**Практическая работа №18: ООП. Полиморфизм**

Выполните следующие задания:

***Задания №1***

В качестве практической работы попробуйте самостоятельно перегрузить оператор сложения. Для его перегрузки используется метод \_\_add\_\_(). Он вызывается, когда объекты класса, имеющего данный метод, фигурируют в операции сложения, причем с левой стороны. Это значит, что в выражении a + b у объекта a должен быть метод \_\_add\_\_(). Объект b может быть чем угодно, но чаще всего он бывает объектом того же класса. Объект b будет автоматически передаваться в метод \_\_add\_\_() в качестве второго аргумента (первый – self).

Отметим, в Python также есть правосторонний метод перегрузки сложения - \_\_radd\_\_().

Согласно полиморфизму ООП, возвращать метод \_\_add\_\_() может что угодно. Может вообще ничего не возвращать, а "молча" вносить изменения в какие-то уже существующие объекты. Допустим, в вашей программе метод перегрузки сложения будет возвращать новый объект того же класса.

***Задания №2***

Пересмотрите алгоритм решения прошлой практической работы, с использованием инкапсуляции. Реализуйте старый алгоритм с использованием полиморфизма.

Написать программу, в которой есть главный класс Games со статическим полем Year, опишите конструктор присваивающий значение полю Year, также опишите метод getName, который возвращает имя игры. На основе главного класса путем наследования опишите четыре класса PCGames, PS4Games, XboxGames, MobileGames. Добавьте каждому классу дополнительные поля и переопределите у всех классов метод getName.

***Задания №3***

Пересмотрите алгоритм решения прошлой практической работы, с использованием инкапсуляции. Реализуйте старый алгоритм с использованием полиморфизма.

Напишите программу с пустым классом Country. Опишите наследуемые от класса Country классы Russia, Canada, Germany. Добавьте каждому классу поле population и опишите метод setPopulation и getPopulation.