Хурамова Ф.У
Джизакский политехнический институт
г. Джизак, Узбекистан

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Аннотация

Объектно-ориентированное программирование (ООП) – это парадигма программирования, основанная на концепции классов и объектов. Он используется для структурирования программы в простые многократно используемые части схем кода (обычно называемые классами), которые используются для создания отдельных экземпляров объектов.

Ключевые слова

Объектно-ориентированное программирование продукт, технология, JavaScript, C ++, Java и Python, программное обеспечения.

Khuramova F.U
Jizzakh Polytechnic Institute
Jizzak, Uzbekistan

OBJECT-ORIENTED PROGRAMMING

Annotation

Object Oriented Programming (OOP) is a programming paradigm based on the concept of classes and objects. It is used to structure a program into simple, reusable pieces of code schema (commonly called classes) that are used to create separate instances of objects.

Keywords

Object-oriented programming product, technology, JavaScript, C ++, Java and Python, software.

Существует множество объектно-ориентированных языков программирования, включая JavaScript, C ++, Java и Python. На сегодняшний день ООП – наиболее распространённый метод разработки современного программного обеспечения. Но использование этого метода предполагает понимание ряда принципов. В ООП есть 4 принципа:

1) Наследование – дочерние классы наследуют данные и поведение от родительского класса.

2) Инкапсуляция – содержит информацию в объекте, показывая только выбранную информацию.

3) Абстракция – раскрытие только общедоступных методов высокого уровня для доступа к объекту.

4) Полиморфизм – многие методы могут выполнять одну и ту же задачу. Главными преимуществами использования ООП являются:

1) ООП моделирует сложные вещи как воспроизводимые простые структуры.

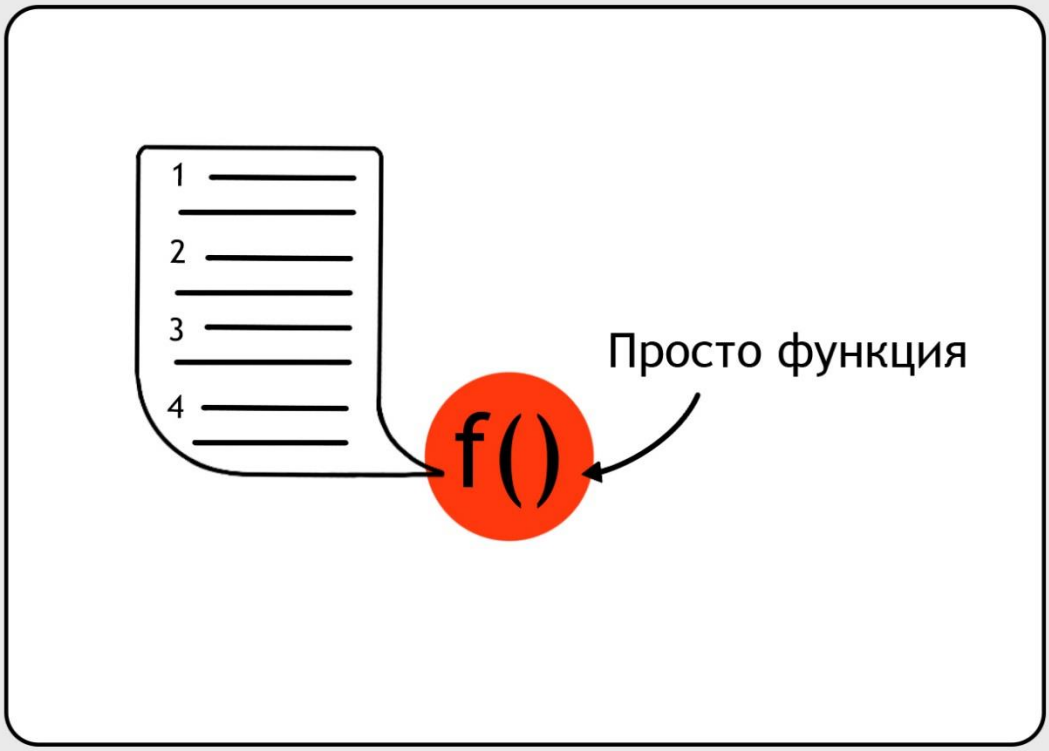
2) Многоразовые объекты ООП могут использоваться в программах.

