    def __init__(self, name, surname, third_name):
        self.name = name
        self.surname = surname
        self.third_name = third_name

    def print_info(self):
        print("Имя:", self.name)
        print("Фамилия:", self.surname)
        print("Отчество:", self.third_name)


class Worker(Person):
    position = ''

    def __init__(self, name, surname, third_name, position):
        super().__init__(name, surname, third_name)
        self.position = position

    def print_info(self):
        print("Фамилия:", self.surname)
        print("Должность:", self.position)


class Visitor(Person):
    age = 0

    def __init__(self, name, surname, third_name, age):
        super().__init__(name, surname, third_name)
        self.age = age

    def print_info(self):
        print("Имя:", self.name)
        print("Возраст:", self.age)
```

```
persons = [Visitor("Даниил", "Зудин", "Васильевич", 19), Worker("Михаил",
"Чузов", "Юрьевич", "Кассир")]
for person in persons:
    person.print_info()
    print()
```

Результат выполнения программы для задания №4

```
Имя: Даниил
Возраст: 19

Фамилия: Чузов
Должность: Кассир
```

Выводы:

В ходе работы были сформированы практические навыки объектно-ориентированного программирования, разработки и отладки программ, овладения методами и средствами разработки и оформления технической документации.