Занятие 37. Лекция.

Конспект лекции присылать на почту tankae@inbox.ru до 21:00

Тема: Понятие об объектно-ориентированном программировании.

План лекции:

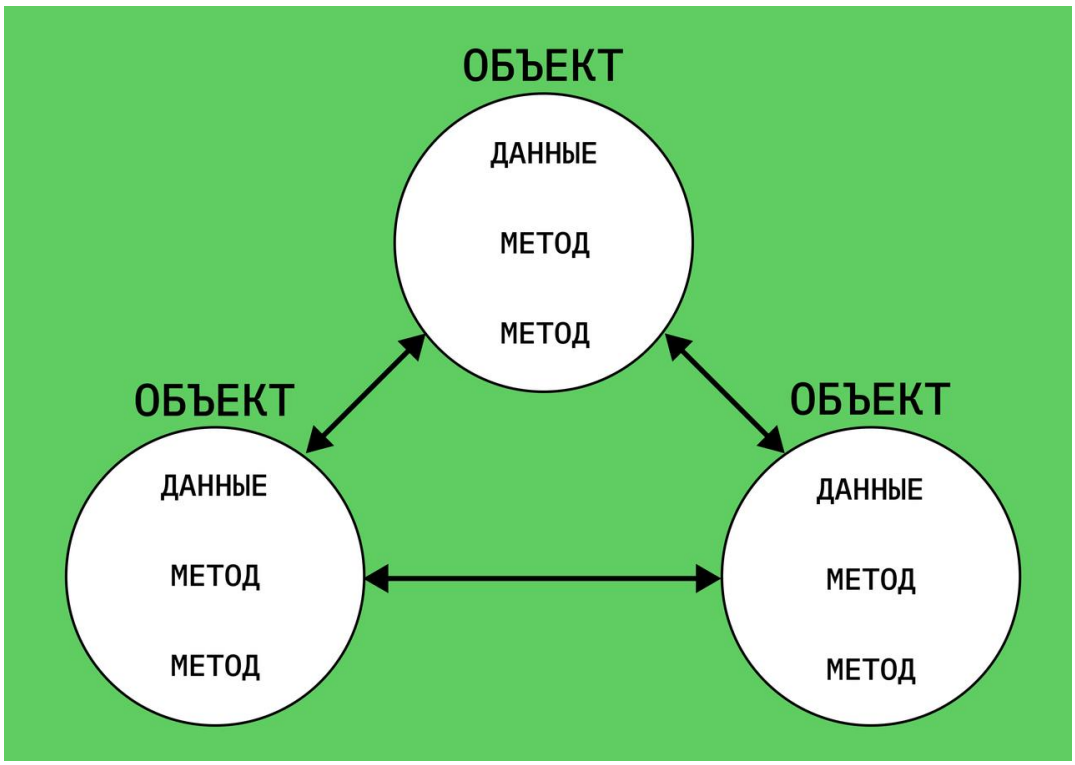
1. Что такое ООП?
2. Структура ООП
3. Основные принципы ООП

Объектно-ориентированное программирование

Объектно-ориентированное программирование, или ООП — это одна из парадигм разработки. Парадигмой называют набор правил и критериев, которые соблюдают разработчики при написании кода. Если представить, что код — это рецепт блюда, то парадигма — то, как рецепт оформлен в кулинарной книге. Парадигма помогает стандартизировать написание кода. Это снижает риск ошибок, ускоряет разработку и делает код более читабельным для других программистов.

Суть понятия объектно-ориентированного программирования в том, что все программы, написанные с применением этой парадигмы, состоят из объектов. Каждый объект — это определённая сущность со своими данными и набором доступных действий.

Например, нужно написать для интернет-магазина каталог товаров. Руководствуясь принципами ООП, в первую очередь нужно создать объекты: карточки товаров. Потом заполнить эти карточки данными: названием товара, свойствами, ценой. И потом прописать доступные действия для объектов: обновление, изменение, взаимодействие.



Структура объектно-ориентированного программирования

В коде, написанном по парадигме ООП, выделяют четыре основных элемента:

1. Объект.

Часть кода, которая описывает элемент с конкретными характеристиками и функциями. Карточка товара в каталоге интернет-магазина — это объект. Кнопка «заказать» — тоже.

2. Класс.

Шаблон, на базе которого можно построить объект в программировании. Например, у интернет-магазина может быть класс «Карточка товара», который описывает общую структуру всех карточек. И уже из него создаются конкретные карточки — объекты.

Классы могут наследоваться друг от друга. Например, есть общий класс «Карточка товара» и вложенные классы, или подклассы: «Карточка бытовой техники», «Карточка ноутбука», «Карточка смартфона». Подкласс берёт свойства из родительского класса, например, цену товара, количество штук на

складе или производителя. При этом имеет свои свойства, например, диагональ дисплея для «Карточки ноутбука» или количество сим-карт для «Карточки смартфона».