3) Допускает поведение, зависящее от класса, за счет полиморфизма.

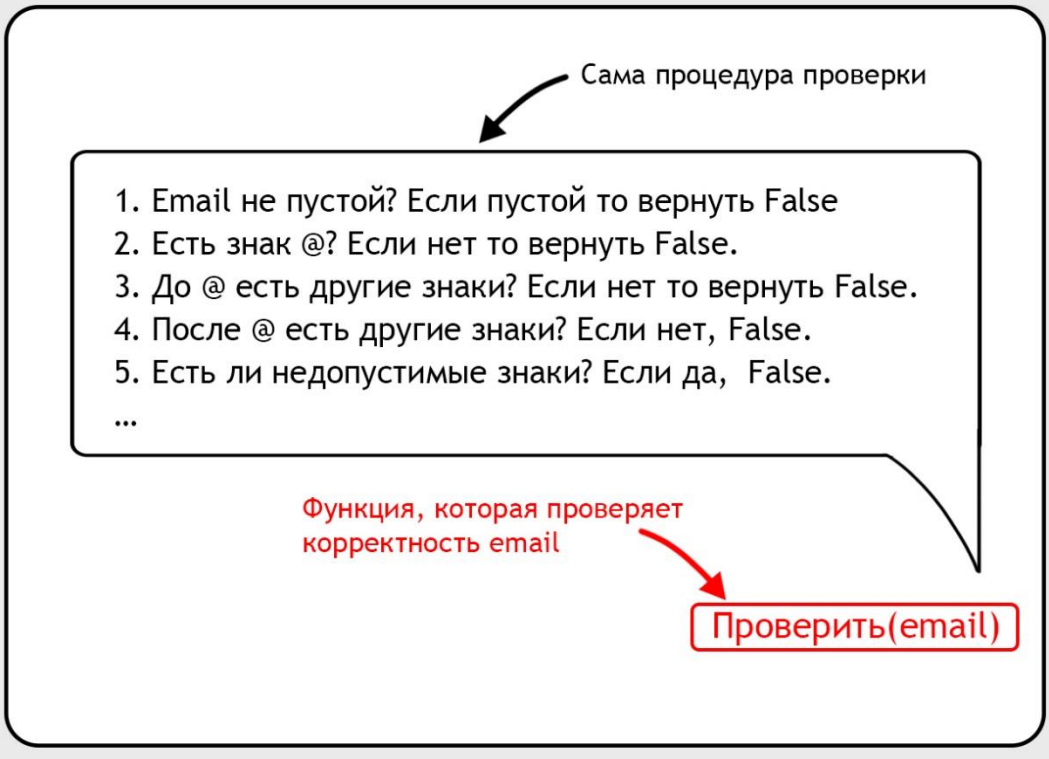
4) Легче отлаживать, классы часто содержат всю применимую к ним информацию.

5) Надежно, защищает информацию за счет инкапсуляции. Однако есть и недостатки, среди которых можно отметить: освоение базовых концепций ООП не требует значительных усилий. Однако разработка библиотек классов и их использование требуют существенных трудозатрат; документирование классов – задача более трудная, чем это было в случае процедур и модулей; в сложных иерархиях классов поля и методы обычно наследуются с разных уровней. Не всегда легко определить, какие поля и методы фактически относятся к данному классу.

Чаще всего под обычным понимают процедурное программирование, в основе которого — процедуры и функции. Функция — это мини-программа, которая получает на вход какие-то данные, что-то делает внутри себя и может отдавать какие-то данные в результате вычислений. Представим, что это такой конвейер, который упакован в коробочку.



