Практическое занятие № 16

Тема: Составление программ с использованием ООП.

Цель: Закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с ООП в IDE PyCharm Community.

Постановка задачи.
Создайте класс «Матрица», который имеет атрибуты количества строк и столбцов. Добавьте методы для сложения, вычитания и умножения матриц.

Текст программы:

```
# Создайте класс «Матрица», который имеет атрибуты количества строк и столбцов.# Добавьте методы для сложения, вычитания и умножения матриц.

class Calculator:
def     init (self, num1:int, num2:int):self.num1 = num1
self.num2 = num2

def add(self):
result = self.num1 + self.num2return result

def sub(self):
result = self.num1 - self.num2return result

def mul(self):
result = self.num1 * self.num2return result

def div(self):
result = self.num1 / self.num2return result

calculate1 = Calculator(4,2)print(calculate1.add()) print(calculate1.sub()) print(calculate1.mul()) print(calculate1.div())
```

Протокол работы программы:

```
6
2
8
2.0

Process finished with exit code 0
```

Постановка задачи.
Создание базового класса "Фигура" и его наследование для создания классов "Квадрат", "Прямоугольник" и "Круг". Класс "Фигура" будет

иметь общие методы, такие как вычисление площади и периметра, а классы-наследники будут иметь специфичные методы и свойства.

Текст программы:

```
# Создание базового класса "Фигура" и его наследование для создания классов
# "Квадрат", "Прямоугольник" и "Круг". Класс "Фигура" будет иметь общие методы,
# такие как вычисление площади и периметра, а классы-наследники будут иметь
# специфичные методы и свойства.
from math import pi


class Figure:

    def __init_(self, name:str, sides:tuple):
        self.sides = sides
        self.name = name

    def area(self):
        print(f"Неизвестная фигура {self.name}, невозможно определить площадь.")

    def perimeter(self):
        try:
            per = sum(self.sides)
        except:
            per = self.side * 4
        return per


fig1 = Figure('ABC', (3,4,5))
fig1.area()
print(f"Периметр {fig1.name} равен {fig1.perimeter()}")

class Square(Figure):

    def __init_(self, name:str, side):
        self.name =  name
        self.side = side

    def area(self):
        ar = self.side**2
        return ar

sq1 = Square('ABCD', 5)
print(f"Площадь квадрата {sq1.name} равна {sq1.area()}")
print(f"Периметр квадрата {sq1.name} равен {sq1.perimeter()}")

class Rectangle(Figure):

    def area(self):
        ar = self.sides[0] * self.sides[1]
        return ar

rect1 = Rectangle('BCED', (2,3))
print(f"Площадь прямоугольника {rect1.name} равна {rect1.area()}")
print(f"Периметр прямоугольника {rect1.name} равен {rect1.perimeter()}")

class Triangle(Figure):

    def area(self):
        p = self.perimeter() / 2
        a = self.sides[0]
        b = self.sides[1]
        c = self.sides[2]
        ar = (p*(p-a)*(p-b)*(p-c))**(1/2)
        return ar
```

```
tr1 = Triangle('BCD', (3,4,5))
print(f"Площадь треугольника {tr1.name} равна {tr1.area()}")
print(f"Периметр треугольника {tr1.name} равен {tr1.perimeter()}")

class Circle(Figure):

    def init (self, name:str, radius):
        self.name = name
        self.radius = radius

    def area(self):
        ar = self.radius**2 * pi
        return ar

    def perimeter(self):
        per = self.radius * 2 * pi
        return per

circ1 = Circle('Or', 5)
print(f"Площадь круга {circ1.name} равна {circ1.area()}")
print(f"Периметр круга {circ1.name} равен {circ1.perimeter()}")
```

Протокол работы программы:

Неизвестная фигура ABC, невозможно определить площадь.
Периметр ABC равен 12
Площадь квадрата ABCD равна 25
Периметр квадрата ABCD равен 20
Площадь прямоугольника BCED равна 6
Периметр прямоугольника BCED равен 5
Площадь треугольника BCD равна 6.0
Периметр треугольника BCD равен 12
Площадь круга Or равна 78.53981633974483
Периметр круга Or равен 31.41592653589793

Process finished with exit code 0

Постановка задачи.

Для задачи из блока 1 создать две функции, save_def и load_def, которые позволяют сохранять информацию из экземпляров класса (3 шт.) в файл и загружать ее обратно. Использовать модуль pickle для сериализации и десериализации объектов Python в бинарном формате.

Текст программы:

```
# Для задачи из блока 1 создать две функции, save_def и load_def, которые позволяют
# сохранять информацию из экземпляров класса (3 шт.) в файл и загружать ее обратно.
# Использовать модуль pickle для сериализации и десериализации объектов Python в
# бинарном формате.

import pickle
print("Вывод из первого блока:")
from PZ_16_1 import Calculator
print("-"*50)


def safe_def(object, file_name):
    with open(file_name, "wb") as f1:
        pickle.dump(object, f1)

def load_def(file_name):
    with open(file_name, "rb") as f1:
        return pickle.load(f1)


calc1 = Calculator(10,8)
calc2 = Calculator(10,10)
```

```
calc3 = Calculator(4,8)

safe_def(calc1, "calc1.bin")
safe_def(calc2, "calc2.bin")
safe_def(calc3, "calc3.bin")

calc1_loaded = load_def("calc1.bin")
calc2_loaded = load_def("calc2.bin")
calc3_loaded = load_def("calc3.bin")

print(calc1_loaded.add())
print(calc2_loaded.add())
print(calc3_loaded.add())
```

Протокол работы программы:

Вывод из первого блока:

6
2
8
2.0

18
20
12

Process finished with exit code 0

Вывод: в процессе выполнения практического занятия я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ и приобрёл навыки составление программ с ООП в IDE PyCharm Community.