**Объектно-ориентированное программирование (ООП)** — это парадигма программирования, которая основана на концепциях **объектов** и **классов**. Основные понятия ООП включают:

Объе́ктно ориенти́рованное программи́рование (сокр. ООП) — методология или стиль программирования на основе описания типов/моделей предметной области и их взаимодействия, представленных порождением из прототипов или как экземпляры классов, которые образуют иерархию наследования.

1. **Объекты**: Экземпляры классов, которые могут содержать данные (поля) и функции (методы), для работы с этими данными.
2. **Наследование**: Механизм, позволяющий создавать новый класс на основе существующего, перенимая его свойства и поведение.
3. **Полиморфизм**: Способность объектов разных классов реагировать на одни и те же методы по-разному.
4. **Инкапсуляция**: Принцип скрытия внутренней реализации объекта и предоставления доступа к данным через публичные методы.

**Понятие класса. Классы и объекты в Java.**

**Класс** —в объектно-ориентированном программировании, модель для создания объектов определённого типа, описывающая их структуру (набор полей и их начальное состояние) и определяющая алгоритмы (функции или методы) для работы с этими объектами.а:

java

复制代码

class Pokemon {

String name;

int level;

Pokemon(String name, int level) {

this.name = name;

this.level = level;

}

void attack() {

System.out.println(name + " атакует!");

}

}

**Объект** — это экземпляр класса, созданный с использованием оператора new:

java

复制代码

Pokemon pikachu = new Pokemon("Пикачу", 5);

pikachu.attack();

**Члены класса: Поля, методы, конструкторы**

* **Поля**: Переменные внутри класса, которые хранят данные объекта.
* **Методы**: Функции, которые описывают поведение объекта.
* **Конструкторы**: Специальные методы, используемые для инициализации объекта.

Пример конструктора:

java

复制代码

Pokemon(String name, int level) {

this.name = name;

this.level = level;

}

**Модификаторы доступа**

Модификаторы доступа определяют видимость членов класса:

* public — доступен везде.
* private — доступен только внутри класса.
* protected — доступен внутри класса, пакета и наследникам.
* По умолчанию (package-private) — доступен внутри класса и пакета.

Пример:

java

复制代码

class Pokemon {

private String name; // Только внутри класса

protected int level; // Доступен в наследниках и пакете

}

**Создание и инициализация объектов. Вызов методов.**

Объекты создаются с помощью ключевого слова new, а их методы вызываются через точку:

java

复制代码w

Pokemon bulbasaur = new Pokemon("Бульбазавр", 10);

bulbasaur.attack();

**Области видимости переменных**

* **Локальные переменные**: Доступны только внутри блока или метода.
* **Поля класса**: Доступны внутри всего класса.

Пример:

java

复制代码

class Example {

int field; // Поле класса

void method() {

int localVariable = 10; // Локальная переменная

}

}

**Модификаторы final и static**

* final — Переменные с этим модификатором нельзя изменить после инициализации. Методы и классы, помеченные как final, не могут быть переопределены или наследованы.

Пример:

java

复制代码

final int MAX\_HP = 100; // Константа

* static — Поля и методы, которые принадлежат классу, а не экземплярам этого класса. Их можно вызывать без создания объекта.

Пример:

java

复制代码

class MathUtil {

static int square(int num) {

return num \* num;

}

}

int result = MathUtil.square(5);

**Пакеты, инструкция import**

**Пакеты** — это способ группировки классов. Пакеты помогают организовать код и избежать конфликтов имен.

Пример объявления пакета:

java

复制代码

package com.example.pokemon;

**Инструкция import** используется для подключения классов из других пакетов:

java

复制代码

import com.example.pokemon.Pokemon;

Пакеты создают логическую структуру программы и способствуют её модульности.

Ясность интерфейса: после инкапсуляции объект раскрывает только ограниченное количество внешних интерфейсов (методов), так что пользователь может взаимодействовать с объектом только через эти интерфейсы, что упрощает взаимодействие между пользователем и объектом.