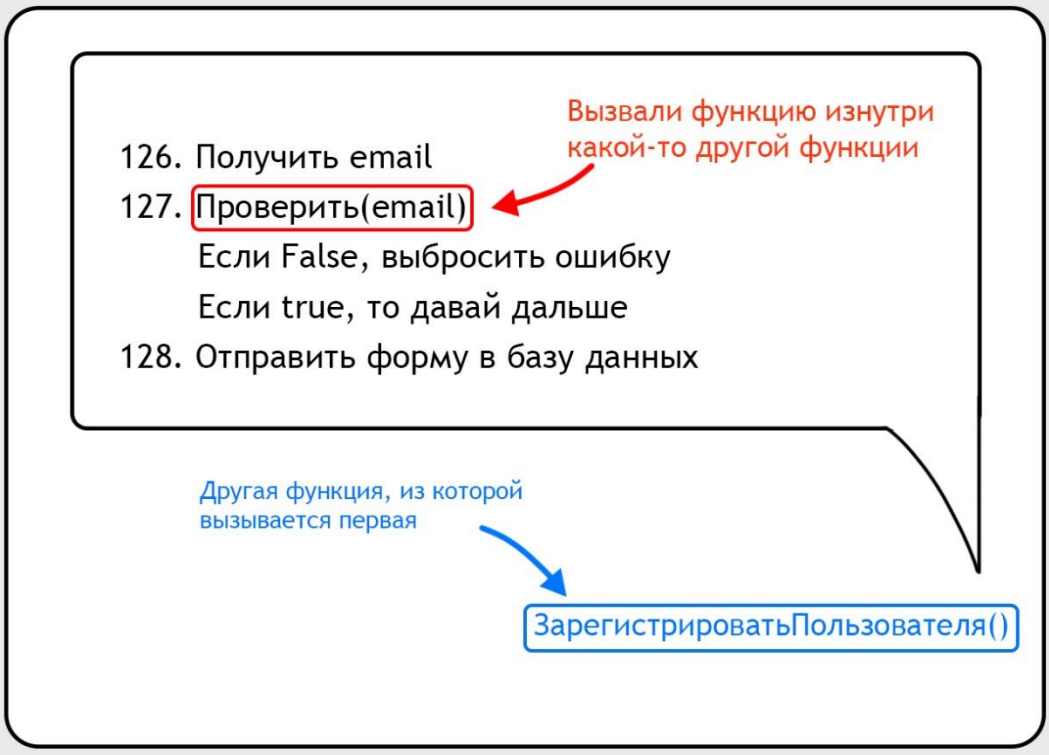
Например, в интернет-магазине может быть функция «Проверить email». Она получает на вход какой-то текст, сопоставляет со своими правилами и выдаёт ответ: это правильный электронный адрес или нет. Если правильный, то true, если нет — то false.



Функции полезны, когда нужно упаковать много команд в одну. Например, проверка электронного адреса может состоять из одной проверки на регулярные выражения, а может содержать множество команд: запросы в словари, проверку по базам спамеров и даже сопоставление с уже известными электронными адресами. В функцию можно упаковать любой комбайн из действий и потом просто вызывать их все одним движением.

Процедурное программирование идеально работает в простых программах, где все задачи можно решить, грубо говоря, десятком функций. Функции аккуратно вложены друг в друга, взаимодействуют друг с другом, можно передать данные из одной функции в другую.

