**Лабораторная работа №7**

**Разработка приложения с использованием ООП.**

**Цель работы**: ознакомиться с методологией объектноориентированного программирования, изучить реализацию данной методологии в языке Python 3.

**Задание:** написать класс Publisher и несколько различных классов с постфиксом Subscriber. Реализовать между данными классами отношения по паттерну Observer.

**Выполнение задания:** Алгоритм кода реализует паттерн "Наблюдатель" (Observer). Он описывает взаимодействие между издателем (Publisher) и подписчиками (Subscriber). Создаются объекты Publisher и два объекта Subscriber с разными именами. Метод update каждого подписчика выводит на консоль сообщение, содержащее имя подписчика и полученное событие. Алгоритм демонстрирует механизм, где издатель уведомляет всех своих подписчиков о событиях, а подписчики реагируют на эти события независимо друг от друга. Это гибкий подход, позволяющий добавлять и удалять подписчиков без изменения кода издателя.

Листинг программы:

*class Publisher:  
 def \_\_init\_\_(self):  
 self.subscribers = []  
  
   
 def register(self, subscriber):  
 if subscriber not in self.subscribers:  
 self.subscribers.append(subscriber)  
  
 def unregister(self, subscriber):  
 try:  
 self.subscribers.remove(subscriber)  
 except ValueError:  
 pass  
  
 def notify\_subscribers(self, event):  
 for subscriber in self.subscribers:  
 subscriber.update(event)  
  
class Subscriber:  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
 def update(self, event):  
 print(f'{self.name}: Получил событие "{event}"')  
  
sub1 = Subscriber("Подписчик 1")  
sub2 = Subscriber("Подписчик 2")  
  
publisher = Publisher()  
  
publisher.register(sub1)  
publisher.register(sub2)  
  
publisher.notify\_subscribers("Событие произошло!")*

Результаты выполнения кода:

