**Лабораторная работа № 7**

РАЗРАБОТКА ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ООП

**Вариант 10.**

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

**Краткая теория**

Объектно-ориентированное программирование (ООП) – это одна из методологий (парадигм) программирования (структурное, функциональное), где компоненты программы представляются в виде объектов. Объект – это экземпляр класса, имеющего поля и методы.

Большинство современных языков программирования, таких как Java, C++, C# и другие, реализуют данный подход, в том числе и Python. В данной лабораторной работе не рассматривается теория ООП, а только реализация данной методологии в Python 3.

**Реализация ООП в Python 3**

Для создания класса требуется указать ключевое слово class указать имя создаваемого класса и поставить знак двоеточие «:». Для указания полей и методов, нужно в теле класса создать переменные и методы соответственно.

Конструктор класса – это метод, который вызывается в начале создания объекта. Метод \_\_init\_\_(). Создание объекта класса происходит путем указания имени класса с круглыми скобками «()». Следует отметить, что для создания полей объекта не обязательного объявлять их в теле класса, достаточно объявить их в конструкторе.

**Модификаторы доступа. Инкапсуляция**

В Python существуют три вида модификаторов доступа: public, protected, private. Модификатор доступа public разрешает доступ к переменным из любой точки вне и внутри класса. Модификатор доступа private допускает обращения к переменным только внутри класса. Модификатор доступа protected разрешает доступ из любой точки вне и внутри класса, внутри пакета, а также в классах-наследниках.

Инкапсуляция – это сокрытие данных одного класса от прямого обращения к его данным другими классами. Суть заключается в том, чтобы предоставлять доступ к переменным класса через методы геттеры и сеттеры, в которых может происходить логика, связанная с работой защищаемой переменной.

Для реализации модификатора доступа protected требуется перед наименованием переменной поставить один знак нижнего подчеркивания self.\_nameAnimal, а для модификатора доступа private – два знака нижнего подчеркивания self.\_\_ageAnimal.

Для объявления переменной workers используется конструктор, а сама переменная объявляется с модификатором доступа private. Далее создается метод для получения значений из данной переменной, однако, так как эта переменная хранит в себе список, с помощью метода get\_workers передается копия данного списка. Таким образом, классы-пользователи данной переменной не смогут её изменить. Такое состояние называется иммутабельностью. В данном случае при прямом обращении из точки кода вне класса Factory к переменной workers возникнет исключение AttributeError. Однако данное исключение можно обойти и получить доступ к переменной с модификатором доступа private с помощью рефлексии.

**Наследование**

Наследование – это предоставление характеристик классародителя классу-потомку. Данный механизм позволяет в несколько раз сократить количество кода за счет его повторного использования. Стоит также отметить, что в Python существует поддержка множественного наследования.

Для реализации наследования в Python 3 необходимо при объявлении класса в круглых скобках указать классыродителей.

**Полиморфизм**

Полиморфизм заключается в способности объекта вести себя по-разному. Под полиморфизмом рассмотрим перегрузку и переопределение методов.

Перегрузка метода – это способность метода вести себя поразному при определенном наборе параметров. Переопределение метода заключается в изменении логики метода с таким же наименованием и количеством параметров, что и метод в классе-родителе.

**Задание.** **Вариант 10.**

Написать класс ChartDrawer, который принимает список списков, содержащих координаты точек и содержит методы add\_point(x, y) для добавления новой точки в список, remove\_point(x, y) для удаления точек с указанными координатами. Класс должен правильно использовать инкапсуляцию

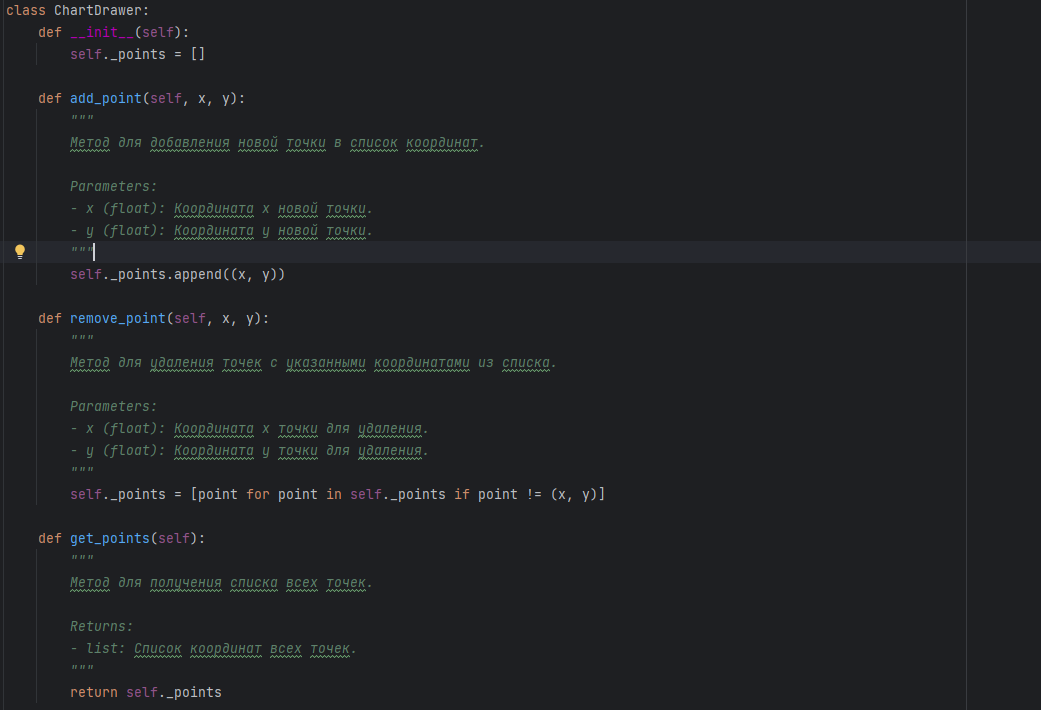


Рисунок 7.1 – код программы.

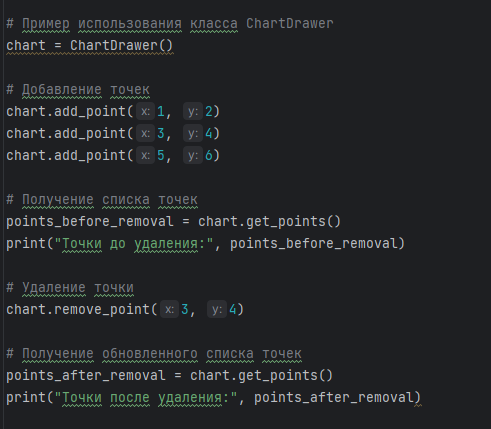


Рисунок 7.2 – код программы.

Результат работы программы представлен на рисунке 7.3

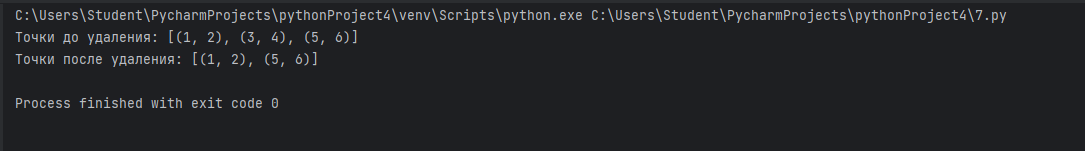


Рисунок 7.3 – результат работы программы

**Вывод**: в результате проделанной работы ознакомились с методологией объектноориентированного программирования, изучили реализацию данной методологии в языке Python 3

|  |  |
| --- | --- |
| Выполнил | Рогов М.А. |
| Проверил | Елкин Н.С. |