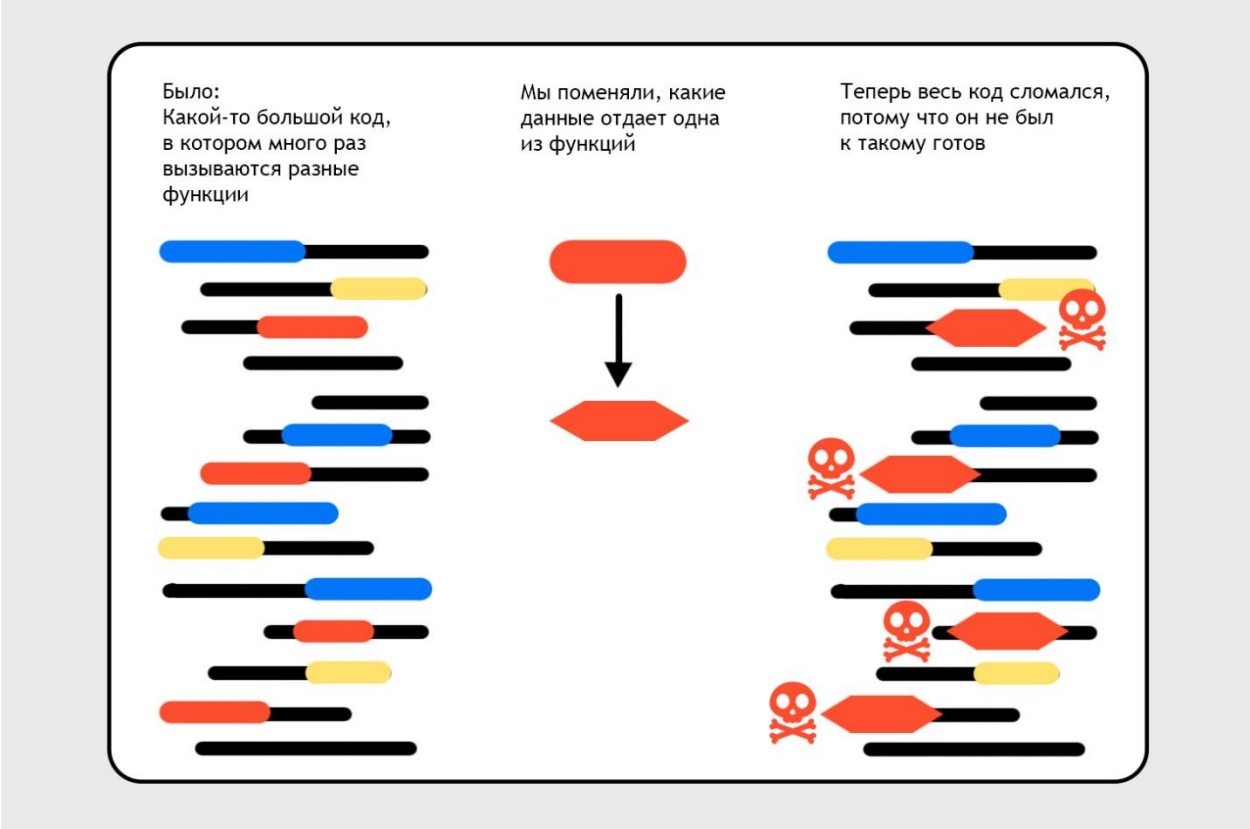
Например, вы пишете функцию «Зарегистрировать пользователя интернет-магазина». Внутри неё вам нужно проверить его электронный адрес. Вы вызываете функцию «Проверить email» внутри функции «Зарегистрировать пользователя», и в зависимости от ответа функции вы либо регистрируете пользователя, либо выводите ошибку. И у вас эта функция встречается ещё в десяти местах. Функции как бы переплетены.



Тут приходит продакт - менеджер и говорит: «Хочу, чтобы пользователь точно знал, в чём ошибка при вводе электронного адреса». Теперь вам нужно научить функцию выдавать не просто true — false, а ещё и код ошибки: например, если в адресе опечатка, то код 01, если адрес спамерский — код 02 и так далее. Это несложно реализовать.

Вы залезаете внутрь этой функции и меняете её поведение: теперь она вместо true — false выдаёт код ошибки, а если ошибки нет — пишет «ОК».

