Программирование на языке Си#

ООП

(Объектно-Ориентированное Программирование)

ООП - это способ организации кода программы.

- **Класс** - это структура данных, которая воедино объединяет данные и функции для работы с ними(поля класса и методы класса);
- **Объект** - это конкретный экземпляр класса, полям которого заданы конкретные значения.

Пример:

В окружающем нас мире существуют различные классы объектов — растения, животные, голодные люди, инопланетяне и так далее.

Пальма - объект (она входит в класс, называемый «растениями»).

Слон – объект (Слон входит в класс под названием «животные» и имеет некоторые важные для нас свойства).

Во-первых, все животные могут двигаться.

Во-вторых, слон очень тяжелый, наверное, тонны две.

В-третьих, у него скверный характер, что тоже может оказаться полезным.

Об объектах можно думать как о существах, которые «живут» в вашей программе и коллективно решают некоторую прикладную задачу.

Вы создаете этих существ, распределяете между ними обязанности и устанавливаете правила их взаимодействия.

В общем случае каждый объект «помнит» необходимую информацию, «умеет» выполнять некоторый набор действий и характеризуется набором свойств.
То, что объект «помнит», хранится в его полях.

То, что объект «умеет делать», реализуется в виде его внутренних процедур и функций, называемых методами.

Свойства объектов аналогичны свойствам, которые мы наблюдаем у обычных предметов. Значения свойств можно устанавливать и читать. Программно свойства реализуются через поля и методы.

Пример:
Объект «кнопка» имеет свойство «цвет».
Значение цвета кнопка запоминает в одном из своих полей.
При изменении значения свойства «цвет» вызывается метод, который перерисовывает кнопку.

Этот пример позволяет сделать важный вывод: свойства имеют первостепенное значение для программиста, использующего объект.
Чтобы понять суть и назначение объекта, вы обязательно должны знать его свойства, иногда — методы, очень редко — поля (объект и сам знает, что с ними делать).

ООП включает три ключевых подхода:

Инкапсуляция — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе.
Наследование — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.
Полиморфизм — свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Инкапсуляция — объект независим: каждый объект устро ен так, что нужные для него данные живут внутри этого объекта, а не где-то снаружи в программе.

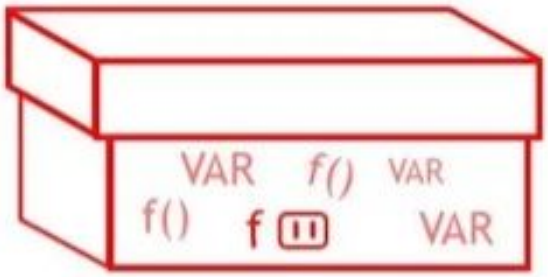
Например, если у меня есть объект «Пользователь», то у меня в нём будут все данные о пользователе: и имя, и адрес, и всё остальное. И в нём же будут методы «Прове рить адрес» или «Подписать на рассылку».

Абстракция — у объекта есть «интерфейс»: у объекта есть методы и свойства, к которым мы можем обратиться извне этого объекта. Так же, как мы можем нажать кнопку на блендере. У блендера есть много всего внутри, что застав ляет его работать, но на главной панели есть только кноп ка. Вот эта кнопка и есть абстрактный интерфейс.

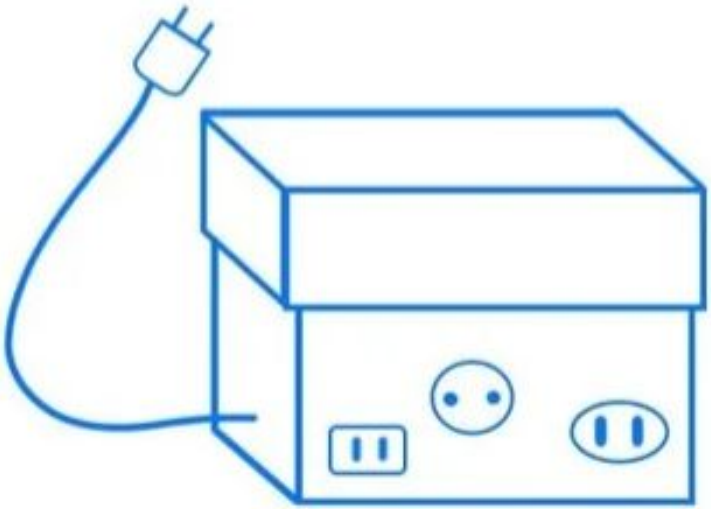
В программе мы можем сказать: «Удалить пользователя». На языке ООП это будет «пользователь.удалить()» — то есть мы обращаемся к объекту «пользователь» и вызыва ем метод «удалить». Нам не так важно, как именно будет происходить удаление: ООП позволяет нам не думать об этом в момент обращения.

Например, над магазином работают два программиста: один пишет модуль заказа, а второй — модуль доставки.

У первого в объекте «заказ» есть метод «отменить». И вот второму нужно из-за доставки отменить заказ. И он спо койно пишет: «заказ.отменить()». Ему неважно, как другой программист будет реализовывать отмену: какие он отправит письма, что запишет в базу данных, какие выве дет предупреждения.



Это как бы интерфейс объекта
Я знаю: что бы в объекте ни произошло, этот интерфейс будет работать всегда одинаково



У других объектов могут быть свои интерфейсы

Наследование — способность к копированию.

ООП позволяет создавать много объектов по образу и подобию другого объекта. Это позволяет не копировать код по двести раз, а один раз нормально написать и потом много раз использовать.

Например, у вас может быть некий идеальный объект «Пользователь»: в нём вы прописываете всё, что может происходить с пользователем.

У вас могут быть свойства: имя, возраст, адрес, номер кар ты. И могут быть методы «Дать скидку», «Проверить заказ», «Найти заказы», «Позвонить».

На основе этого идеального пользователя вы можете создать реального «Покупателя Ивана». У него при созда нии будут все свойства и методы, которые вы задали у идеального покупателя, плюс могут быть какие-то свои, если захотите.

Идеальные объекты программисты называют классами.

Полиморфизм — единый язык общения.

В ООП важно, чтобы все объекты общались друг с другом на понятном им языке. И если у разных объ ектов есть метод «Удалить», то он должен делать именно это и писаться везде одинаково.
Нельзя, чтобы у одного объекта это было «Уда лить», а у другого «Стереть».

Нельзя, чтобы у одного объекта это было «Уда
лить», а у другого «Стереть».

При этом внутри объекта методы могут быть реали
зованы по-разному.
Например, удалить товар — это выдать предупре
ждение, а потом пометить товар в базе данных как
удалённый.
А удалить пользователя — это отменить его покуп
ки, отписать от рассылки и заархивировать историю
его покупок.
События разные, но для программиста это неваж
но. У него просто есть метод «Удалить()», и он ему
доверяет.