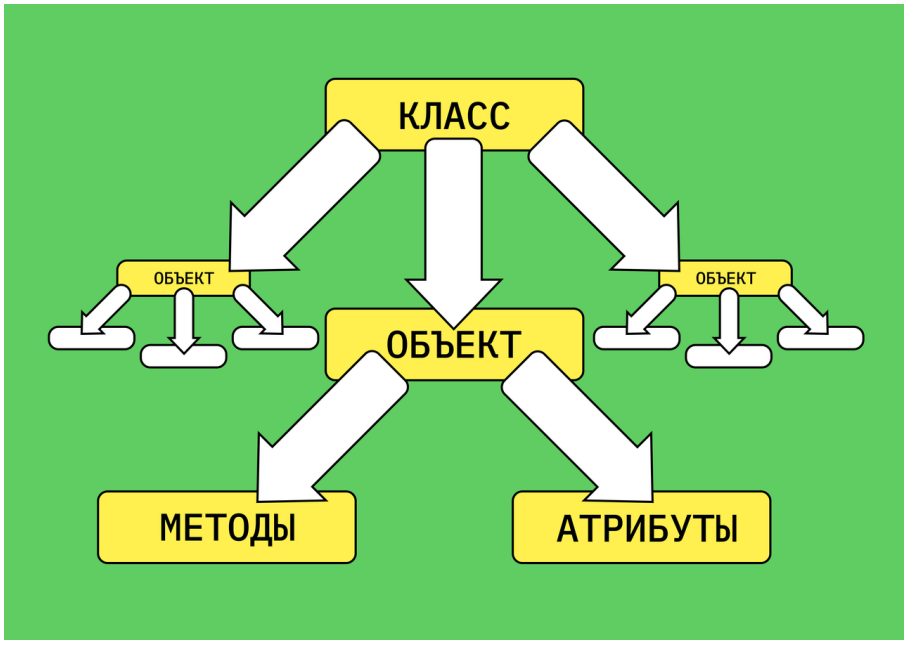
3. Метод.

Функция внутри объекта или класса, которая позволяет взаимодействовать с ним или другой частью кода. В примере с карточками товара метод может:

- Заполнить карточку конкретного объекта нужной информацией.
- Обновлять количество товара в наличии, сверяясь с БД.
- Сравнивать два товара между собой.
- Предлагать купить похожие товары.

4. Атрибут.

Характеристики объекта в программировании — например, цена, производитель или объём оперативной памяти. В классе прописывают, что такие атрибуты есть, а в объектах с помощью методов заполняют эти атрибуты данными.



Основные принципы объектно-ориентированного программирования

Объектно-ориентированное программирование базируется на трёх основных принципах, которые обеспечивают удобство использования этой парадигмы.

- **Инкапсуляция**

Вся информация, которая нужна для работы конкретного объекта, должна храниться внутри этого объекта. Если нужно вносить изменения, методы для этого тоже должны лежать в самом объекте — посторонние объекты и классы этого делать не могут. Для внешних объектов доступны только публичные атрибуты и методы.

Например, метод для внесения данных в карточку товара должен обязательно быть прописан в классе «Карточка товара». А не в классе «Корзина» или «Каталог товаров».

Такой принцип обеспечивает безопасность и не даёт повредить данные внутри какого-то класса со стороны. Ещё он помогает избежать случайных зависимостей, когда из-за изменения одного объекта что-то ломается в другом.

- **Наследование**

В этом принципе — вся суть объектно-ориентированного программирования.

Разработчик создаёт:

- ✓ Класс с определёнными свойствами;
- ✓ Подкласс на его основе, который берёт свойства класса и добавляет свои;
- ✓ Объект подкласса, который также копирует его свойства и добавляет свои.

Каждый дочерний элемент наследует методы и атрибуты, прописанные в родительском. Он может использовать их все, отбросить часть или добавить новые. При этом заново прописывать эти атрибуты и методы не нужно.

Например, в каталоге товаров:

1. У класса «Карточка товара» есть атрибуты тип товара, название, цена, производитель, а также методы «Вывести карточку» и «Обновить цену».

2. Подкласс «Смартфон» берёт все атрибуты и методы, записывает в атрибут «тип товара» слово «смартфон плюс добавляет свои атрибуты — «Количество сим-карт» и «Ёмкость аккумулятора».

3. Объект «Смартфон Xiaomi 11» заполняет все атрибуты своими значениями и может использовать методы класса «Карточка товара».

Наследование хорошо видно в примере кода выше, когда сначала создавали класс, потом подкласс, а затем объект с общими свойствами.

- **Полиморфизм**

Один и тот же метод может работать по-разному в зависимости от объекта, где он вызван, и данных, которые ему передали. Например, метод «Удалить» при вызове в корзине удалит товар только из корзины, а при вызове в карточке товара — удалит саму карточку из каталога.

То же самое с объектами. Можно использовать их публичные методы и атрибуты в других функциях и быть уверенным, что всё сработает нормально.

Этот принцип ООП, как и другие, обеспечивает отсутствие ошибок при использовании объектов.