И тут ваш код ломается: все десять мест, которые ожидали от поверяльщика true или false, теперь получают «ОК» и из-за этого ломаются.



Теперь нам нужно:

либо переписывать все функции, чтобы научить их понимать новые ответы поверяльщика адресов;

либо переделать сам поверяльщик адресов, чтобы он остался совместимым со старыми местами, но в нужном вам месте как-то ещё выдавал коды ошибок;

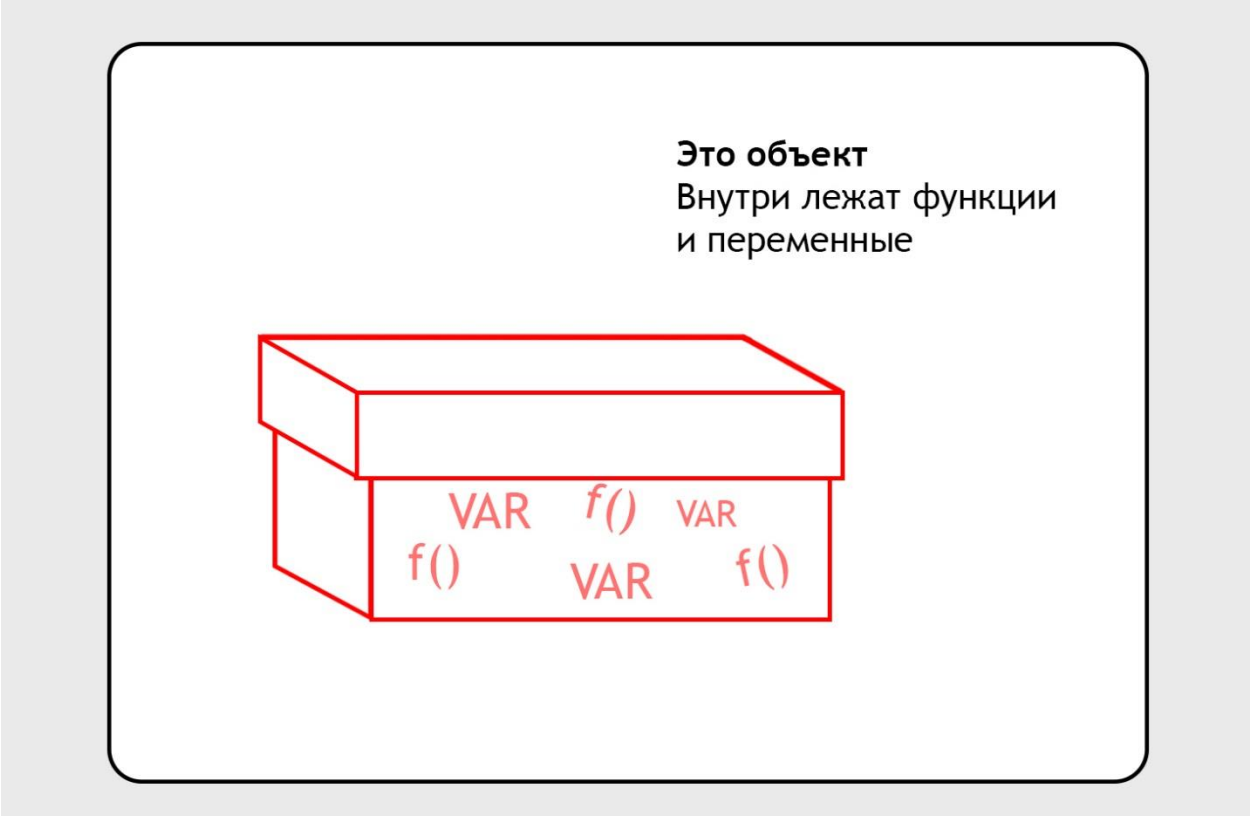
либо написать новый поверяльщик, который выдаёт коды ошибок, а в старых местах использовать старый поверяльщик.

