Объектно-ориентированное программирование

Это что еще за покемон?

Что это такое?

«ООП» или «Объектно-Ориентированное Программирование» – это парадигма программирования, т.е. один из подходов к написанию программ, который основывается на классах и объектах.

В чем отличия от структурного программирования?

Объектно-ориентированное программирование — это расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов.

Эта модель ставит в центр внимания объекты, а не действия, данные, а не логику.

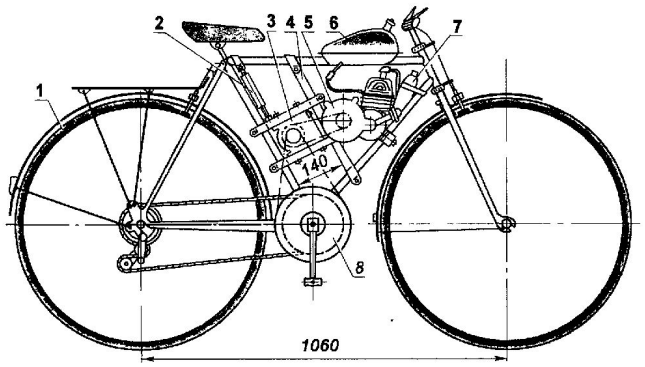
В чем плюсы такого подхода?

ООП сильно упрощает процесс организации и создания структуры программы. Отдельные объекты, которые можно менять без воздействия на остальные части программы, упрощают также и внесение в программу изменений.

Так как с течением времени программы становятся всё более крупными, а их поддержка всё более тяжёлой, эти два аспекта ООП становятся всё более актуальными.

Проведем аналогию из жизни

Чертеж



Класс

Готовая модель

Объект класса

Таким образом…

Классы — это некоторые описания, схемы, чертежи по которым создаются объекты.

Для создания объекта в ООП необходимо сначала составить чертежи, то есть классы.

Объект — это сущность, экземпляр класса, у которой есть свои свойства и которой можно посылать сообщения, чтобы она на них реагировала.

Еще немного аналогий…

Это объекты одного класса, их свойства и действия(методы)

Цвет = серебряный;
Диаметр колес = 26;
Размер рамы = 18;
Текущая скорость = 0;
Сломаться();
Заесть цепь();
Опустить седло();
Поднять седло();

Цвет = розовый;
Диаметр колес = 22;
Размер рамы = 14;
Текущая скорость = 0;
Сломаться();
Заесть цепь();
Опустить седло();
Поднять седло();

Цвет = голубой;
Диаметр колес = 12;
Размер рамы = 7;
Текущая скорость = 0;
Сломаться();
Заесть цепь();
Опустить седло();
Поднять седло();

Подведем итоги.

Классы — это абстракция, описывающая методы, свойства ещё не существующих объектов.

Объекты — конкретное представление этой абстракции, имеющее свои свойства. Созданные объекты на основе одного класса называются экземплярами этого класса.

Эти объекты могут иметь различное поведение, свойства, но все равно будут являться объектами одного класса и выполнять описанные в нем действия.

Двигаемся дальше: концепции ООП

1. **Абстракция данных**: подробности внутренней логики скрыты от конечного пользователя. Пользователю не нужно знать, как работает те или иные классы и методы, чтоб их использовать.

На примере того же велосипеда — когда мы ездим на нём или меняем деталь, нам не нужно знать, как педаль приводит его в движение или как закреплена цепь.

Двигаемся дальше: концепции ООП

2. Наследование: самый популярный принцип ООП.

Оно делает возможным повторное использование кода — если какой-то класс уже имеет какую-то логику и функции, нам не нужно переписывать всё это заново для создания нового класса, мы можем просто включить старый класс в новый, целиком.

Например: мы сначала создали обычный велосипед, а потом захотели создать скоростной. Просто наследовали основное от обычного, добавили скорости и ручные тормоза.

Двигаемся дальше: концепции ООП

3. Инкапсуляция: включение в класс объектов другого класса, вопросы доступа к ним, их видимости.

Это механизм, который объединяет данные и код, манипулирующий этими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования.

Двигаемся дальше: концепции ООП

4. Полиморфизм: «поли» значит «много», а «морфизм» — «изменение» или «вариативность», таким образом, «полиморфизм» — это свойство одних и тех же объектов и методов принимать разные формы.

В более общем смысле, концепцией полиморфизма является идея "один интерфейс, множество методов". Это означает, что можно создать общий интерфейс для группы близких по смыслу действий. (перегрузка)

Двигаемся дальше: концепции ООП

5. Обмен сообщениями: способность одних объектов вызывать методы других объектов, передавая им управление.

Немаловажный аспект

Самое важное в работе с объектно-ориентированным программированием – научиться мыслить объектно!

В структурном программировании	**В объектно-ориентированном**
Переменные – это контейнеры каких-то значений	Есть классы, описывающие свойства, действия (методы) и общие черты создаваемых объектов.
Функции – это список действий, которые нужно выполнить	
Мы можем передать в функцию переменную, чтобы она с ней что-нибудь сделала	Есть объекты, реализующие все то, что описано в этих классах, которые могут исполнять описанные в классе действия и содержать описанные свойства.