Задача, можно решать за час-другой.

Но теперь представим, что у нас этих функций — сотни. И изменений в них нужно делать десятки в день. И каждое изменение, как правило, заставляет функции вести себя более сложным образом и выдавать более сложный результат. И каждое изменение в одном месте ломает три других места. В итоге у нас будут нарождаться десятки клонированных функций, в которых вы сначала будете разбираться, а потом уже нет.

Основная задача ООП — сделать сложный код проще. Для этого программу разбивают на независимые блоки, которые мы называем объектами.

Объект — это не какая-то космическая сущность. Это всего лишь набор данных и функций — таких же, как в традиционном функциональном программировании. Можно представить, что просто взяли кусок программы и положили его в коробку и закрыли крышку. Вот эта коробка с крышками — это объект.



Программисты договорились, что данные внутри объекта будут называться свойствами, а функции — методами. Но это просто слова, по сути это те же переменные и функции.

Плюсы и минусы ООП

У объектно-ориентированного программирования много плюсов, и именно поэтому этот подход использует большинство современных программистов.

Визуально код становится проще, и его легче читать. Когда всё разбито на объекты и у них есть понятный набор правил, можно сразу понять, за что отвечает каждый объект и из чего он состоит.

Меньше одинакового кода. Если в обычном программировании одна функция считает повторяющиеся символы в одномерном массиве, а другая — в двумерном, то у них большая часть кода будет одинаковой. В ООП это решается наследованием.

Сложные программы пишутся проще. Каждую большую программу можно разложить на несколько блоков, сделать им минимальное наполнение, а потом раз за разом подробно наполнить каждый блок.

Увеличивается скорость написания. На старте можно быстро создать нужные компоненты внутри программы, чтобы получить минимально работающий прототип.

А теперь про минусы:

Сложно понять и начать работать. Подход ООП намного сложнее обычного процедурного программирования — нужно знать много теории, прежде чем будет написана хоть одна строчка кода.

Требует больше памяти. Объекты в ООП состоят из данных, интерфейсов, методов и много другого, а это занимает намного больше памяти, чем простая переменная.

Иногда производительность кода будет ниже. Из-за особенностей подхода часть вещей может быть реализована сложнее, чем могла бы быть. Поэтому бывает такое, что ООП-программа работает медленнее, чем

процедурная (хотя с современными мощностями процессоров это мало кого волнует).

В заключении можно отметить, что объектно-ориентированное программирование требует обдумывания структуры программы и планирования в начале написания кода.

Рассмотрение того, как разбить требования на простые многоразовые классы, которые можно использовать для создания схем экземпляров объектов. В целом реализация ООП позволяет улучшить структуры данных и возможность их повторного использования, что в конечном итоге сэкономит время.

Список использованных литературы

1. Новые педагогические и информационные технологии в системе образования. Под. ред. Н.С.Полат. М: “Академия”, 2005.

2. Дьялонов В.П. Intel. Новейшие информационные технологии. Достижения и люди. - М.: Солон – Пресс, 2004. – 416с.

3. Туропов У. У. и др. Создание группы кафедры «Информационные технологии» в социальной сети «Facebook». – 2019.

4. Хурамова Ф. У. ПРОБЛЕМЫ ВНЕДРЕНИЯ НОВЫХ ТЕХНОЛОГИЙ В УЗБЕКСТАНЕ //Матрица научного познания. – 2020. – №. 3. – С. 57-60.

5. Хурамова Ф. У., Жафярова Ф. С. Совершенствование методов автоматизация обработки данных в условиях текстильного предприятие //Пути повышения результативности современных научных. – 2019. – С. 119.

6. Хурамова Ф. У., Жафярова Ф. С. Улучшение взаимодействий с клиентами в условиях сезонности продажи //Современная наука как основа инновационного прогресса. Актуальные проблемы развития современной системы методов научного познания. – 2019. – С. 63-66.